Unit2_objectsQuestionsPartBJLC9_18_2023_

1. A pair of number cubes is used in a game of chance. Each number cube has six sides, numbered from 1 to 6, inclusive, and there is an equal probability for each of the numbers to appear on the top side (indicating the cube's value) when the number cube is rolled. The following incomplete statement appears in a program that computes the sum of the values produced by rolling two number cubes.

int sum = / * *missing code* * / ;

Which of the following replacements for /* *missing code* */ would best simulate the value produced as a result of rolling two number cubes?

(A) 2 * (int) (Math.random() * 6)

(B) 2 * (int) (Math.random() * 7)

(C) (int) (Math.random() * 6) + (int) (Math.random() * 6)

(D) (int) (Math.random() * 13)

(E) 2 + (int) (Math.random() * 6) + (int) (Math.random() * 6)

2. Consider the following code segment.

```
String oldStr = "ABCDEF";
String newStr = oldStr.substring(1, 3) + oldStr.substring(4);
System.out.println(newStr);
```

What is printed as a result of executing the code segment?

(A) `ABCD`

(B) `BCDE`

(C) `BCEF`

(D) `BCDEF`

(E) `ABCDEF`

Unit2_objectsQuestionsPartBJLC9_18_2023_

3. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves analyzing integer values that are obtained using the `getInt` method in the following `IntegerAnalysis` class. You will write one method in the class.

```
public class IntegerAnalysis
{
 /** Returns an int from simulated user input */
 public static int getInt()
 {  /* implementation not shown */  }

 /** Analyzes n values obtained from the getInt method and returns the
 proportion
   *  of these values that meet the criteria described in part (a)
  *  Precondition: max > 0, n > 0
  */
 public static double analyzeInts(int max, int n)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write method `analyzeInts`, which obtains `n` values using the `getInt` method and returns the proportion of the obtained values that meet all the following criteria.

- The value is greater than `0`
- The value is less than `max`
- The value is divisible by `3`

For example, if `max` is `10` and the values obtained by `getInt` are `6`, `-3`, `5`, `0`, `12`, `3`, `3`, and `9`, then `analyzeInts` should return `0.5` because four of the eight values (`6`, `3`, `3`, and `9`) meet all the criteria.

Complete method `analyzeInts`.

```
/** Analyzes n values obtained from the getInt method and returns the
proportion of
  *  these values that meet the criteria described in part (a)
 *   Precondition: max > 0, n > 0
 */
public static double analyzeInts(int max, int n)
```

(b) A programmer wants to modify the `IntegerAnalysis` class so that in the `analyzeInts` method, the check for divisibility by an integer can vary between method calls. For example, in one call to `analyzeInts`, the method might check for divisibility by `3`, and in another call to `analyzeInts`, the method might check for divisibility by `10`. The programmer wants to implement this change without making any changes to the signature of the `analyzeInts` method or overloading `analyzeInts`.

Write a description of how you would change the `IntegerAnalysis` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

4. Assume that myList is an ArrayList that has been correctly constructed and populated with objects. Which of the following expressions produces a valid random index for myList?

(A) (int) (Math.random () * myList.size ()) - 1

(B) (int) (Math.random () * myList.size ())

(C) (int) (Math.random () * myList.size ()) + 1

(D) (int) (Math.random () * (myList.size () + 1))

(E) Math.random (myList.size ())

Unit2_objectsQuestionsPartBJLC9_18_2023_

5. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves adding apples to a bag until the weight of the bag is within a specified range. The weight of each apple is obtained using the `getApple` method in the following `AppleBagger` class. You will write one method in the class.

```
public class AppleBagger
{
 /** Returns the weight of the next apple to be added to the bag, with a
 different weight
  *   being returned with each call
  */
 public static double getApple()
 {  /* implementation not shown */  }

 /** Returns the number of apples that are added to a bag, as described
 in part (a)
  *  Precondition: 0 < allowedVariation < targetWeight
  */
 public static int bagApples(double targetWeight,
        double allowedVariation)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the method `bagApples`, which obtains the weights of apples to be added to a bag using calls to the `getApple` method and returns the number of apples that are added to the bag until the combined weight exceeds `targetWeight` minus `allowedVariation`.

For example, if `targetWeight` is `10.0` and `allowedVariation` is `0.5`, the `bagApples` method will return the number of apples that are added until the combined weight of the apples exceeds `9.5`.

Complete method `bagApples`.

```
/** Returns the number of apples that are added to a bag, as described in
part (a)
 *  Precondition: 0 < allowedVariation < targetWeight
 */
public static int bagApples(double targetWeight,
       double allowedVariation)
```

(b) A programmer wants to modify the `AppleBagger` class so that the allowed variation in the `bagApples` method is always equal to 20 percent of the weight of the first apple added to the bag during a call to `bagApples`, rather than a parameter passed to the method.

Write a description of how you would change the `AppleBagger` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

6. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

   Notes:

   - Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
   - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

   This question involves reasoning about strings made up of uppercase letters. You will implement two related methods that appear in the same class (not shown). The first method takes a single string parameter and returns a scrambled version of that string. The second method takes a list of strings and modifies the list by scrambling each entry in the list. Any entry that cannot be scrambled is removed from the list.

   a. Write the method scrambleWord, which takes a given word and returns a string that contains a scrambled version of the word according to the following rules.

      - The scrambling process begins at the first letter of the word and continues from left to right.
      - If two consecutive letters consist of an "A" followed by a letter that is not an "A", then the two letters are swapped in the resulting string.
      - Once the letters in two adjacent positions have been swapped, neither of those two positions can be involved in a future swap.

      The following table shows several examples of words and their scrambled versions.

`word`	Result returned by `scrambleWord(word)`
"TAN"	"TNA"
"ABRACADABRA"	"BARCADABARA"
"WHOA"	"WHOA"
"AARDVARK"	"ARADVRAK"
"EGGS"	"EGGS"
"A"	"A"
""	""

      Complete method scrambleWord below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
/** Scrambles a given word.
 *  @param word the word to be scrambled
 *  @return the scrambled word (possibly equal to word)
 *  Precondition: word is either an empty string or contains only uppercase letters.
 *  Postcondition: the string returned was created from word as follows:
 *    - the word was scrambled, beginning at the first letter and continuing from left to right
 *    - two consecutive letters consisting of "A" followed by a letter that was not "A" were swapped
 *    - letters were swapped at most once
 */
public static String scrambleWord(String word)
```

b. Write the method scrambleOrRemove, which replaces each word in the parameter wordList with its scrambled version and removes any words that are unchanged after scrambling. The relative ordering of the entries in wordList remains the same as before the call to scrambleOrRemove.

The following example shows how the contents of wordList would be modified as a result of calling scrambleOrRemove.

Before the call to scrambleOrRemove:

	0	1	2	3	4
wordList	"TAN"	"ABRACADABRA"	"WHOA"	"APPLE"	"EGGS"

After the call to scrambleOrRemove:

	0	1	2
wordList	"TNA"	"BARCADABARA"	"PAPLE"

Assume that scrambleWord is in the same class as scrambleOrRemove and works as specified, regardless of what you wrote in part (a).

Complete method scrambleOrRemove below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
/** Modifies wordList by replacing each word with its scrambled
 *  version, removing any words that are unchanged as a result of scrambling.
 *  @param wordList the list of words
 *  Precondition: wordList contains only non-null objects
 *  Postcondition:
 *    - all words unchanged by scrambling have been removed from wordList
 *    - each of the remaining words has been replaced by its scrambled version
 *    - the relative ordering of the entries in wordList is the same as it was
 *          before the method was called
 */
public static void scrambleOrRemove(List<String> wordList)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

7. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves generating a `String` based on a numeric value. You will write the `buildString` method of the following `Converter` class.

```
public class Converter
{
 /** Returns a String based on the single-digit integer num
  *  Precondition: num is a single-digit integer.
  */
 public static String convertToString(int num)
 {  /* implementation not shown */  }

 /** Returns a string based on an integer input value, as described in
 part (a)
  *  Precondition: input > 0
  */
 public static String buildString(int input)
 {  /* to be implemented in part (a) */  }

 // There may be variables and other methods that are not shown.
}
```

(a) The `buildString` method takes an integer value as input, obtains strings based on each digit of the input, and returns the concatenated strings.

A helper method, `convertToString`, has been provided. The `convertToString` method returns a string based on a single-digit integer input value. For example, the method call `convertToString(7)` might return the `String "paper"`.

The strings returned by `convertToString` should appear in the `String` returned by `buildString` in the same order that the digits appeared in the original input value.

For example, assume that the following calls to `convertToString` are made from within the `Converter`

class and that `convertToString(1)` returns `"apple"`, `convertToString(2)` returns `"orange"`, and `convertToString(3)` returns `"banana"`.

Then the method call `Converter.buildString(3)` should return `"banana"`, the method call `Converter.buildString(123)` should return `"appleorangebanana"`, and the method call `Converter.buildString(321)` should return `"bananaorangeapple"`.

Complete method `buildString`. You must use `convertToString` appropriately to receive full credit.

```
/** Returns a string based on an integer input value, as described in
part (a)
 *  Precondition: input > 0
 */
public static String buildString(int input)
```

(b) A programmer wants to modify the `Converter` class so that the `buildString` method stores the value it returns so that the value is available to other methods in the `Converter` class. The programmer would like to implement this change without making any changes to the signature of the `buildString` method or overloading `buildString`.

Write a description of how you would change the `Converter` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

8. Consider the following method, which is intended to calculate and return the expression $\sqrt{\frac{(x+y)^2}{|a-b|}}$.

```
public double calculate(double x, double y, double a, double b)
{
     return /* missing code */;
}
```

Which of the following can replace `/* missing code */` so that the method works as intended?

(A) `Math.sqrt(x ^ 2, y ^ 2, a - b)`

(B) `Math.sqrt((x + y) ^ 2) / Math.abs(a, b)`

(C) `Math.sqrt((x + y) ^ 2 / Math.abs(a - b))`

(D) `Math.sqrt(Math.pow(x + y, 2) / Math.abs(a, b))`

(E) `Math.sqrt(Math.pow(x + y, 2) / Math.abs(a - b))`

Unit2_objectsQuestionsPartBJLC9_18_2023_

9. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves analyzing a salesperson's sales. Sales amounts are obtained using the `getSales` method in the following `SalesSimulator` class. You will write one method in the class.

```
public class SalesSimulator
{
 /** Simulates and returns the sales, in dollars, made by a salesperson
 on a particular
  *  day
  */
 public static int getSales()
 {  /* implementation not shown */  }

 /** Analyzes sales for numDays days obtained from the getSales method
 and
   *   returns the total bonus earned by the salesperson, in dollars, as
 described in part (a)
  *  Precondition: goal > 0, numDays > 0
  */
 public static int calculateBonus(int goal, int numDays)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `SalesSimulator` method `calculateBonus`, which obtains the daily sales of a salesperson and applies a bonus based on how close to meeting or exceeding the daily sales goal the salesperson came. It obtains daily sales for `numDays` days using the `getSales` method. For each day, if the daily sales are at least 80 percent of `goal` but less than `goal`, the salesperson receives a $50 bonus. The salesperson receives a $75 bonus if the daily sales are greater than or equal to `goal`. Method `calculateBonus` returns the total bonus received, in dollars, over all days.

The following table shows the bonuses received as a result of the method call

`SalesSimulator.calculateBonus(200, 3)`.

Day	`getSales()` Return Value	Daily Bonus	Explanation
1	`180`	`50`	Because $180 is greater than or equal to 80 percent of the goal ($160) but is less than the goal, the salesperson earns a $50 bonus.
2	`150`	`0`	Because $150 is less than 80 percent of the goal ($160), the salesperson earns no bonus.
3	`200`	`75`	Because $200 is greater than or equal to the goal, the salesperson earns a $75 bonus.

For the sales shown in the table above, `SalesSimulator.calculateBonus(200, 3)` should return the total bonus `125`.

Complete method `calculateBonus`.

```
/** Analyzes sales for numDays days obtained from the getSales method and returns
  *   the total bonus earned by the salesperson, in dollars, as described in part (a)
 *  Precondition: goal > 0, numDays > 0
 */
public static int calculateBonus(int goal, int numDays)
```

(b) A programmer wants to modify the `SalesSimulator` class so that the daily goal is maintained in a variable with a value that cannot be changed once it is initialized.

Write a description of how you would change the `SalesSimulator` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

10. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

The `APCalendar` class contains methods used to calculate information about a calendar. You will write two methods of the class.

```
public class APCalendar
{

    /** Returns true if year is a leap year and false otherwise */
    private static boolean isLeapYear(int year)
    {  /* implementation not shown */  }

    /** Returns the number of leap years between year1 and year2,
inclusive.
 *  Precondition: 0 <= year1 <= year2
 */
    public static int numberOfLeapYears(int year1, int year2)
    {  /* to be implemented in part (a) */  }

    /** Returns the value representing the day of the week for the
first day of year,
 *  where 0 denotes Sunday, 1 denotes Monday, ..., and 6 denotes
 Saturday.
 */
    private static int firstDayOfYear(int year)
    {  /* implementation not shown */  }

    /** Returns n, where month, day, and year specify the nth day of
the year.
 *  Returns 1 for January 1 (month = 1, day = 1) of any year.
 *  Precondition: The date represented by month, day, year is a valid
 date.
 */
    private static int dayOfYear(int month, int day, int year)
    {  /* implementation not shown */  }

    /** Returns the value representing the day of the week for the
given date
 *  (month, day, year), where 0 denotes Sunday, 1 denotes Monday,
```

```
...,
*  and 6 denotes Saturday.
*  Precondition: The date represented by month, day, year is a valid
date.
*/
    public static int dayOfWeek(int month, int day, int year)
    {  /* to be implemented in part (b) */  }

    // There may be instance variables, constructors, and other
methods not shown.

}
```

Write the static method `numberOfLeapYears`, which returns the number of leap years between `year1` and `year2`, inclusive.

In order to calculate this value, a helper method is provided for you.

`isLeapYear(year)` returns `true` if `year` is a leap year and `false` otherwise.

Complete method `numberOfLeapYears` below. You must use `isLeapYear` appropriately to receive full credit.

```
   /** Returns the number of leap years between year1 and year2,
inclusive.
     *  Precondition: 0 <= year1 <= year2
     */
   public static int numberOfLeapYears(int year1, int year2)
```

(b) Write the static method `dayOfWeek`, which returns the integer value representing the day of the week for the given date `(month, day, year)`, where `0` denotes Sunday, `1` denotes Monday, ..., and `6` denotes Saturday. For example, 2019 began on a Tuesday, and January 5 is the fifth day of 2019. As a result, January 5, 2019, fell on a Saturday, and the method call `dayOfWeek(1, 5, 2019)` returns `6`.

As another example, January 10 is the tenth day of 2019. As a result, January 10, 2019, fell on a Thursday, and the method call `dayOfWeek(1, 10, 2019)` returns `4`.

In order to calculate this value, two helper methods are provided for you.

- `firstDayOfYear(year)` returns the integer value representing the day of the week for the first day of `year`, where `0` denotes Sunday, `1` denotes Monday, ..., and `6` denotes Saturday. For example, since 2019 began on a Tuesday, `firstDayOfYear(2019)` returns `2`.
- `dayOfYear(month, day, year)` returns n, where `month`, `day`, and `year` specify the nth day of the year. For the first day of the year, January 1 `(month = 1, day = 1)`, the value `1` is returned. This method accounts for whether `year` is a leap year. For example,

Unit2_objectsQuestionsPartBJLC9_18_2023_

`dayOfYear(3, 1, 2017)` returns `60`, since 2017 is not a leap year, while

`dayOfYear(3, 1, 2016)` returns `61`, since 2016 is a leap year.

Class information for this question

```
public class APCalendar
private static boolean isLeapYear(int year)
public static int numberOfLeapYears(int year1, int year2)
private static int firstDayOfYear(int year)
private static int dayOfYear(int month, int day, int year)
public static int dayOfWeek(int month, int day, int year)
```

Complete method `dayOfWeek` below. You must use `firstDayOfYear` and `dayOfYear` appropriately to receive full credit.

```
   /** Returns the value representing the day of the week for the given
date
     *  (month, day, year), where 0 denotes Sunday, 1 denotes Monday,
 ...,
     *  and 6 denotes Saturday.
     *  Precondition: The date represented by month, day, year is a
 valid date.
     */
   public static int dayOfWeek(int month, int day, int year)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

11. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves manipulating strings that are made up of uppercase letters. The `Puzzle` class contains methods used to modify keys. You will write one method in this class. A partial declaration of the `Puzzle` class is shown below.

```
public class Puzzle
{
  /** Returns a String based on a key and a letter, as described in part
  (a)
    * Precondition: letter consists of one uppercase letter.
    *                key has at least 2 letters and all letters are
  uppercase and unique.
   */
  public static String changeKey(String letter, String key)
  {  /* to be implemented in part (a) */  }

  // There may be variables and methods that are not shown.
}
```

(a) Write the `Puzzle` method `changeKey`, which uses the parameter `letter` to return a string based on the parameter `key`. The parameter `letter` contains a single uppercase letter. The parameter `key` has at least two letters and all letters in `key` are uppercase and unique. A sample key is `"ABC"`.

The following describes the string that should be returned by the method.

If the letter is contained in the key, the method should return a new string containing all letters in the key except `letter`, in their original order.
If the letter does not appear in the key, the method should return the letter.

The following table shows the results of several calls to `changeKey`.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Method Call	Return Value
`Puzzle.changeKey("G", "GXAD")`	`"XAD"`
`Puzzle.changeKey("D", "GXAD")`	`"GXA"`
`Puzzle.changeKey("A", "GXAD")`	`"GXD"`
`Puzzle.changeKey("M", "GXAD")`	`"M"`

Complete method `changeKey`.

```
/** Returns a String based on a key and a letter, as described in part
(a)
 *   Precondition: letter consists of one uppercase letter.
 *                 key has at least 2 letters and all letters are
uppercase and unique.
 */
public static String changeKey(String letter, String key)
```

(b) The programmer would like to modify the `Puzzle` class to keep track of how many times a call to `changeKey` results in a change to a key. Any time a call to `changeKey` results in a change to the key, the count of key modifications should be increased by one. The programmer would like to implement this change without making any changes to the signature of the `changeKey` method or overloading `changeKey`.

Write a description of how you would change the `Puzzle` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves the use of *check digits*, which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
     /** Returns the check digit for num, as described in part (a).
       * Precondition: The number of digits in num is between one and six,
   inclusive.
       * num >= 0
       */
     public static int getCheck(int num)
     {
         /* to be implemented in part (a) */
     }

     /** Returns true if numWithCheckDigit is valid, or false otherwise, as
  described in part (b).
      * Precondition: The number of digits in numWithCheckDigit is between two
  and seven, inclusive.
       * numWithCheckDigit >= 0
      */
     public static boolean isValid(int numWithCheckDigit)
     {
         /* to be implemented in part (b) */
     }

     /** Returns the number of digits in num. */
     public static int getNumberOfDigits(int num)
     {
         /* implementation not shown */
     }

     /** Returns the nth digit of num.
       * Precondition: n >= 1 and n <= the number of digits in num
       */
     public static int getDigit(int num, int n)
     {
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
        /* implementation not shown */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

12. **(a)** Write the `getCheck` method, which computes the check digit for a number according to the following rules.

Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
Add the products calculated in the previous step.
Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

The sum to calculate is
$(2x7) + (8x6) + (3x5) + (4x4) + (1x3) + (5x2) = 14 + 48 + 15 + 16 + 3 + 10 = 106.$
The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value `6`.

Example 2, where num has the value 2183

The sum to calculate is $(2x7) + (1x6) + (8x5) + (3x4) = 14 + 6 + 40 + 12 = 72.$
The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value `2`.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

`getNumberOfDigits` returns the number of digits in its `int` parameter.
`getDigit` returns the `n`th digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	`6`	The number `283415` has 6 digits.
`getDigit(283415, 1)`	`2`	The first digit of `283415` is `2`.
`getDigit(283415, 5)`	`1`	The fifth digit of `283415` is `1`.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 * Precondition: The number of digits in num is between one and six,
inclusive.
 * num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost

digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	`2`	The check digit for `159` is `2`.
`isValid(1592)`	`true`	The number `1592` is a valid combination of a number (`159`) and its check digit (`2`).
`isValid(1593)`	`false`	The number `1593` is not a valid combination of a number (`159`) and its check digit (`3`) because `2` is the check digit for `159`.

Complete method `isValid` below. Assume that `getCheck` works as specified, regardless of what you wrote in part (a). You must use `getCheck` appropriately to receive full credit.

```
/** Returns true if numWithCheckDigit is valid, or false otherwise, as
described in part (b).
 * Precondition: The number of digits in numWithCheckDigit is between two
 and seven, inclusive.
 * numWithCheckDigit >= 0
 */
public static boolean isValid(int numWithCheckDigit)
```

13. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves guessing strings that may or may not appear in a secret phrase. Guesses are obtained using the `getGuess` method in the following `StringGuess` class. You will write one method in the class.

```
public class StringGuess
{
  /** Returns a guess string. Assume that this method never returns the
  same string twice.
   */
  public static String getGuess()
  {  /* implementation not shown */  }

  /** Returns the proportion of guesses that appear in phrase, as
  described in part (a)
   *  Precondition: num > 0, phrase.length() > 0
   */
  public static double checkGuesses(String phrase, int num)
  {  /* to be implemented in part (a) */  }

  // There may be variables and methods that are not shown.
}
```

(a) Write the `StringGuess` method `checkGuesses`, which obtains `num` guesses using the `getGuess` method. Method `checkGuesses` returns the proportion of obtained guesses that appear in `phrase`.

For example, consider the call `StringGuess.checkGuesses("open_sesame", 4)`. If `getGuess` returns the values `"same"`, `"hello"`, `"open"`, and `"car"`, then the method call should return `0.5` because, of the four values returned by `getGuess`, two appear in `phrase`.

You must use `getGuess` appropriately to receive full credit.

Complete method `checkGuesses`.

```
/** Returns the proportion of guesses that appear in phrase, as described
```

```
in part (a)
 *  Precondition: num > 0, phrase.length() > 0
 */
public static double checkGuesses(String phrase, int num)
```

(b) A programmer wants to modify the `StringGuess` class so that the number of guesses analyzed by the `checkGuesses` method is a random value between `1` and `100`, inclusive, and is generated in the `StringGuess` class. For example, in one call to `checkGuesses`, the number of guesses to analyze might be `12`, and in another call to `checkGuesses`, the number of guesses to analyze might be `77`. Each value between `1` and `100` must have an equal chance of being used as the number of guesses.

Write a description of how you would change the `StringGuess` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

14. Consider the following method.

```
/** Precondition: Strings one and two have the same length. */
public static String combine(String one, String two)
{
     String res = "";
     for (int k = 0; k < one.length(); k++)
     {
         if (one.substring(k, k + 1).equals(two.substring(k, k + 1)))
         {
            res += one.substring(k, k + 1);
         }
         else
         {
            res += "0";
         }
     }
     return res;
}
```

What is returned as a result of the call `combine("10110", "01100")` ?

(A) `"00000"`

(B) `"00100"`

(C) `"00101"`

(D) `"10110"`

(E) `"11011"`

15. Consider the following code segment.

```
String temp = "comp";
System.out.print(temp.substring(0) + " " +
            temp.substring(1) + " " +
            temp.substring(2) + " " +
            temp.substring(3));
```

What is printed when the code segment is executed?

(A) `comp`

(B) `c o m p`

(C) `comp com co c`

(D) `comp omp mp p`

(E) `comp comp comp comp`

16. The following statement assigns an integer value to `x`.

```
int x = (int)(Math.random() * 5) + 10;
```

Consider the statement that would result if the positions of `5` and `10` were swapped in the preceding statement and the resulting integer were assigned to `y`.

```
int y = (int)(Math.random() * 10) + 5;
```

Which of the following are true statements about how the possible values assigned to `y` differ from the possible values assigned to `x` ?

I. There are more possible values of `x` than there are possible values of `y`.
II. There are more possible values of `y` than there are possible values of `x`.
III. The value assigned to `x` can sometimes be the same as the value assigned to `y`.

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III

Unit2_objectsQuestionsPartBJLC9_18_2023_

17. Consider the following code segment.

```
String str = "CompSci";
System.out.println(str.substring(0, 3));
int num = str.length();
```

What is the value of `num` when the code segment is executed?

(A) `3`

(B) `4`

(C) `5`

(D) `6`

(E) `7`

Unit2_objectsQuestionsPartBJLC9_18_2023_

18. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

6. This question involves computing an integer based on two input strings. You will write the `compute` method of the following `Operator` class.

```
public class Operator
{
 /** Returns an integer value based on the input string */
 public static int transform(String str)
 {  /* implementation not shown */  }

 /** Returns an integer based on two input strings, as described in part
 (a)
  *   Precondition: input and op are not null.
  *              The length of input is even.
  *             op is either "$$", "^^", or "##".
  */
 public static int compute(String input, String op)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `compute` method of the `Operator` class. The method calculates an integer value based on an input string with even length. The first half and the second half of `input` are transformed into numeric values by calls to `transform`. The numeric values are then used in a calculation based on the mathematical operation represented by the string, `op`. The method returns the calculated result.

The helper method `transform` has been provided. The method returns an integer value based on its `String` parameter. For example, the method call `transform("pear")` might return the `int` value `3`.

The mathematical operations represented by `op` are described in the following table. Assume that `value1` is the numeric value obtained by calling `transform` on the first half of `input` and `value2` is the numeric value obtained by calling `transform` on the second half of `input`.

Value of `op`	Mathematical Operation	Description
`"$$"`	Addition	`value1` plus `value2`
`"^^"`	Multiplication	`value1` times `value2`
`"##"`	Subtraction	`value1` minus `value2`

For example, assume that calls to `transform` are made from the `Operator` class and that `transform("orang")` returns `2` and `transform("epear")` returns `6`. Then the method call `Operator.compute("orangepear", "$$")` should return `8`.

As another example, the method call `Operator.compute("orangepear", "##")` should return `-4`.

Complete method `compute`. You must use `transform` appropriately to receive full credit.

```
/** Returns an integer based on two input strings, as described in part
(a)
 *   Precondition: input and op are not null.
 *         The length of input is even.
 *         op is either "$$", "^^", or "##".
 */
public static int compute(String input, String op)
```

(b) A programmer wants to modify the `Operator` class so that the `compute` method has a single `String` parameter containing both `input` and `op`.

Write a description of how you would change the `Operator` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

19. Consider the following code segment.

```
String str = "0";
str += str + 0 + 8;
System.out.println(str);
```

What is printed as a result of executing the code segment?

(A) `8`

(B) `08`

(C) `008`

(D) `0008`

(E) Nothing is printed, because numerical values cannot be added to a `String` object.

20. Consider the following code segment.

```
int one = 1;
int two = 2;
String zee = "Z";
System.out.println(one + two + zee);
```

What is printed as a result of executing the code segment?

(A) `12Z`

(B) `3Z`

(C) `12zee`

(D) `3zee`

(E) `onetwozee`

21. Consider the following instance variable and method.

```
private List<String> animals;

public void manipulate()
{
  for (int k = animals.size() - 1; k > 0; k--)
  {
    if (animals.get(k).substring(0, 1).equals("b"))
    {
      animals.add(animals.size() - k, animals.remove(k));
    }
  }
}
```

Assume that animals has been instantiated and initialized with the following contents.

```
["bear", "zebra", "bass", "cat", "koala", "baboon"]
```

What will the contents of animals be as a result of calling manipulate?

Unit2_objectsQuestionsPartBJLC9_18_2023_

(A) ["baboon", "zebra", "bass", "cat", "bear", "koala"]
(B) ["bear", "zebra", "bass", "cat", "koala", "baboon"]
(C) ["baboon", "bear", "zebra", "bass", "cat", "koala"]
(D) ["bear", "baboon", "zebra", "bass", "cat", "koala"]
(E) ["zebra", "cat", "koala", "baboon", "bass", "bear"]

22. Consider the following method, which is intended to return true if at least one of the three strings s1, s2, or s3 contains the substring "art". Otherwise, the method should return false.

```
public static boolean containsArt(String s1, String s2, String s3)
{

  String all = s1 + s2 + s3;

  return (all.indexOf("art") != -1);

}
```

Which of the following method calls demonstrates that the method does not work as intended?

(A) containsArt ("rattrap", "similar", "today")
(B) containsArt ("start", "article", "Bart")
(C) containsArt ("harm", "chortle", "crowbar")
(D) containsArt ("matriculate", "carat", "arbitrary")
(E) containsArt ("darkroom", "cartoon", "articulate")

23. Consider the following method, which is intended to return the element of a 2-dimensional array that is closest in value to a specified number, val.

```
/** @return the element of 2-dimensional array mat whose value is closest to val */
public double findClosest(double[][] mat, double val)
{
  double answer = mat[0][0];
  double minDiff = Math.abs(answer - val);
  for (double[] row : mat)
  {
    for (double num : row)
    {
      if ( /* missing code */ )
      {
        answer = num;
        minDiff = Math.abs(num - val);
      }
    }
  }
  return answer;
}
```

Which of the following could be used to replace / * *missing code* * / so that findClosest will work as intended?

(A) val - row [num] < minDiff

(B) Math.abs (num - minDiff) < minDiff

(C) val - num < 0.0

(D) Math.abs (num - val) < minDiff

(E) Math.abs (row [num] - val) < minDiff

24. Consider the following method.

```
public String mystery(String input)
{
  String output = "";

  for (int k = 1; k < input.length(); k = k + 2)
  {
    output += input.substring(k, k + 1);
  }

  return output;
}
```

What is returned as a result of the call mystery("computer") ?

(A) "computer"
(B) "cmue"
(C) "optr"
(D) "ompute"
(E) Nothing is returned because an IndexOutOfBoundsException is thrown.

25. Consider the following method.

```
public static String scramble(String word, int howFar)
{
  return word.substring(howFar + 1, word.length()) +
         word.substring(0, howFar);
}
```

What value is returned as a result of the call scramble("compiler", 3)?

(A) "compiler"
(B) "pilercom"
(C) "ilercom"
(D) "ilercomp"
(E) No value is returned because an IndexOutOfBoundsException will be thrown.

26. Consider the following method.

```
public static boolean mystery(String str)
{
  String temp = "";

  for (int k = str.length(); k > 0; k--)
  {
    temp = temp + str.substring(k - 1, k);
  }

  return temp.equals(str);
}
```

Which of the following calls to mystery will return true?

(A) mystery ("no")
(B) mystery ("on")
(C) mystery ("nnoo")
(D) mystery ("nono")
(E) mystery ("noon")

27. Consider the following method, which is intended to count the number of times the letter `"A"` appears in the string `str`.

```
public static int countA(String str)
{
     int count = 0;

     while (str.length() > 0)
     {
         int pos = str.indexOf("A");
         if (pos >= 0)
         {
            count++;
            /* missing code */
         }
         else
         {
            return count;
         }
     }
     return count;
}
```

Which of the following should be used to replace `/* missing code */` so that method `countA` will work as intended?

(A) `str = str.substring(0, pos);`

(B) `str = str.substring(0, pos + 1);`

(C) `str = str.substring(pos - 1);`

(D) `str = str.substring(pos);`

(E) `str = str.substring(pos + 1);`

Unit2_objectsQuestionsPartBJLC9_18_2023_

28. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves analyzing numbers that are obtained using the `getNumber` method in the following `NumberChecker` class. You will write one method in the class.

```
public class NumberChecker
{
 /** Returns an int value to be analyzed */
 public static int getNumber()
 {  /* implementation not shown */  }

 /** Returns true if x is a target number and returns false otherwise */
 public static boolean isTarget(int x)
 {  /* implementation not shown */  }

 /** Analyzes values obtained using the getNumber method, as described
 in part (a)
  *  Precondition: 0 < n < max
  */
 public static int countNumbers(int n, int max)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write method `countNumbers`, which obtains values using the `getNumber` method and returns the number of calls to `getNumber` that are made until `n` values are obtained that meet both of the following criteria.

- Is NOT a target number, as determined by the `isTarget` method
- Is divisible by `3`

If `max` calls are made to `getNumber` without obtaining `n` values that meet the criteria, `-1` is returned.

Unit2_objectsQuestionsPartBJLC9_18_2023_

A helper method, `isTarget`, has been provided. The method returns `true` if its integer parameter is a target number and returns `false` otherwise. You must use `isTarget` appropriately to receive full credit.

Complete method `countNumbers`.

```
/** Analyzes values obtained using the getNumber method, as described in
part (a)
 *  Precondition: 0 < n < max
 */
public static int countNumbers(int n, int max)
```

(b) A programmer wants to modify the `NumberChecker` class so that in the `countNumbers` method, the check for divisibility by an integer can vary between method calls. For example, in one call to `countNumbers`, the method might check for divisibility by 3, and in another call to `countNumbers`, the method might check for divisibility by 2.

The programmer would like to implement this change without making any changes to the signature of the `countNumbers` method or overloading `countNumbers`.

Write a description of how you would change the `NumberChecker` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

29. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves analyzing strings. You will write the `countRepeat` method of the following `StringAnalyzer` class.

```
public class StringAnalyzer
{
 /** Returns a count of how many times smallStr occurs in largeStr, as
  *   described in part (a)
  *   Precondition: largeStr is not null; smallStr is not null.
  *                 The length of smallStr is less than the length of
 largeStr.
  */
 public static int countRepeat(String largeStr, String smallStr)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) The `countRepeat` method is used to count and return the number of times the string parameter `smallStr` appears in the string parameter `largeStr`. In the case of overlapping occurrences of `smallStr`, only the first occurrence is included in the overall count, as shown in the second and third calls to `countRepeat` in the following table of examples.

Call to `countRepeat`	**Return Value**
`StringAnalyzer.countRepeat("BAAB", "AA")`	1
`StringAnalyzer.countRepeat("AAAAA", "AA")`	2
`StringAnalyzer.countRepeat("AABABABAA", "ABA")`	2
`StringAnalyzer.countRepeat("ABBAABB", "ABA")`	0

Complete method `countRepeat`.

```
/** Returns a count of how many times smallStr occurs in largeStr,
 *   as described in part (a)
 *   Precondition: largeStr is not null; smallStr is not null.
 *                 The length of smallStr is less than the length of
largeStr.
 */
public static int countRepeat(String largeStr, String smallStr)
```

(b) A programmer would like to modify the `StringAnalyzer` class to limit the length of the large string that can be passed as input to `countRepeat`. The limit could vary between calls to `countRepeat`. A call to `countRepeat` with a value of `largeStr` that is longer than the maximum length would return `-1`. The programmer would like to implement this change without making any changes to the signature of the `countRepeat` method or overloading `countRepeat`.

Write a description of how you would change the `StringAnalyzer` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

30.

- **Directions:** SHOW ALL YOUR WORK, REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN Java.
- **Note:** Unless otherwise noted in the question, assume that parameters in method calls are not *null* and that methods are called only when their preconditions are satisfied.

The following class WordList is designed to store and manipulate a list of words. The incomplete class declaration is shown below. You will be asked to implement two methods.

```
public class WordList
{
  private ArrayList myList;  // contains Strings made up of letters

  // postcondition: returns the number of words in this WordList that
  //                are exactly len letters long
  public int numWordsOfLength(int len)
  {  /* to be implemented in part (a) */  }

  // postcondition: all words that are exactly len letters long
  //                have been removed from this WordList, with the
  //                order of the remaining words unchanged
  public void removeWordsOfLength(int len)
  {  /* to be implemented in part (b) */  }

  // ... constructor and other methods not shown
}
```

(a) Write the WordList method numWordsOfLength. Method numWordsOfLength returns the number of words in the WordList that are exactly len letters long. For example, assume that the instance variable myList of the WordList animals contains the following.

["cat", "mouse", "frog", "dog", "dog"]

The table below shows several sample calls to numWordsOfLength.

Call	Result returned by call
`animals.numWordsOfLength(4)`	1
`animals.numWordsOfLength(3)`	3
`animals.numWordsOfLength(2)`	0

Complete method numWordsofLength below.

// postcondition: returns the number of words in this WordList that

// are exactly len letters long

Unit2_objectsQuestionsPartBJLC9_18_2023_

public int numWordsOfLength(int len)

(b) Write the WordList method removeWordsOfLength.Method removeWordsOfLength removes all words from the WordList that are exactly len letters long, leaving the order of the remaining words unchanged. For example, assume that the instance variable myList of the WordList animals contains the following.

["cat", "mouse", "frog", "dog", "dog"]

The table below shows a sequence of calls to the removeWordsOfLength method.

Call	myList after the call
`animals.removeWordsOfLength(4);`	`["cat", "mouse", "dog", "dog"]`
`animals.removeWordsOfLength(3);`	`["mouse"]`
`animals.removeWordsOfLength(2);`	`["mouse"]`

Complete method removeWordsOfLength below.

```
// postcondition: all words that are exactly len letters long
// have been removed from this WordList, with the
// order of the remaining words unchanged public void removeWordsOfLength(int len)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

31. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

Consider a guessing game in which a player tries to guess a hidden word. The hidden word contains only capital letters and has a length known to the player. A guess contains only capital letters and has the same length as the hidden word.

After a guess is made, the player is given a hint that is based on a comparison between the hidden word and the guess. Each position in the hint contains a character that corresponds to the letter in the same position in the guess. The following rules determine the characters that appear in the hint.

If the letter in the guess is …	the corresponding character in the hint is
also in the same position in the hidden word,	the matching letter
also in the hidden word, but in a different position,	"+"
not in the hidden word,	"*"

The HiddenWord class will be used to represent the hidden word in the game. The hidden word is passed to the constructor. The class contains a method, getHint, that takes a guess and produces a hint.

For example, suppose the variable puzzle is declared as follows.

HiddenWord puzzle = new HiddenWord("HARPS");

The following table shows several guesses and the hints that would be produced.

If the letter in the guess is …	the corresponding character in the hint is
also in the same position in the hidden word,	the matching letter
also in the hidden word, but in a different position,	"+"
not in the hidden word,	"*"

Write the complete HiddenWord class, including any necessary instance variables, its constructor, and the method, getHint, described above. You may assume that the length of the guess is the same as the length of the hidden word.

32. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

Consider the following partial declaration for a WordScrambler class. The constructor for theWordScrambler class takes an even-length array of String objects and initializes the instance variable scrambledWords.

```
public class WordScrambler
{
  private String[] scrambledWords;

  /** @param wordArr an array of String objects
   *          Precondition: wordArr.length is even
   */
  public WordScrambler(String[] wordArr)
  {
    scrambledWords = mixedWords(wordArr);
  }


  /** @param word1 a String of characters
   *  @param word2 a String of characters
   *  @return a String that contains the first half of word1 and the second half of word2
   */
  private String recombine(String word1, String word2)
  {  /* to be implemented in part (a) */  }


  /** @param words an array of String objects
   *          Precondition: words.length is even
   *  @return an array of String objects created by recombining pairs of strings in array words
   *  Postcondition: the length of the returned array is words.length
   */
  private String[] mixedWords(String[] words)
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

(a) Write the WordScrambler method recombine. This method returns a String created from its two String

parameters as follows.

- take the first half of word1
- take the second half of word2
- concatenate the two halves and return the new string.

For example, the following table shows some results of calling recombine. Note that if a word has an odd number of letters, the second half of the word contains the extra letter.

word1	word2	recombine(word1, word2)
"apple"	"pear"	"apar"
"pear"	"apple"	"peple"

Complete method recombine below.

```
/** @param word1 a String of characters
 *  @param word2 a String of characters
 *  @return a String that contains the first half of word1 and the second half of word2
 */
private String recombine(String word1, String word2)
```

(b) Write the WordScrambler method mixedWords. This method creates and returns a new array of String objects as follows.

It takes the first pair of strings in words and combines them to produce a pair of strings to be included in the array returned by the method. If this pair of strings consists of w1 and w2, the method should include the result of calling recombine with w1 and w2 as arguments and should also include the result of calling recombine with w2 and w1 as arguments. The next two strings, if they exist, would form the next pair to be processed by this method. The method should continue until all the strings in words have been processed in this way and the new array has been filled. For example, if the array words contains the following elements:

{"apple", "pear", "this", "cat"}

then the call mixedWords(words) should return the following array.

{"apar", "peple", "that", "cis"}

In writing mixedWords, you may call recombine. Assume that recombine works as specified, regardless of what

you wrote in part (a).

Complete method mixedWords below.

```
/** @param words an array of String objects
 *         Precondition: words.length is even
 *  @return an array of String objects created by recombining pairs of strings in array words
 *  Postcondition: the length of the returned array is words.length
 */
private String[] mixedWords(String[] words)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

33. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

An array of positive integer values has the mountain property if the elements are ordered such that successive values increase until a maximum value (the peak of the mountain) is reached and then the successive values decrease. The Mountain class declaration shown below contains methods that can be used to determine if an array has the mountain property. You will implement two methods in the Mountain class.

```
public class Mountain
{
  /** @param array an array of positive integer values
   *  @param stop the last index to check
   *          Precondition: 0 ≤ stop < array.length
   *  @return true if for each j such that 0 ≤ j < stop, array[j] < array[j + 1] ;
   *          false otherwise
   */
  public static boolean isIncreasing(int[] array, int stop)
  {  /*  implementation not shown  */  }

  /** @param array an array of positive integer values
   *  @param start the first index to check
   *         Precondition: 0 ≤ start < array.length - 1
   *  @return true if for each j such that start ≤ j < array.length - 1,
   *               array[j] > array[j + 1];
   *          false otherwise
   */
  public static boolean isDecreasing(int[] array, int start)
  {  /*  implementation not shown  */  }

  /** @param array an array of positive integer values
   *          Precondition: array.length > 0
   *  @return the index of the first peak (local maximum) in the array, if it exists;
   *          -1 otherwise
   */
  public static int getPeakIndex(int[] array)
  {  /*  to be implemented in part (a)  */  }

  /** @param array an array of positive integer values
   *          Precondition: array.length > 0
   *  @return true if array contains values ordered as a mountain;
   *          false otherwise
   */
  public static boolean isMountain(int[] array)
  {  /*  to be implemented in part (b)  */  }

  // There may be instance variables, constructors, and methods that are not shown.

}
```

(a) Write the Mountain method getPeakIndex. Method getPeakIndex returns the index of the first peak found in the parameter array, if one exists. A peak is defined as an element whose value is greater than the value of the

element immediately before it and is also greater than the value of the element immediately after it. Method getPeakIndex starts at the beginning of the array and returns the index of the first peak that is found or -1 if no peak is found.
For example, the following table illustrates the results of several calls to getPeakIndex.

arr	getPeakIndex(arr)
{11, 22, 33, 22, 11}	2
{11, 22, 11, 22, 11}	1
{11, 22, 33, 55, 77}	-1
{99, 33, 55, 77, 120}	-1
{99, 33, 55, 77, 55}	3
{33, 22, 11}	-1

Complete method getPeakIndex below.

```
/** @param array an array of positive integer values
 *          Precondition: array.length > 0
 *  @return the index of the first peak (local maximum) in the array, if it exists;
 *          -1 otherwise
 */
public static int getPeakIndex(int[] array)
```

(b) Write the Mountain method isMountain. Method isMountain returns true if the values in the parameter array are ordered as a mountain; otherwise, it returns false. The values in array
are ordered as a mountain if all three of the following conditions hold.

- There must be a peak.
- The array elements with an index smaller than the peak's index must appear in increasing order.
- The array elements with an index larger than the peak's index must appear in decreasing order.

For example, the following table illustrates the results of several calls to isMountain.

Unit2_objectsQuestionsPartBJLC9_18_2023_

arr	isMountain(arr)
{1, 2, 3, 2, 1}	true
{1, 2, 1, 2, 1}	false
{1, 2, 3, 1, 5}	false
{1, 4, 2, 1, 0}	true
{9, 3, 5, 7, 5}	false
{3, 2, 1}	false

In writing isMountain, assume that getPeakIndex works as specified, regardless of what you wrote in part (a). Complete method isMountain below.

```
/** @param array an array of positive integer values
 *         Precondition: array.length > 0
 *  @return true if array contains values ordered as a mountain;
 *          false otherwise
 */
public static boolean isMountain(int[] array)
```

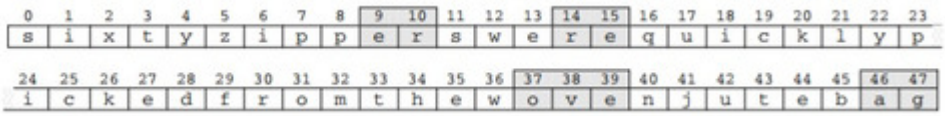
Unit2_objectsQuestionsPartBJLC9_18_2023_

34. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

Consider a method of encoding and decoding words that is based on a master string. This master string will contain all the letters of the alphabet, some possibly more than once. An example of a master string is "sixtyzipperswerequicklypickedfromthewovenjutebag". This string and its indexes are shown below.



An encoded string is defined by a list of string parts. A string part is defined by its starting index in the master string and its length. For example, the string "overeager" is encoded as the list of string parts [(37, 3), (14, 2), (46, 2), (9, 2)] denoting the substrings "ove", "re", "ag", and "er".

String parts will be represented by the StringPart class shown below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class StringPart
{
  /** @param start  the starting position of the substring in a master string
   *  @param length  the length of the substring in a master string
   */
  public StringPart(int start, int length)
  {  /* implementation not shown */  }

  /** @return  the starting position of the substring in a master string
   */
  public int getStart()
  {  /* implementation not shown */  }

  /** @return  the length of the substring in a master string
   */
  public int getLength()
  {  /* implementation not shown */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

The class StringCoder provides methods to encode and decode words using a given master string. When encoding, there may be multiple matching string parts of the master string. The helper method findPart is provided to choose a string part within the master string that matches the beginning of a given string.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class StringCoder
{
  private String masterString;

  /** @param master the master string for the StringCoder
   *         Precondition: the master string contains all the letters of the alphabet
   */
  public StringCoder(String master)
  {  masterString = master;  }

  /** @param parts an ArrayList of string parts that are valid in the master string
   *         Precondition: parts.size() > 0
   *  @return the string obtained by concatenating the parts of the master string
   */
  public String decodeString(ArrayList<StringPart> parts)
  {  /* to be implemented in part (a) */  }

  /** @param str the string to encode using the master string
   *         Precondition: all of the characters in str appear in the master string;
   *                       str.length() > 0
   *  @return a string part in the master string that matches the beginning of str.
   *          The returned string part has length at least 1.
   */
  private StringPart findPart(String str)
  {  /* implementation not shown */  }

  /** @param word the string to be encoded
   *         Precondition: all of the characters in word appear in the master string;
   *                       word.length() > 0
   *  @return an ArrayList of string parts of the master string that can be combined
   *          to create word
   */
  public ArrayList<StringPart> encodeString(String word)
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

a. Write the StringCoder method decodeString. This method retrieves the substrings in the master string represented by each of the StringPart objects in parts, concatenates them in the order in which they appear in parts, and returns the result.

   Complete method decodeString below.

```
/** @param parts an ArrayList of string parts that are valid in the master string
 *         Precondition: parts.size() > 0
 *  @return the string obtained by concatenating the parts of the master string
 */
public String decodeString(ArrayList<StringPart> parts)
```

b. Write the StringCoder method encodeString. A string is encoded by determining the substrings in the master string that can be combined to generate the given string. The encoding starts with a string part that matches the beginning of the word, followed by a string part that matches the beginning of the rest of the word, and so on. The string parts are returned in an array list in the order in which they appear in word.

   The helper method findPart must be used to choose matching string parts in the master string.

   Complete method encodeString below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
/** @param word the string to be encoded
 *         Precondition: all of the characters in word appear in the master string;
 *                       word.length() > 0
 *  @return an ArrayList of string parts of the master string that can be combined
 *          to create word
 */
public ArrayList<StringPart> encodeString(String word)
```

35. **Directions**: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

An organization raises money by selling boxes of cookies. A cookie order specifies the variety of cookie and the number of boxes ordered. The declaration of the CookieOrder class is shown below.

```
public class CookieOrder
{
  /** Constructs a new CookieOrder object. */
  public CookieOrder(String variety, int numBoxes)
  {  /* implementation not shown */  }

  /** @return the variety of cookie being ordered
   */
  public String getVariety()
  {  /* implementation not shown */  }

  /** @return the number of boxes being ordered
   */
  public int getNumBoxes()
  {  /* implementation not shown */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

The MasterOrder class maintains a list of the cookies to be purchased. The declaration of the MasterOrder class is shown below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class MasterOrder
{
  /** The list of all cookie orders */
  private List<CookieOrder> orders;

  /** Constructs a new MasterOrder object. */
  public MasterOrder()
  {  orders = new ArrayList<CookieOrder>();  }

  /** Adds theOrder to the master order.
   *  @param theOrder the cookie order to add to the master order
   */
  public void addOrder(CookieOrder theOrder)
  {  orders.add(theOrder);  }

  /** @return the sum of the number of boxes of all of the cookie orders
   */
  public int getTotalBoxes()
  {  /* to be implemented in part (a) */  }

  /** Removes all cookie orders from the master order that have the same variety of
   *  cookie as cookieVar and returns the total number of boxes that were removed.
   *  @param cookieVar the variety of cookies to remove from the master order
   *  @return the total number of boxes of cookieVar in the cookie orders removed
   */
  public int removeVariety(String cookieVar)
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

a. The getTotalBoxes method computes and returns the sum of the number of boxes of all cookie orders. If there are no cookie orders in the master order, the method returns 0.

   Complete method getTotalBoxes below.

```
/** @return the sum of the number of boxes of all of the cookie orders
 */
public int getTotalBoxes()
```

b. The removeVariety method updates the master order by removing all of the cookie orders in which the variety of cookie matches the parameter cookieVar. The master order may contain zero or more cookie orders with the same variety as cookieVar. The method returns the total number of boxes removed from the master order.

   For example, consider the following code segment.

```
MasterOrder goodies = new MasterOrder();
goodies.addOrder(new CookieOrder("Chocolate Chip", 1));
goodies.addOrder(new CookieOrder("Shortbread", 5));
goodies.addOrder(new CookieOrder("Macaroon", 2));
goodies.addOrder(new CookieOrder("Chocolate Chip", 3));
```

   After the code segment has executed, the contents of the master order are as shown in the following table.

"Chocolate Chip" 1	"Shortbread" 5	"Macaroon" 2	"Chocolate Chip" 3

The method call goodies.removeVariety("Chocolate Chip") returns 4 because there were two Chocolate Chip cookie orders totaling 4 boxes. The master order is modified as shown below.

"Shortbread" 5	"Macaroon" 2

The method call goodies.removeVariety("Brownie") returns 0 and does not change the master order.

Complete method removeVariety below.

```
/** Removes all cookie orders from the master order that have the same variety of
 *  cookie as cookieVar and returns the total number of boxes that were removed.
 *  @param cookieVar the variety of cookies to remove from the master order
 *  @return the total number of boxes of cookieVar in the cookie orders removed
 */
public int removeVariety(String cookieVar)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

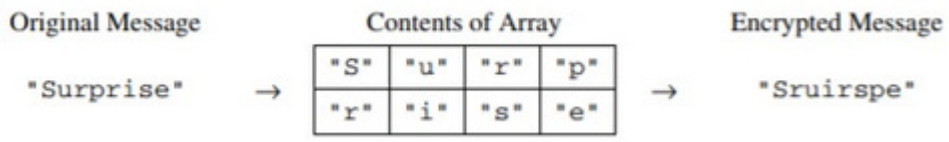
36. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit. Digital sounds can be represented as an array of integer values. For this question, you will write two unrelated methods of the Sound class.

In this question you will write two methods for a class RouteCipher that encrypts (puts into a coded form) a message by changing the order of the characters in the message. The route cipher fills a two-dimensional array with single-character substrings of the original message in row-major order, encrypting the message by retrieving the single-character substrings in column-major order.

For example, the word "Surprise" can be encrypted using a 2-row, 4-column array as follows.



An incomplete implementation of the RouteCipher class is shown below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class RouteCipher
{
  /** A two-dimensional array of single-character strings, instantiated in the constructor */
  private String[][] letterBlock;

  /** The number of rows of letterBlock, set by the constructor */
  private int numRows;

  /** The number of columns of letterBlock, set by the constructor */
  private int numCols;

  /** Places a string into letterBlock in row-major order.
   *  @param str the string to be processed
   *  Postcondition:
   *    if str.length() < numRows * numCols, "A" is placed in each unfilled cell
   *    if str.length() > numRows * numCols, trailing characters are ignored
   */
  private void fillBlock(String str)
  {  /* to be implemented in part (a) */  }

  /** Extracts encrypted string from letterBlock in column-major order.
   *  Precondition: letterBlock has been filled
   *  @return the encrypted string from letterBlock
   */
  private String encryptBlock()
  {  /* implementation not shown */  }

  /** Encrypts a message.
   *  @param message the string to be encrypted
   *  @return the encrypted message;
   *          if message is the empty string, returns the empty string
   */
  public String encryptMessage(String message)
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

a. Write the method fillBlock that fills the two-dimensional array letterBlock with one-character strings from the string passed as parameter str.

   The array must be filled in row-major order—the first row is filled from left to right, then the second row is filled from left to right, and so on, until all rows are filled.

   If the length of the parameter str is smaller than the number of elements of the array, the string "A" is placed in each of the unfilled cells. If the length of str is larger than the number of elements in the array, the trailing characters are ignored.

   For example, if letterBlock has 3 rows and 5 columns and str is the string "Meet at noon", the resulting contents of letterBlock would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"n"	"o"
"o"	"n"	"A"	"A"	"A"

   If letterBlock has 3 rows and 5 columns and str is the string "Meet at midnight", the resulting contents of letterBlock would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"m"	"i"
"d"	"n"	"i"	"g"	"h"

The following expression may be used to obtain a single-character string at position k of the string str.

str.substring(k, k + 1)

Complete method fillBlock below.

```
/** Places a string into letterBlock in row-major order.
 *  @param str the string to be processed
 *  Postcondition:
 *    if str.length() < numRows * numCols, "A" is placed in each unfilled cell
 *    if str.length() > numRows * numCols, trailing characters are ignored
 */
private void fillBlock(String str)
```

b. Write the method encryptMessage that encrypts its string parameter message. The method builds an encrypted version of message by repeatedly calling fillBlock with consecutive, nonoverlapping substrings of message and concatenating the results returned by a call to encryptBlock after each call to fillBlock. When all of message has been processed, the concatenated string is returned. Note that if message is the empty string, encryptMessage returns an empty string.

The following example shows the process carried out if letterBlock has 2 rows and 3 columns and encryptMessage("Meet at midnight") is executed.

Substring	letterBlock after Call to fillBlock	Value Returned by encryptBlock	Concatenated String
"Meet a"	"M" "e" "e" "t" " " "a"	"Mte ea"	"Mte ea"
"t midn"	"t" " " "m" "i" "d" "n"	"ti dmn"	"Mte eati dmn"
"ight"	"i" "g" "h" "t" "A" "A"	"itgAhA"	"Mte eati dmnitgAhA"

In this example, the method returns the string "Mte eati dmnitgAhA".

Assume that fillBlock and encryptBlock methods work as specified. Solutions that reimplement the functionality of one or both of these methods will not receive full credit.

Complete method encryptMessage below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
/** Encrypts a message.
 *  @param message the string to be encrypted
 *  @return the encrypted message;
 *          if message is the empty string, returns the empty string
 */
public String encryptMessage(String message)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

37. **Directions:** SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

A multiplayer game called Token Pass has the following rules. Each player begins with a random number of tokens (at least 1, but no more than 10) that are placed on a linear game board. There is one position on the game board for each player. After the game board has been filled, a player is randomly chosen to begin the game. Each position on the board is numbered, starting with 0.

The following rules apply for a player’s turn.

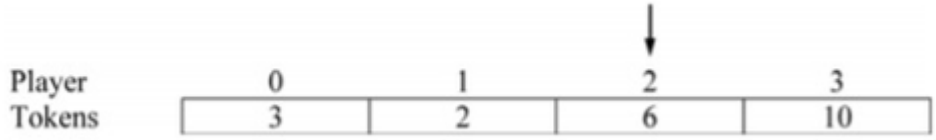
- The tokens are collected and removed from the game board at that player's position.
- The collected tokens are distributed one at a time, to each player, beginning with the next player in order of increasing position.
- If there are still tokens to distribute after the player at the highest position gets a token, the next token will be distributed to the player at position 0.
- The distribution of tokens continues until there are no more tokens to distribute.

The Token Pass game board is represented by an array of integers. The indexes of the array represent the player positions on the game board, and the corresponding values in the array represent the number of tokens that each player has. The following example illustrates one player's turn.

Example

The following represents a game with 4 players. The player at position 2 was chosen to go first.

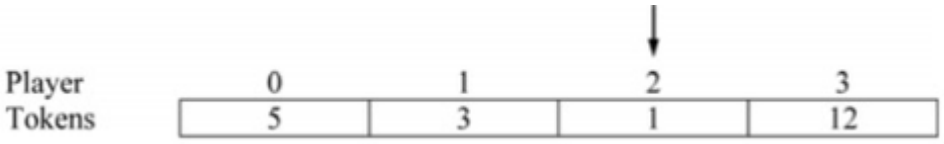
Unit2_objectsQuestionsPartBJLC9_18_2023_



The tokens at position 2 are collected and distributed as follows.

- 1st token - to position 3 (The highest position is reached, so the next token goes to position 0.)
- 2nd token - to position 0
- 3rd token - to position 1
- 4th token - to position 2
- 5th token - to position 3 (The highest position is reached, so the next token goes to position 0.)
- 6th token - to position 0

After player 2's turn, the values in the array will be as follows.



The Token Pass game is represented by the TokenPass class.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class TokenPass
{
  private int[] board;
  private int currentPlayer;

  /** Creates the board array to be of size playerCount and fills it with
   *  random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
   *  random integer value in the range between 0 and playerCount-1, inclusive.
   *  @param playerCount the number of players
   */
  public TokenPass(int playerCount)
  {  /* to be implemented in part (a) */  }

  /** Distributes the tokens from the current player's position one at a time to each player in
   *  the game. Distribution begins with the next position and continues until all the tokens
   *  have been distributed. If there are still tokens to distribute when the player at the
   *  highest position is reached, the next token will be distributed to the player at position 0.
   *  Precondition: the current player has at least one token.
   *  Postcondition: the current player has not changed.
   */
  public void distributeCurrentPlayerTokens()
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

a. Write the constructor for the TokenPass class. The parameter playerCount represents the number of players in the game. The constructor should create the board array to contain playerCount elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable currentPlayer to a random number between 0 and playerCount-1, inclusive.

   Complete the TokenPass constructor below.

```
/** Creates the board array to be of size playerCount and fills it with
 *  random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
 *  random integer value in the range between 0 and playerCount-1, inclusive.
 *  @param playerCount the number of players
 */
public TokenPass(int playerCount)
```

b. Write the distributeCurrentPlayerTokens method.

   The tokens are collected and removed from the game board at the current player's position. These tokens are distributed, one at a time, to each player, beginning with the next higher position, until there are no more tokens to distribute.

Class information repeated from the beginning of the question

```
public class TokenPass

private int[] board
private int currentPlayer
public TokenPass(int playerCount)
public void distributeCurrentPlayerTokens()
```

Complete method distributeCurrentPlayerTokens below.

```
/** Distributes the tokens from the current player's position one at a time to each player in
 *  the game. Distribution begins with the next position and continues until all the tokens
 *  have been distributed. If there are still tokens to distribute when the player at the
 *  highest position is reached, the next token will be distributed to the player at position 0.
 *  Precondition: the current player has at least one token.
 *  Postcondition: the current player has not changed.
 */
public void distributeCurrentPlayerTokens()
```

38. Consider the following code segment.

```
double d1 = 10.0;
Double d2 = 20.0;
Double d3 = new Double(30.0);
double d4 = new Double(40.0);

System.out.println(d1 + d2 + d3.doubleValue() + d4);
```

What, if anything, is printed when the code segment is executed?

(A) `100.0`

(B) `10.050.040.0`

(C) `10.020.070.0`

(D) `10.020.030.040.0`

(E) There is no output due to a compilation error.

Consider the following method.

```
public static String[] strArrMethod(String[] arr)
{
     String[] result = new String[arr.length];
     for (int j = 0; j < arr.length; j++)
     {
         String sm = arr[j];
         for (int k = j + 1; k < arr.length; k++)
         {
            if (arr[k].length() < sm.length())
            {
        sm = arr[k]; // Line 12
            }
         }
         result[j] = sm;
     }
     return result;
}
```

39. Consider the following code segment.

```
String[] testOne = {"first", "day", "of\", "spring"};
String[] resultOne = strArrMethod(testOne);
```

What are the contents of `resultOne` when the code segment has been executed?

(A) `{"day", "first", "of\", "spring"}`

(B) `{"of\", "day", "first", "spring"}`

(C) `{"of\", "day", "of\", "spring"}`

(D) `{"of\", "of\", "of\", "spring"}`

(E) `{"spring", "first", "day", "of\"}`

40. Consider the following code segment.

```
String[] testTwo = {"last", "day", "of\", "the", "school", "year"};
String[] resultTwo = strArrMethod(testTwo);
```

How many times is the line labeled `// Line 12` in the `strArrMethod` executed as a result of executing the code segment?

(A) 4 times

(B) 5 times

(C) 6 times

(D) 15 times

(E) 30 times

Unit2_objectsQuestionsPartBJLC9_18_2023_

41. Consider the following method.

```
public static int what(String str, String check)
{
     int num = -1;
     int len = check.length();
     for (int k = 0; k + len <= str.length(); k++)
     {
         String a = str.substring(k, k + len);
         if (a.equals(check))
         {
            num = k;
         }
     }
     return num;
}
```

Assume that `check` occurs at least once in `str`. Which of the following best describes the value returned by the `what` method?

(A) The number of times the string `check` occurs in `str`

(B) The index of the first occurrence of `check` inside `str`

(C) The index of the last occurrence of `check` inside `str`

(D) The number of substrings in `str` with the same length as `check`

(E) The number of substrings in `str` that do not match `check`

Unit2_objectsQuestionsPartBJLC9_18_2023_

42. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves the creation and use of a spinner to generate random numbers in a game. A `GameSpinner` object represents a spinner with a given number of sectors, all equal in size. The `GameSpinner` class supports the following behaviors.

- Creating a new spinner with a specified number of sectors
- Spinning a spinner and reporting the result
- Reporting the length of the *current run*, the number of consecutive spins that are the same as the most recent spin

The following table contains a sample code execution sequence and the corresponding results.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Statements	Value Returned (blank if no value returned)	Comment
`GameSpinner g = new GameSpinner(4);`		Creates a new spinner with four sectors
`g.currentRun();`	`0`	Returns the length of the current run. The length of the current run is initially `0` because no spins have occurred.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because there has been one spin of `3` so far.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`2`	The length of the current run is `2` because there have been two `3`s in a row.
`g.spin();`	`4`	Returns a random integer between `1` and `4`, inclusive. In this case, `4` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `4` is different from the value of the spin in the previous run of two `3`s.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `3` is different from the value of the spin in the previous run of one `4`.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.currentRun();`	`3`	The length of the current run is `3` because there have been three consecutive `1`s since the previous run of one `3`.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Write the complete `GameSpinner` class. Your implementation must meet all specifications and conform to the example.

Unit2_objectsQuestionsPartBJLC9_18_2023_

43. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves generating a `String` that will be used as an identifier. You will write the `generateID` method of the following `Identifier` class.

```
public class Identifier
{
 /** Encodes a string as an integer and returns the encoded int value */
 public static int encodeToNumber(String str)
 {  /* implementation not shown */  }

 /** Returns an identifier string based on an input string, as described
 in part (a)
  *   Precondition: input is not null.
  */
 public static String generateID(String input)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `generateID` method, which is used to transform an input string into a string that can be used as an identifier. The method creates and returns the identifier string based on the following rules.

- If the length of the input string is not divisible by 4, the method returns the string `"error"`.
- Every non-overlapping 4-character grouping of the input string is encoded as an integer using the helper method `encodeToNumber`. The sum of all the encoded values is calculated.
- If the sum is greater than 100, the method returns the original input string with `"3"` appended.
- Otherwise, the method returns the original input string with `"X"` appended.

The following table shows some examples of calls to the `generateID` method. Assume that all calls occur in the `Identifier` class.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Call to `generateID`	**Possible Values Returned by** `encodeToNumber`	`generateID` **Return Value**
`generateID("treebook")`	`encodeToNumber("tree")` returns `17` `encodeToNumber("book")` returns `2`	`"treebookX"`

`generateID("doordesklion")`	`encodeToNumber("door")` returns `56`
	`encodeToNumber("desk")` returns `35`
	`encodeToNumber("lion")` returns `86`
		`"doordesklion3"`

`generateID("today")`		`"error"` (because the length of `"today"` is not divisible by 4)

Complete method `generateID`. You must use `encodeToNumber` appropriately to receive full credit.

```
/** Returns an identifier string based on an input string, as described
in part (a)
 *  Precondition: input is not null.
 */
public static String generateID(String input)
```

(b) A programmer wants to modify the `Identifier` class to keep track of how many times a call to `generateID` returns `"error"`. The programmer would like to implement this change without making any changes to the signatures of `generateID` or `encodeToNumber` or overloading either method.

Write a description of how you would change the `Identifier` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

The `Gizmo` class represents gadgets that people purchase. Some `Gizmo` objects are electronic and others are not. A partial definition of the `Gizmo` class is shown below.

```
public class Gizmo
{
     /** Returns the name of the manufacturer of this Gizmo. */
     public String getMaker()
     {
         /* implementation not shown */
     }

     /** Returns true if this Gizmo is electronic, and false otherwise. */
     public boolean isElectronic()
     {
         /* implementation not shown */
     }

     /** Returns true if this Gizmo is equivalent to the Gizmo object
   represented by the
      * parameter, and false otherwise.
      */
     public boolean equals(Object other)
     {
         /* implementation not shown */
     }

     // There may be instance variables, constructors, and methods not shown.
}
```

The `OnlinePurchaseManager` class manages a sequence of `Gizmo` objects that an individual has purchased from an online vendor. You will write two methods of the `OnlinePurchaseManager` class. A partial definition of the `OnlinePurchaseManager` class is shown below.

```
public class OnlinePurchaseManager
{
     /** An ArrayList of purchased Gizmo objects, instantiated in the
   constructor. */
     private ArrayList<Gizmo> purchases;

     /** Returns the number of purchased Gizmo objects that are electronic and
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
  are
      * manufactured by maker, as described in part (a).
      */
     public int countElectronicsByMaker(String maker)
     {
         /* to be implemented in part (a) */
     }

     /** Returns true if any pair of adjacent purchased Gizmo objects are
  equivalent, and
      * false otherwise, as described in part (b).
      */
     public boolean hasAdjacentEqualPair()
     {
         /* to be implemented in part (b) */
     }

     // There may be instance variables, constructors, and methods not shown.
}
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

44. **(a)** Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in `purchases`	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 * whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
equivalent, and
 * false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

45. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves analyzing simulated student grades that are obtained using the `getGrade` method in the following `GradeAnalysis` class. You will write one method in the class.

```
public class GradeAnalysis
{
 /** Returns a simulated student grade greater than or equal to 0.0,
   *   or -1.0 if no more grades are available
  */
 public static double getGrade()
 {  /* implementation not shown */  }

 /** Analyzes grades obtained from the getGrade method and returns the
 proportion
   *   of grades that are 90.0 and above, as described in part (a)
  */
 public static double gradeSimulation()
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Method `gradeSimulation` obtains grades using the `getGrade` method until the `getGrade` method returns the value `-1.0`. The `gradeSimulation` method returns the proportion of grades that are at least `90.0`.

For example, if `getGrade` returns the values `60.0`, `80.0`, `110.0`, `100.0`, and `-1.0`, then the `gradeSimulation` method should return `0.5` because of the four grade values returned by `getGrade`, two are at least `90.0`.

Assume that `getGrade` always returns at least one grade before returning `-1.0`. You must use `getGrade` appropriately to receive full credit.

Complete method `gradeSimulation`.

```
/** Analyzes grades obtained from the getGrade method and returns the
proportion
  *   of grades that are 90.0 and above, as described in part (a)
 */
public static double gradeSimulation()
```

(b) A programmer wants to modify the `GradeAnalysis` class to maintain a maximum grade in a variable with a value that cannot be changed once it is initialized. Any grades that exceed the maximum grade are to be ignored in the proportion calculated by the `gradeSimulation` method. The programmer would like to implement this change without making any changes to the signature of the `gradeSimulation` method or overloading `gradeSimulation`.

Write a description of how the `GradeAnalysis` class would need to change in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

46. Consider the method `getHours`, which is intended to calculate the number of hours that a vehicle takes to travel between two *mile markers* on a highway if the vehicle travels at a constant speed of 60 miles per hour. A mile marker is a sign showing the number of miles along a road between some fixed location (for example, the beginning of a highway) and the current location.

The following table shows two examples of the intended behavior of `getHours`, based on the `int` parameters `marker1` and `marker2`.

marker1	marker2	**Return Value**
100	220	2.0
100	70	0.5

Consider the following implementation of `getHours`.

```
public static double getHours(int marker1, int marker2)
{
     /* missing statement */
     return hours;
}
```

Which of the following statements can replace `/* missing statement */` so `getHours` works as intended?

Unit2_objectsQuestionsPartBJLC9_18_2023_

(A) `double hours = (Math.abs(marker1) - Math.abs(marker2)) / 60.0;`

(B) `double hours = Math.abs(marker1 - marker2 / 60.0);`

(C) `double hours = Math.abs(marker1 - marker2) / 60.0;`

(D) `double hours = Math.abs((marker1 - marker2) / 60);`

(E) `double hours = (double) (Math.abs(marker1 - marker2) / 60);`

Unit2_objectsQuestionsPartBJLC9_18_2023_

47. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves validating a `String` that is being used as an identifier. You will write the `isValid` method of the following `Authenticate` class.

```
public class Authenticate
{
 /** Returns true if the input string has some property and returns
 false
  *  otherwise
  */
 public static boolean hasProperty(String str)
 {  /* implementation not shown */  }

 /** Returns true if the input string passes validation and returns
 false otherwise, as
  *   described in part (a)
  *  Precondition: input is not null.
  */
 public static boolean isValid(String input)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `isValid` method, which validates an input string and returns a `boolean` value based on the following conditions. Every non-overlapping 3-character grouping of the input string is verified individually using the helper method `hasProperty`.

- If the length of the input string is not divisible by 3, the `isValid` method returns `false`.
- If the input string contains at least two 3-character groupings for which `hasProperty` returns `true`, the `isValid` method returns `true`.
- If there are fewer than two 3-character groupings for which `hasProperty` returns true, the `isValid` method returns `false`.

The following table shows some examples of the intended behavior of `isValid`. Assume that all calls are made from within the `Authenticate` class.

Call to `isValid`	**Possible Values Returned by** `hasProperty`	`isValid` **Return** **Value**

`isValid("butterfly")`	`hasProperty("but")` returns `true`
	
	`hasProperty("ter")` returns `false`
	
	`hasProperty("fly")` returns `false`
		`false`

`isValid("turtle")`	`hasProperty("tur")` returns `true`
	
	`hasProperty("tle")` returns `true`
		`true`

`isValid("bear")`		`false` (because the length of `"bear"` is not divisible by 3)

You must use `hasProperty` appropriately to receive full credit.

Complete method `isValid`.

```
/** Returns true if the input string passes validation and returns false
otherwise, as
 *   described in part (a)
 *  Precondition: input is not null.
```

```
   */
  public static boolean isValid(String input)
```

(b) A programmer wants to modify the `Authenticate` class so that, in the `isValid` method, the rules for character groupings of 3 can vary between method calls. For example, in one call to `isValid`, the rules might check for divisibility by `4` with 4-character groupings, and in another call to `isValid`, the rule might check for divisibility by `5` with 5-character groupings. The programmer would like to implement this change without making any changes to the signature of the `isValid` method or overloading `isValid`.

Write a description of how you would change the `Authenticate` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

48. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves manipulating strings. The following class, `StringPatterns`, contains methods that look for letters in patterns. You will write one method in this class.

```
public class StringPatterns
{
 /** Returns the String that results when letter and pattern are
 compared,
  *   as described in part (a)
  *   Precondition: letter consists of one uppercase letter.
  *                 pattern has at least 2 letters and all letters are
 uppercase and
  *             unique.
  */
 public static String letterAndPattern(String letter,
      String pattern)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `StringPatterns` method `letterAndPattern`, which returns a string based on the parameter `letter` and the parameter `pattern`. The parameter `letter` contains a single uppercase letter and the parameter `pattern` has at least 2 letters. All letters in the pattern are uppercase and unique, such as `"ABC"`.

The method returns a string value based on the following rules.

If the letter is contained in the pattern, the letter is returned.
If the letter does not appear in the pattern, a string containing the letters of the pattern, in reverse order, is returned.

The following table shows the results of calls to `letterAndPattern`. Assume the calls are made within the `StringPatterns` class.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Method Call	Return Value
`letterAndPattern("Q", "QUICK")`	`"Q"`
`letterAndPattern("C", "QUICK")`	`"C"`
`letterAndPattern("Z", "QUICK")`	`"KCIUQ"`

Complete method `letterAndPattern`.

```
/** Returns the String that results when letter and pattern are compared,
  *  as described in part (a)
  *  Precondition: letter consists of one uppercase letter.
  *                pattern has at least 2 letters and all letters are
uppercase and unique.
 */
public static String letterAndPattern(String letter, String pattern)
```

(b) The programmer would like to modify the `StringPatterns` class to limit the length of the string that can be passed as a pattern to `letterAndPattern`. This limit can vary between calls to `letterAndPattern`. Any call to `letterAndPattern` with a longer pattern than is allowed would return the empty string. The programmer would like to implement this change without making any changes to the signature of the `letterAndPattern` method or overloading `letterAndPattern`.

Write a description of how you would change the `StringPatterns` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

49. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

The `LightBoard` class models a two-dimensional display of lights, where each light is either on or off, as represented by a Boolean value. You will implement a constructor to initialize the display and a method to evaluate a light.

```
public class LightBoard
{
     /** The lights on the board, where true represents on and false
   represents off.
      */
      private boolean[][] lights;

     /** Constructs a LightBoard object having numRows rows and numCols
   columns.
      *  Precondition: numRows > 0, numCols > 0
      *  Postcondition: each light has a 40% probability of being set to
    on.
      */
     public LightBoard(int numRows, int numCols)
     {  /* to be implemented in part (a) */  }

     /** Evaluates a light in row index row and column index col and
   returns a status
      *  as described in part (b).
      *  Precondition: row and col are valid indexes in lights.
      */
     public boolean evaluateLight(int row, int col)
     {  /* to be implemented in part (b) */  }

     // There may be additional instance variables, constructors, and
   methods not shown.

}
```

(a)

Write the constructor for the `LightBoard` class, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Complete the `LightBoard` constructor below.

```
/** Constructs a LightBoard object having numRows rows and numCols
columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols)
```

(b)

Write the method `evaluateLight`, which computes and returns the status of a light at a given row and column based on the following rules.

1. If the light is on, return `false` if the number of lights in its column that are on is even, including the current light.
2. If the light is off, return `true` if the number of lights in its column that are on is divisible by three.
3. Otherwise, return the light's current status.

For example, suppose that `LightBoard sim = new LightBoard(7, 5)` creates a light board with the initial state shown below, where `true` represents a light that is on and `false` represents a light that is off. Lights that are off are shaded.

`lights`

	0	1	2	3	4
0	true	true	false	true	true
1	true	false	false	true	false
2	true	false	false	true	true
3	true	false	false	false	true
4	true	false	false	false	true
5	true	true	false	true	true
6	false	false	false	false	false

Sample calls to `evaluateLight` are shown below.

Call to `evaluateLight`	Value Returned	Explanation
`sim.evaluateLight(0, 3);`	`false`	The light is on, and the number of lights that are on in its column is even.
`sim.evaluateLight(6, 0);`	`true`	The light is off, and the number of lights that are on in its column is divisible by 3.
`sim.evaluateLight(4, 1);`	`false`	Returns the light's current status.
`sim.evaluateLight(5, 4);`	`true`	Returns the light's current status.

Class information for this question

```
public class LightBoard
private boolean[][] lights
public LightBoard(int numRows, int numCols)
public boolean evaluateLight(int row, int col)
```

Complete the `evaluateLight` method below.

```
/** Evaluates a light in row index row and column index col and returns a
status
 *  as described in part (b).
 *  Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
```

50. This question involves two classes that are used to process log messages. A list of sample log messages is given below.

```
CLIENT3:security alert - repeated login failures
Webserver:disk offline
SERVER1:file not found
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
Webserver:error on /dev/disk
```

Log messages have the format *machineId:description*, where *machineId* identifies the computer and *description* describes the event being logged. Exactly one colon (":") appears in a log message. There are no blanks either immediately before or immediately after the colon.

The following LogMessage class is used to represent a log message.

```
public class LogMessage
{
    private String machineId;
    private String description;

    /** Precondition: message is a valid log message. */
    public LogMessage(String message)
    {  /* to be implemented in part (a) */  }

    /** Returns true if the description in this log message properly contains keyword;
     *          false otherwise.
     */
    public boolean containsWord(String keyword)
    {  /* to be implemented in part (b) */  }

    public String getMachineId()
    {  return machineId;  }

    public String getDescription()
    {  return description;  }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

a. Write the constructor for the LogMessage class. It must initialize the private data of the object so that getMachineId returns the *machineId* part of the message and getDescription returns the *description* part of the message.

   Complete the LogMessage constructor below.

```
/** Precondition: message is a valid log message. */
public LogMessage(String message)
```

b. Write the LogMessage method containsWord, which returns true if the description in the log message *properly contains* a given keyword and returns false otherwise.

A description *properly contains* a keyword if all three of the following conditions are true.

- the keyword is a substring of the description;
- the keyword is either at the beginning of the description or it is immediately preceded by a space;
- the keyword is either at the end of the description or it is immediately followed by a space.

The following tables show several examples. The descriptions in the left table properly contain the keyword "disk". The descriptions in the right table do not properly contain the keyword "disk".

Descriptions that properly contain "disk"
"disk"
"error on disk"
"error on /dev/disk disk"
"error on disk DSK1"

Descriptions that do not properly contain "disk"
"DISK"
"error on disk3"
"error on /dev/disk"
"diskette"

Assume that the LogMessage constructor works as specified, regardless of what you wrote in part (a). Complete method containsWord below.

```
/** Returns true if the description in this log message properly contains keyword;
 *          false otherwise.
 */
public boolean containsWord(String keyword)
```

c. The SystemLog class represents a list of LogMessage objects and provides a method that removes and returns a list of all log messages (if any) that properly contain a given keyword. The messages in the returned list appear in the same order in which they originally appeared in the system log. If no message properly contains the keyword, an empty list is returned. The declaration of the SystemLog class is shown below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
public class SystemLog
{
   /** Contains all the entries in this system log.
    *  Guaranteed not to be null and to contain only non-null entries.
    */
   private List<LogMessage> messageList;

   /** Removes from the system log all entries whose descriptions properly contain keyword,
    *  and returns a list (possibly empty) containing the removed entries.
    *  Postcondition:
    *    - Entries in the returned list properly contain keyword and
    *      are in the order in which they appeared in the system log.
    *    - The remaining entries in the system log do not properly contain keyword and
    *      are in their original order.
    *    - The returned list is empty if no messages properly contain keyword.
    */
   public List<LogMessage> removeMessages(String keyword)
   {  /* to be implemented in part (c) */  }

   // There may be instance variables, constructors, and methods that are not shown.
}
```

Write the SystemLog method removeMessages, which removes from the system log all entries whose descriptions properly contain keyword and returns a list of the removed entries in their original order. For example, assume that theLog is a SystemLog object initially containing six LogMessage objects representing the following list of log messages.

```
CLIENT3:security alert - repeated login failures
Webserver:disk offline
SERVER1:file not found
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
Webserver:error on /dev/disk
```

The call theLog.removeMessages("disk") would return a list containing the LogMessage objects representing the following log messages.

```
Webserver:disk offline
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
```

After the call, theLog would contain the following log messages.

```
CLIENT3:security alert - repeated login failures
SERVER1:file not found
Webserver:error on /dev/disk
```

Assume that the LogMessage class works as specified, regardless of what you wrote in parts (a) and (b). You must use containsWord appropriately to receive full credit.

Complete method removeMessages below.

```
/** Removes from the system log all entries whose descriptions properly contain keyword,
 *  and returns a list (possibly empty) containing the removed entries.
 *  Postcondition:
 *    - Entries in the returned list properly contain keyword and
 *      are in the order in which they appeared in the system log.
 *    - The remaining entries in the system log do not properly contain keyword and
 *      are in their original order.
 *    - The returned list is empty if no messages properly contain keyword.
 */
public List<LogMessage> removeMessages(String keyword)
```

51. Consider the following code segment.

```
String str = "abcdef";

for (int rep = 0; rep < str.length() - 1; rep++)

{

  System.out.print(str.substring(rep, rep + 2));

}
```

What is printed as a result of executing this code segment?

(A) abcdef

(B) aabbccddeeff

(C) abbccddeef

(D) abcbcdcdedef

(E) Nothing is printed because an IndexOutOfBoundsException is thrown.

Unit2_objectsQuestionsPartBJLC9_18_2023_

52. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

1. This question involves managing music lessons at a music school. Lessons are defined by the following `Lesson` class.

```
public class Lesson
{
 /** Returns the type of the lesson */
 public String getType()
 {  /* implementation not shown */  }

 /** Returns the cost of the lesson */
 public double getCost()
 {  /* implementation not shown */  }

 /** Sets the cost of the lesson to newCost */
 public void setCost(double newCost)
 {  /* implementation not shown */  }

 /** Returns true if the lesson signup occurred during the early
 registration period
  *  and returns false otherwise
  */
 public boolean isRegEarly()
 {  /* implementation not shown */  }

 // There may be instance variables, constructors, and methods that are
 not shown.
}
```

Information about a music school is stored in a `MusicSchool` object, which contains a list of the lessons that students have signed up for at the school. You will write two methods of the `MusicSchool` class.

```
public class MusicSchool
```

```
{
 /** A list containing information about lessons signed up for at the
 school
   *  Guaranteed not to be null and to contain only non-null entries
  */
 private ArrayList<Lesson> lessonList;

 /** Updates the cost for all lessons, as described in part (a) */
 public void updateCosts(double piano, double voice, double reg)
 {  /* to be implemented in part (a) */  }

 /** Updates the cost of a randomly selected piano lesson by a given
 discount and returns the
   *  updated cost, as described in part (b)
   *  Precondition: At least one lesson meets the eligibility
 requirements and
   *           the discount is less than the cost of the lesson.
  */
 public double getDiscountedLessonCost(double discount)
 {  /* to be implemented in part (b) */  }

 // There may be instance variables, constructors, and methods that are
 not shown.
}
```

(a) Write the `MusicSchool` method `updateCosts`. Once a year, the school updates the costs of lessons based on the following rules.

The cost of all `"piano"` lessons is increased by the value of the parameter `piano`.
The cost of all `"voice"` lessons is increased by the value of the parameter `voice`.
The cost of all other types of lessons is increased by the value of the parameter `reg`.

For example, assume that `chelseaSchool` has been declared as a `MusicSchool` object and that `lesson1`, `lesson2`, `lesson3`, `lesson4`, `lesson5`, and `lesson6` are properly declared and initialized `Lesson` objects in `lessonList`. The following table shows the cost of lessons before and after the method call `chelseaSchool.updateCosts(2.0, 4.0, 1.0)`.

Unit2_objectsQuestionsPartBJLC9_18_2023_

`Lesson` Object	Lesson Type	Lesson Cost Before the Method Call	Lesson Cost After the Method Call
`lesson1`	`"guitar"`	`20.0`	`21.0`
`lesson2`	`"piano"`	`10.0`	`12.0`
`lesson3`	`"piano"`	`15.0`	`17.0`
`lesson4`	`"voice"`	`35.0`	`39.0`
`lesson5`	`"cello"`	`25.0`	`26.0`
`lesson6`	`"piano"`	`20.0`	`22.0`

Complete method `updateCosts`.

```
/** Updates the cost for all lessons, as described in part (a) */
 public void updateCosts(double piano, double voice, double reg)
```

(b) Write the `MusicSchool` method `getDiscountedLessonCost`. Once a month, the music school offers a discount on a randomly selected, eligible lesson. A lesson is eligible for the discount if the type of lesson is `"piano"` and the lesson was signed up for during the early registration period. Each eligible lesson must have an equal chance of being selected.

The method updates the cost of the selected lesson and returns the updated lesson cost, as described in the following example.

Assume that `chelseaSchool` has been declared as a `MusicSchool` object and contains the following `Lesson` objects.

Lesson **Object**	**Lesson Type**	**Lesson Cost**	isRegEarly() **Return Value**
lesson1	"guitar"	20.0	true
lesson2	"piano"	10.0	false
lesson3	"piano"	15.0	true
lesson4	"voice"	35.0	false
lesson5	"cello"	25.0	false
lesson6	"piano"	20.0	true

There are two eligible lessons, `lesson3` and `lesson6`. Either could be selected by the call `chelseaSchool.getDiscountedLessonCost(4.0)`. If `lesson3` were selected, its cost would be updated to `11.0` and the value `11.0` would be returned. If `lesson6` were selected, its cost would be updated to `16.0` and the value `16.0` would be returned.

Complete method `getDiscountedLessonCost`.

```
/** Updates the cost of a randomly selected piano lesson by a given
discount and returns the
  *  updated cost, as described in part (b)
  *  Precondition: At least one lesson meets the eligibility requirements
and
  *          the discount is less than the cost of the lesson.
 */
public double getDiscountedLessonCost(double discount)
```

(c) The programmer would like to add a method called `getInstructorWithMostLessons`, which returns the name of the instructor who is currently teaching the most lessons.

Write a description of how you would change the `Lesson` and `MusicSchool` classes in order to support this modification.

Make sure to include the following in your response.

- Write the method header for the `getInstructorWithMostLessons` method.
- Identify any new or modified variables, constructors, or methods aside from the `getInstructorWithMostLessons` method. **Do not write the program code for this change.**
- Describe, for each new or revised variable, constructor, or method, how it would change or be implemented, including visibility and type. You do not need to describe the implementation of the `getInstructorWithMostLessons` method. **Do not write the program code for this change.**

Unit2_objectsQuestionsPartBJLC9_18_2023_

53. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

6. This question involves translating a string consisting of a single uppercase letter based on a substitution rule.

The class `Secret` contains methods used to translate letters. You will write one method in this class. A partial declaration of the `Secret` class is shown below.

```
public class Secret
{
  /** Returns the single-character String produced by applying the
  substitution rule to
    *  the given letter, as described in part (a)
    *  Precondition: letter consists of one uppercase letter.
    *                rule has at least 2 letters and all letters are
  uppercase and unique.
   */
  public static String newLetter(String letter, String rule)
  {  /* to be implemented in part (a) */  }

  // There may be variables and methods that are not shown.
}
```

(a) Write the `Secret` method `newLetter`, which is used to translate a string consisting of a single uppercase letter based on a substitution rule. The parameter `letter` is the one-letter string to be translated. The parameter `rule` is the substitution rule.

A substitution rule is represented by a string of at least two letters. All the letters in the substitution rule are uppercase and unique. A sample substitution rule is `"ABC"`.

The substitution rule is used to translate an uppercase letter as follows.

- If the letter is contained in the rule, the method returns the next letter in the rule. For example, if the rule is `"CBA"`, `"C"` would be translated to `"B"`, and `"B"` would be translated to `"A"`. If the letter appears at the end of the rule, the first letter of the rule is returned. Using the rule `"CBA"`, `"A"` would be translated to `"C"`.

- If the letter does not appear in the rule, it is returned. For example, `"F"` would be returned using the rule `"CBA"` because `"F"` does not appear in the rule.

The following table shows the results of several calls to `newLetter`.

Method Call	Return Value
`Secret.newLetter("G", "GXAD")`	`"X"`
`Secret.newLetter("D", "GXAD")`	`"G"`
`Secret.newLetter("M", "GXAD")`	`"M"`

Complete method `newLetter`.

```
/** Returns the single-character String produced by applying the
substitution rule to
  *  the given letter, as described in part (a)
  *  Precondition: letter consists of one uppercase letter.
  *                rule has at least 2 letters and all letters are
uppercase and unique.
 */
public static String newLetter(String letter, String rule)
```

(b) The programmer would like to modify the `Secret` class so that the substitution rule is stored in the class instead of passed as a parameter. The modification would need to support the ability for a calling program to change the rule.

Write a description of how you would change the `Secret` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

54. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

6. The `MysteryNumber` class contains methods used to analyze properties of numbers. You will write one method of the `MysteryNumber` class.

```
public class MysteryNumber
{
 /** Returns true if num is a mystery number and false otherwise */
 private static boolean isMystery(int num)
 {  /* implementation not shown */  }

 /** Returns the nth mystery number starting with 1, as described in
 part (a)
  *  Precondition: 1 <= n
  *                There are at least n mystery numbers between 1 and
  *             Integer.MAX_VALUE.
  */
 public static int nthMysteryNumber(int n)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the static method `nthMysteryNumber`, which returns the nth mystery number, starting with 1. A helper method, `isMystery`, has been provided. The method returns `true` if its parameter is a mystery number and returns `false` otherwise.

Assume that the following numbers are the first 7 mystery numbers and that there are more mystery numbers than the ones shown below.

$1, 3, 6, 7, 11, 12, 20$

The following table shows examples of values returned by `nthMysteryNumber`.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Method Call	Return Value
`MysteryNumber.nthMysteryNumber(3)`	6
`MysteryNumber.nthMysteryNumber(5)`	11

Complete method `nthMysteryNumber`. You must use `isMystery` appropriately to receive full credit.

```
/** Returns the nth mystery number starting with 1, as described in part
(a)
 *  Precondition: 1 <= n
 *                There are at least n mystery numbers between 1 and
 *              Integer.MAX_VALUE.
 */
public static int nthMysteryNumber(int n)
```

(b) A programmer wants to modify the `MysteryNumber` class so that the `nthMysteryNumber` method returns the `n`th mystery number that is greater than or equal to a given integer value. For example, in one call to `nthMysteryNumber`, the method might return the third mystery number that is greater than or equal to 5, and in another call to `nthMysteryNumber`, the method might return the fifth mystery number that is greater than or equal to 11.

Write a description of how you would change the `MysteryNumber` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

55. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves simulating the behavior of a dog, based on `boolean` values obtained by the `getResponse` method in the following `DogAnalysis` class. You will write one method in the class.

```
public class DogAnalysis
{
  /** Returns true if the dog sat on command and returns false otherwise
  */
  public static boolean getResponse()
  {  /* implementation not shown */  }

  /** Returns the number of times the dog was asked to sit, as described
  in part (a)
   *  Precondition: n >= 1
   */
  public static int numUntilObedient(int n)
  {  /* to be implemented in part (a) */  }

  // There may be variables and methods that are not shown.
}
```

(a) Write the method `numUntilObedient`, which simulates repeatedly asking the dog to sit until the dog sits on command `n` times in a row. The `numUntilObedient` method returns the number of times the dog was commanded to sit before the dog obeyed the command `n` times in a row.

The helper method `getResponse` simulates the dog's response to the sit command. It returns `true` if the dog sat on command and returns `false` otherwise.

For example, if `n` is `3` and the values returned from `getResponse` are `true`, `true`, `false`, `true`, `false`, `true`, `true`, and `true`, the `numUntilObedient` method should return `8` since the dog obeyed the command on the 6th, 7th, and 8th calls to `getResponse`.

Complete method `numUntilObedient`.

```
/** Returns the number of times the dog was asked to sit, as described in
part (a)
 *  Precondition: n >= 1
 */
public static int numUntilObedient(int n)
```

(b) A programmer wants to modify the `DogAnalysis` class so that the `numUntilObedient` method is given a value that represents the maximum number of times the dog should be commanded to sit. The method would return `true` if the dog sat `n` times in a row within that maximum number of attempts and would return `false` otherwise.

Write a description of how you would change the method in the `DogAnalysis` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

56. Consider the code segment below.

```
int a = 1988;
int b = 1990;

String claim = " that the world's athletes " +
          "competed in Olympic Games in ";

String s = "It is " + true + claim + a +
       " but " + false + claim + b + ".";

System.out.println(s);
```

What, if anything, is printed when the code segment is executed?

(A) `It is trueclaima but falseclaimb.`

(B) `It is trueclaim1998 but falseclaim1990.`

(C) `It is true that the world's athletes competed in Olympic Games in a but false that the world's athletes competed in Olympic Games in b.`

(D) `It is true that the world's athletes competed in Olympic Games in 1988 but false that the world's athletes competed in Olympic Games in 1990.`

(E) Nothing is printed because the code segment does not compile.

Unit2_objectsQuestionsPartBJLC9_18_2023_

57. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves a two-dimensional array of integers that represents a collection of randomly generated data. A partial declaration of the `Data` class is shown. You will write two methods of the `Data` class.

```
public class Data
{
     public static final int MAX = /* value not shown */;
     private int[][] grid;

     /** Fills all elements of grid with randomly generated values, as
  described in part (a)
       * Precondition: grid is not null.
       *     grid has at least one element.
       */
     public void repopulate()
     { /* to be implemented in part (a) */ }

     /** Returns the number of columns in grid that are in increasing
  order, as described
       * in part (b)
       * Precondition: grid is not null.
       *     grid has at least one element.
       */
     public int countIncreasingCols()
     { /* to be implemented in part (b) */ }

     // There may be instance variables, constructors, and methods that
  are not shown.
}
```

Write the `repopulate` method, which assigns a newly generated random value to each element of `grid`. Each value is computed to meet all of the following criteria, and all valid values must have an equal chance of being generated.

- The value is between 1 and `MAX`, inclusive.
- The value is divisible by 10.
- The value is not divisible by 100.

Complete the `repopulate` method.

```
/** Fills all elements of grid with randomly generated values, as
described in part (a)
 * Precondition: grid is not null.
 *     grid has at least one element.
 */
public void repopulate()
```

Write the `countIncreasingCols` method, which returns the number of columns in `grid` that are in increasing order. A column is considered to be in increasing order if the element in each row after the first row is greater than or equal to the element in the previous row. A column with only one row is considered to be in increasing order.

The following examples show the `countIncreasingCols` return values for possible contents of `grid`.

The return value for the following contents of `grid` is `1`, since the first column is in increasing order but the second and third columns are not.

10	50	40
20	40	20
30	50	30

The return value for the following contents of `grid` is `2`, since the first and third columns are in increasing order but the second and fourth columns are not.

10	540	440	440
220	450	440	190

Complete the `countIncreasingCols` method.

```
/** Returns the number of columns in grid that are in increasing order,
as described
 * in part (b)
 * Precondition: grid is not null.
 *     grid has at least one element.
 */
public int countIncreasingCols()
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

58. Consider the `processWords` method. Assume that each of its two parameters is a `String` of length two or more.

```
public void processWords(String word1, String word2)
{
     String str1 = word1.substring(0, 2);
     String str2 = word2.substring(word2.length() - 1);
     String result = str2 + str1;
     System.out.println(result.indexOf(str2));
}
```

Which of the following best describes the value printed when `processWords` is called?

(A) The value `0` is always printed.

(B) The value `1` is always printed.

(C) The value `result.length() - 1` is printed.

(D) A substring containing the last character of `word2` is printed.

(E) A substring containing the last two characters of `word2` is printed.

Unit2_objectsQuestionsPartBJLC9_18_2023_

59. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. The `PrimeNumber` class contains methods used with prime numbers. You will write one method of the `PrimeNumber` class.

```
public class PrimeNumber
{
 /** Returns true if num is a prime number and false otherwise */
 private static boolean isPrimeNumber(int num)
 {  /* implementation not shown */  }

 /** Returns the proportion of numbers between start and end, inclusive,
 that are
   *   prime, as described in part (a)
  *  Precondition: 1 < start <= end <= Integer.MAX_VALUE
  */
 public static double propOfPrimes(int start, int end)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the method `propOfPrimes`, which returns the proportion of numbers between `start` and `end`, inclusive, that are prime. A helper method, `isPrimeNumber`, has been provided. The `isPrimeNumber` method returns `true` if its parameter is a prime number and returns `false` otherwise.

For example, there are 10 numbers between 5 and 14, inclusive, and 4 of them are prime (5, 7, 11, and 13), so the method call `PrimeNumber.propOfPrimes(5, 14)` returns `0.4`.

You must use `isPrimeNumber` appropriately to receive full credit.

Complete method `propOfPrimes`.

```
/** Returns the proportion of numbers between start and end, inclusive,
```

```
that are prime,
  *  as described in part (a)
 *  Precondition: 1 < start <= end <= Integer.MAX_VALUE
 */
public static double propOfPrimes(int start, int end)
```

(b) A programmer wants to modify the `PrimeNumber` class so that the `propOfPrimes` method always returns the proportion of numbers between 2 and a given number, inclusive, that are prime.

Write a description of how you would change the `PrimeNumber` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

60. Consider the following method.

```
public double puzzle(int x)
{
     Double y = x / 2.0;
     y /= 2;

     return y.doubleValue();
}
```

Assume that the method call `puzzle(3)` appears in a method in the same class as `puzzle`. What value is returned as a result of the method call?

(A) `0.0`

(B) `0.5`

(C) `0.75`

(D) `1.0`

(E) `1.5`

61. Which of the following statements assigns a random integer between 25 and 60, inclusive, to `rn` ?

(A) `int rn = (int) (Math.random() * 25) + 36;`

(B) `int rn = (int) (Math.random() * 25) + 60;`

(C) `int rn = (int) (Math.random() * 26) + 60;`

(D) `int rn = (int) (Math.random() * 36) + 25;`

(E) `int rn = (int) (Math.random() * 60) + 25;`

Unit2_objectsQuestionsPartBJLC9_18_2023_

62. Which of the following statements assigns a random integer between `1` and `10`, inclusive, to `rn` ?

(A) `int rn = (int) (Math.random()) * 10;`

(B) `int rn = (int) (Math.random()) * 10 + 1;`

(C) `int rn = (int) (Math.random() * 10);`

(D) `int rn = (int) (Math.random() * 10) + 1;`

(E) `int rn = (int) (Math.random() + 1) * 10;`

63. Consider the following code segment, which is intended to assign to `num` a random integer value between `min` and `max`, inclusive. Assume that `min` and `max` are integer variables and that the value of `max` is greater than the value of `min`.

```
double rn = Math.random();
int num = /* missing code */;
```

Which of the following could be used to replace `/* missing code */` so that the code segment works as intended?

(A) `(int) (rn * max) + min`

(B) `(int) (rn * max) + min - 1`

(C) `(int) (rn * (max - min)) + min`

(D) `(int) (rn * (max - min)) + 1`

(E) `(int) (rn * (max - min + 1)) + min`

64. Consider the following code segment. Assume that `a` is greater than zero.

```
int a = /* value not shown */;
int b = a + (int) (Math.random() * a);
```

Which of the following best describes the value assigned to `b` when the code segment is executed?

(A) `a`

(B) `2 * a`

(C) A random integer between `0` and `a - 1`, inclusive

(D) A random integer between `a` and `2 * a`, inclusive

(E) A random integer between `a` and `2 * a - 1`, inclusive

Unit2_objectsQuestionsPartBJLC9_18_2023_

65. Assume that the following variable declarations have been made.

```
double d = Math.random();

double r;
```

Which of the following assigns a value to `r` from the uniform distribution over the range $0.5 \leq r < 5.5$?

(A) `r = d + 0.5;`

(B) `r = d + 0.5 * 5.0;`

(C) `r = d * 5.0;`

(D) `r = d * 5.0 + 0.5;`

(E) `r = d * 5.5;`

66. This question involves the implementation and extension of a RandomStringChooser class.

a. A RandomStringChooser object is constructed from an array of non-null String values. When the object is first constructed, all of the strings are considered available. The RandomStringChooser class has a getNext method, which has the following behavior. A call to getNext returns a randomly chosen string from the available strings in the object. Once a particular string has been returned from a call to getNext, it is no longer available to be returned from subsequent calls to getNext. If no strings are available to be returned, getNext returns "NONE".

The following code segment shows an example of the behavior of RandomStringChooser.

```
String[] wordArray = {"wheels", "on", "the", "bus"};
RandomStringChooser sChooser = new RandomStringChooser(wordArray);
for (int k = 0; k < 6; k++)
{
   System.out.print(sChooser.getNext() + " ");
}
```

One possible output is shown below. Because sChooser has only four strings, the string "NONE" is printed twice.

```
bus the wheels on NONE NONE
```

Write the entire RandomStringChooser class. Your implementation must include an appropriate constructor and any necessary methods. Any instance variables must be private. The code segment in the example above should have the indicated behavior (that is, it must compile and produce a result like the possible output shown). Neither the constructor nor any of the methods should alter the parameter passed to the constructor, but your implementation may copy the contents of the array.

b. The following partially completed RandomLetterChooser class is a subclass of the RandomStringChooser class. You will write the constructor for the RandomLetterChooser class.

```
public class RandomLetterChooser extends RandomStringChooser
{
   /** Constructs a random letter chooser using the given string str.
    *  Precondition: str contains only letters.
    */
   public RandomLetterChooser(String str)
   {  /* to be implemented in part (b) */  }

   /** Returns an array of single-letter strings.
    *  Each of these strings consists of a single letter from str. Element k
    *  of the returned array contains the single letter at position k of str.
    *  For example, getSingleLetters("cat") returns the
    *  array { "c", "a", "t" }.
    */
   public static String[] getSingleLetters(String str)
   {  /* implementation not shown */  }
}
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

The following code segment shows an example of using RandomLetterChooser.

```
RandomLetterChooser letterChooser = new RandomLetterChooser("cat");
for (int k = 0; k < 4; k++)
{
   System.out.print(letterChooser.getNext());
}
```

The code segment will print the three letters in "cat" in one of the possible orders. Because there are only three letters in the original string, the code segment prints "NONE" the fourth time through the loop. One possible output is shown below.

```
actNONE
```

Assume that the RandomStringChooser class that you wrote in part (a) has been implemented correctly and that getSingleLetters works as specified. You must use getSingleLetters appropriately to receive full credit.

Complete the RandomLetterChooser constructor below.

```
/** Constructs a random letter chooser using the given string str.
 *  Precondition: str contains only letters.
 */
public RandomLetterChooser(String str)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

67. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. The `NumberProperties` class contains methods used to determine various properties of numbers. You will write one method of the `NumberProperties` class.

```
public class NumberProperties
{
 /** Returns true if num is a perfect square and false otherwise */
 private static boolean isSquare(int num)
 {  /* implementation not shown */  }

 /** Returns true if num is a perfect cube and false otherwise */
 private static boolean isCube(int num)
 {  /* implementation not shown */  }

 /** Returns the ratio of the sum of all the perfect cubes between start
 and end,
   *   inclusive, to the sum of all the perfect squares between start
 and end, inclusive,
   *   as described in part (a)
  *  Precondition: 1 <= start <= end <= Integer.MAX_VALUE
  *                There is at least one perfect square between start
 and end,
  *             inclusive.
  */
 public static double ratioOfCubeSumsToSquareSums(int start, int end)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the method `ratioOfCubeSumsToSquareSums`, which returns the ratio of the sum of all perfect cubes between `start` and `end`, inclusive, to the sum of all perfect squares between `start` and

`end`, inclusive.

Two helper methods, `isSquare` and `isCube`, have been provided. The `isSquare` method returns `true` if its parameter is a perfect square and returns `false` otherwise. The `isCube` method returns `true` if its parameter is a perfect cube and returns `false` otherwise.

For example, of the numbers between 5 and 30, inclusive, two are perfect cubes (8 and 27) and three are perfect squares (9, 16, and 25). The sum of the two perfect cubes is 35 and the sum of the three perfect squares is 50. The method call `NumberProperties.ratioOfCubeSumsToSquareSums(5, 30)` returns the ratio of the sums `35` and `50`, which is `0.7`.

You must use `isSquare` and `isCube` appropriately to receive full credit. Assume that there is at least one perfect square between `start` and `end`, inclusive.

Complete method `ratioOfCubeSumsToSquareSums`.

```
/** Returns the ratio of the sum of all the perfect cubes between start
and end, inclusive,
  *  to the sum of all the perfect squares between start and end,
inclusive,
  *  as described in part (a)
 *  Precondition: 1 <= start <= end <= Integer.MAX_VALUE
  *              There is at least one perfect square between start and
end, inclusive.
 */
public static double ratioOfCubeSumsToSquareSums(int start, int end)
```

(b) A programmer wants to modify the `NumberProperties` class so that the `ratioOfCubeSumsToSquareSums` method examines values beginning at `start` and continues until it finds a given positive number of perfect squares.

Write a description of how you would change the `NumberProperties` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

68. Consider the following method.

```
public static String rearrange(String str)
{
     String temp = "";
     for (int i = str.length() - 1; i > 0; i--)
     {
         temp += str.substring(i - 1, i);
     }
     return temp;
}
```

What, if anything, is returned by the method call `rearrange("apple")` ?

(A) `"appl"`

(B) `"apple"`

(C) `"elppa"`

(D) `"lppa"`

(E) Nothing is returned due to a run-time error.

Unit2_objectsQuestionsPartBJLC9_18_2023_

69. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

A student plans to analyze product reviews found on a Web site by looking for keywords in posted reviews. The `ProductReview` class, shown below, is used to represent a single review. A product review consists of a product name and a review of that product.

```
public class ProductReview
{
     private String name;
     private String review;
  /** Constructs a ProductReview object and initializes the instance
variables. */
     public ProductReview(String pName, String pReview)
     {
         name = pName;
         review = pReview;
     }
  /** Returns the name of the product. */
     public String getName()
     { return name; }
  /** Returns the review of the product. */
     public String getReview()
     { return review; }
}
```

The `ReviewCollector` class, shown below, is used to represent a collection of reviews to be analyzed.

```
public class ReviewCollector
{
     private ArrayList<ProductReview> reviewList;
     private ArrayList<String> productList;
  /** Constructs a ReviewCollector object and initializes the instance
variables. */
     public ReviewCollector()
     {
         reviewList = new ArrayList<ProductReview>();
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
        productList = new ArrayList<String>();
    }
  /** Adds a new review to the collection of reviews, as described in
part (a). */
    public void addReview(ProductReview prodReview)
    { /* to be implemented in part (a) */ }
  /** Returns the number of good reviews for a given product name, as
described in part (b). */
    public int getNumGoodReviews(String prodName)
    { /* to be implemented in part (b) */ }
  // There may be instance variables, constructors, and methods not
shown.
}
```

(a) Write the `addReview` method, which adds a single product review, represented by a `ProductReview` object, to the `ReviewCollector` object. The `addReview` method does the following when it adds a product review.

- The `ProductReview` object is added to the `reviewList` instance variable.
- The product name from the `ProductReview` object is added to the `productList` instance variable if the product name is not already found in `productList`.

Elements may be added to `reviewList` and `productList` in any order.

Complete method `addReview`.

```
/** Adds a new review to the collection of reviews, as described in part
(a). */
public void addReview(ProductReview prodReview)
```

(b) Write the `getNumGoodReviews` method, which returns the number of *good* reviews for a given product name. A review is considered good if it contains the string `"best"` (all lowercase). If there are no reviews with a matching product name, the method returns `0`. Note that a review that contains `"BEST"` or `"Best"` is not considered a good review (since not all the letters of `"best"` are lowercase), but a review that contains `"asbestos"` is considered a good review (since all the letters of `"best"` are lowercase).

Complete method `getNumGoodReviews`.

```
/** Returns the number of good reviews for a given product name, as
described in part (b). */
public int getNumGoodReviews(String prodName)
```

Class information for this question

```
public class ProductReview
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
private String name
private String review
public ProductReview(String pName, String pReview)
public String getName()
public String getReview()
public class ReviewCollector
private ArrayList<ProductReview> reviewList
private ArrayList<String> productList
public ReviewCollector()
public void addReview(ProductReview prodReview)
public int getNumGoodReviews(String prodName)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

70. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

4. This question involves analyzing integer values that are obtained using the `getInt` method in the following `TargetIntegers` class. You will write one method in the class.

```
public class TargetIntegers
{
 /** Returns an int from simulated user input */
 public static int getInt()
 {  /* implementation not shown */  }

 /** Returns true if x is considered a target number; returns false
 otherwise
   *   Some target numbers are positive and some are negative.
  */
 public static boolean isTarget(int x)
 {  /* implementation not shown */  }

 /** Analyzes sampleSize values obtained using the getInt method, as
 described
  *   in part (a)
  *   Precondition: sampleSize is a positive even integer.
  */
 public static boolean runSimulation(int sampleSize)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Some integer values are considered target numbers. A helper method, `isTarget`, has been provided. Method `isTarget` returns `true` if its parameter is a target number and returns `false` otherwise. Some target numbers are positive, and some are negative.

Method `runSimulation` obtains `sampleSize` values using the `getInt` method. The method returns `true` if there are more positive target values (target values that are greater than 0) in the first half of the obtained values than in the second half of the obtained values.

Complete method `runSimulation`. You must use `getInt` and `isTarget` appropriately in order to receive full credit.

```
/** Analyzes sampleSize values obtained using the getInt method, as
described in
 *  part (a)
 *   Precondition: sampleSize is a positive even integer.
 */
public static boolean runSimulation(int sampleSize)
```

(b) A programmer wants to modify the `TargetIntegers` class so that the number of values to be analyzed by the `runSimulation` method is maintained in a variable with a value that cannot change once it is initialized.

Write a description of how you would change the `TargetIntegers` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

71. Consider the following method.

```
public static String abMethod(String a, String b)
{
     int x = a.indexOf(b);
     while (x >= 0)
     {
         a = a.substring(0, x) + a.substring(x + b.length());
         x = a.indexOf(b);
     }
     return a;
}
```

What, if anything, is returned by the method call `abMethod("sing the song", "ng")` ?

(A) `"si"`

(B) `"si the so"`

(C) `"si the song"`

(D) `"sig the sog"`

(E) Nothing is returned because a `StringIndexOutOfBoundsException` is thrown.

72. The StringChecker interface describes classes that check if strings are valid, according to some criterion.

```
public interface StringChecker
{
    /** Returns true if str is valid. */
    boolean isValid(String str);
}
```

A CodeWordChecker is a StringChecker. A CodeWordChecker object can be constructed with three parameters: two integers and a string. The first two parameters specify the minimum and maximum code word lengths, respectively, and the third parameter specifies a string that must not occur in the code word. A CodeWordChecker object can also be constructed with a single parameter that specifies a string that must not occur in the code word; in this case the minimum and maximum lengths will default to 6 and 20, respectively.

The following examples illustrate the behavior of CodeWordChecker objects.

Example 1

```
StringChecker sc1 = new CodeWordChecker(5, 8, "$");
```

Valid code words have 5 to 8 characters and must not include the string "$".

Method call	Return value	Explanation
`sc1.isValid("happy")`	`true`	The code word is valid.
`sc1.isValid("happy$")`	`false`	The code word contains `"$"`.
`sc1.isValid("Code")`	`false`	The code word is too short.
`sc1.isValid("happyCode")`	`false`	The code word is too long.

Example 2

```
StringChecker sc2 = new CodeWordChecker("pass");
```

Valid code words must not include the string "pass". Because the bounds are not specified, the length bounds are 6 and 20, inclusive.

Method call	Return value	Explanation
`sc2.isValid("MyPass")`	`true`	The code word is valid.
`sc2.isValid("Mypassport")`	`false`	The code word contains `"pass"`.
`sc2.isValid("happy")`	`false`	The code word is too short.
`sc2.isValid("1,000,000,000,000,000")`	`false`	The code word is too long.

Unit2_objectsQuestionsPartBJLC9_18_2023_

Write the complete CodeWordChecker class. Your implementation must meet all specifications and conform to all examples.

Unit2_objectsQuestionsPartBJLC9_18_2023_

73. This question involves the process of taking a list of words, called wordList, and producing a formatted string of a specified length. The list wordList contains at least two words, consisting of letters only.

When the formatted string is constructed, spaces are placed in the gaps between words so that as many spaces as possible are evenly distributed to each gap. The equal number of spaces inserted into each gap is referred to as the *basic gap width*. Any *leftover spaces* are inserted one at a time into the gaps from left to right until there are no more leftover spaces.

The following three examples illustrate these concepts. In each example, the list of words is to be placed into a formatted stri length 20.

Example 1: `wordList: ["AP", "COMP", "SCI", "ROCKS"]`

Total number of letters in words: 14
Number of gaps between words: 3
Basic gap width: 2
Leftover spaces: 0

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	P			C	O	M	P			S	C	I			R	O	C	K	S

Example 2: `wordList: ["GREEN", "EGGS", "AND", "HAM"]`

Total number of letters in words: 15
Number of gaps between words: 3
Basic gap width: 1
Leftover spaces: 2

The leftover spaces are inserted one at a time between the words from left to right until there are no more leftover spaces. In this example, the first two gaps get an extra space.

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
G	R	E	E	N			E	G	G	S			A	N	D		H	A	M

Example 3: `wordList: ["BEACH", "BALL"]`

Total number of letters in words: 9
Number of gaps between words: 1
Basic gap width: 11
Leftover spaces: 0

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
B	E	A	C	H												B	A	L	L

Unit2_objectsQuestionsPartBJLC9_18_2023_

You will implement three static methods in a class named StringFormatter that is not shown.

a. Write the StringFormatter method totalLetters, which returns the total number of letters in the words in its paramete wordList. For example, if the variable List<String> words is ["A", "frog", "is"], then the call StringFormatter.totalLetters(words) returns 7. You may assume that all words in wordList consist of one or more le

Complete method totalLetters below.

```
/** Returns the total number of letters in wordList.
 *  Precondition: wordList contains at least two words, consisting of letters only.
 */
public static int totalLetters(List<String> wordList)
```

b. Write the StringFormatter method basicGapWidth, which returns the basic gap width as defined earlier.

Class information for this question

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

Assume that totalLetters works as specified regardless of what you wrote in part (a). You must use totalLetters appropriately to receive full credit.

Complete method basicGapWidth below.

```
/** Returns the basic gap width when wordList is used to produce
 *  a formatted string of formattedLen characters.
 *  Precondition: wordList contains at least two words, consisting of letters only.
 *                formattedLen is large enough for all the words and gaps.
 */
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
```

c. Write the StringFormatter method format, which returns the formatted string as defined earlier. The StringFormatte class also contains a method called leftoverSpaces, which has already been implemented. This method returns the number of leftover spaces as defined earlier and is shown below.

Unit2_objectsQuestionsPartBJLC9_18_2023_

```
/** Returns the number of leftover spaces when wordList is used to produce
 *  a formatted string of formattedLen characters.
 *  Precondition: wordList contains at least two words, consisting of letters only.
 *                formattedLen is large enough for all the words and gaps.
 */
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
{  /* implementation not shown */  }
```

Class information for this question

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

Assume that basicGapWidth works as specified, regardless of what you wrote in part (b). You must use basicGapW and leftoverSpaces appropriately to receive full credit.

Complete method format below.

```
/** Returns a formatted string consisting of the words in wordList separated by spaces.
 *  Precondition: The wordList contains at least two words, consisting of letters only.
 *                formattedLen is large enough for all the words and gaps.
 *  Postcondition: All words in wordList appear in the formatted string.
 *    - The words appear in the same order as in wordList.
 *    - The number of spaces between words is determined by basicGapWidth and the
 *      distribution of leftoverSpaces from left to right, as described in the question.
 */
public static String format(List<String> wordList, int formattedLen)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

74. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

Consider the following `Thing` class. Each `Thing` object has a name attribute, which can be set in the constructor or by using the `setName` method. The name of a `Thing` object can be returned by the `getName` method.

```
public class Thing
{
     // attributes not shown

     /** Constructs a new Thing named myName
       */
     public Thing(String myName)
     { /* implementation not shown */ }

     /** Returns this Thing's name
       */
     public String getName()
     { /* implementation not shown */ }

     /** Sets this Thing's name to newName
       */
     public void setName(String newName)
     { /* implementation not shown */ }

     /** Returns a message as described in part (b)
       */
     public void printMessage()
     { /* implementation not shown */ }
}
```

(a) Write a statement to create a new `Thing` object `snack` that has the name `"potato chip"`.

Write the statement below.

(b) The `Thing` method `printMessage` prints a string consisting of the name of the object followed by `"_is_great"`.

Suppose the name of the `Thing` object `favFood` is `"pizza"`. Write a statement that uses the `printMessage`

method to print the string "pizza_is_great".

Write the statement below.

(c) Write a code segment to change the name of the Thing object something such that the new name consists of the old name with one character removed at random. For example, if something has name "ABCD", its new name could be set to "ACD".

Write the code segment below.

75. Consider the following method.

```
public int timesTwo (int n)
{
     return n * 2;
}
```

The following code segment appears in a method in the same class as the timesTwo method.

```
Integer val = 10;
int result1 = timesTwo(val);
Integer result2 = result1;
System.out.print(result2);
```

What, if anything, is printed as a result of executing the code segment?

(A) 10

(B) 20

(C) Nothing; the code segment will not compile because timesTwo cannot accept an Integer parameter.

(D) Nothing; the code segment will not compile because the value returned by timesTwo cannot be assigned to result1.

(E) Nothing; the code segment will not compile because the int variable result1 cannot be assigned to the Integer variable result2.

76. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves analyzing simulated flights a traveler might take to determine the number of points the traveler would earn. Travelers earn points based on the number of miles traveled. The points can be used for special services. The number of miles traveled on each simulated flight is obtained by calling the `milesTraveled` method in the following `AirTravel` class. You will write one method in the class.

```
public class AirTravel
{
 /** Returns the number of miles traveled in a simulated flight. Assume
 the value returned
  *  is greater than 0.
  */
 public static int milesTraveled()
 {  /* implementation not shown */  }

 /** Returns the total number of points earned for numFlights flights,
 as described in
  *  part (a)
  *  Precondition: 0 < numFlights
  */
 public static int totalPointsEarned(int numFlights)
 {  /* to be implemented in part (a) */  }

 // There may be variables and methods that are not shown.
}
```

(a) Write the `totalPointsEarned` method, which returns the total number of points earned for `numFlights` simulated flights. Miles for each simulated flight are retrieved by calling the `milesTraveled` method, which returns the number of miles traveled in one flight. After the miles for all of the flights have been accumulated, points are awarded according to the following rules.

- For the first 1,000 miles traveled, 1 point is earned per mile.
- For additional miles traveled up to 10,000 miles, 2 points are earned per mile.

- For additional miles traveled beyond $10{,}000$ miles, 5 points are earned per mile.

For example, for a total of $12{,}000$ miles, a traveler earns $1{,}000$ points for the first $1{,}000$ miles, plus $2 \times 9{,}000$ points for miles $1{,}001$ to $10{,}000$, plus $5 \times 2{,}000$ points for miles $10{,}001$ to $12{,}000$. The total number of points earned is $1{,}000 + (2 \times 9{,}000) + (5 \times 2{,}000) = 29{,}000$.

Complete method `totalPointsEarned`.

```
/** Returns the total number of points earned for numFlights flights, as
described in
 *  part (a)
 *  Precondition: 0 < numFlights
 */
public static int totalPointsEarned(int numFlights)
```

(b) A programmer wants to modify the `AirTravel` class so that the number of points returned by the `totalPointsEarned` method is limited to a given maximum value. If the number of points earned exceeds the maximum value, the maximum value is returned. This new implementation would also support the ability for the airline to change the maximum possible number of points. The programmer would like to implement this change without making any changes to the signature of the `totalPointsEarned` method or overloading `totalPointsEarned`.

Write a description of how you would change the `AirTravel` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Unit2_objectsQuestionsPartBJLC9_18_2023_

77. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves the `WordMatch` class, which stores a secret string and provides methods that compare other strings to the secret string. You will write two methods in the `WordMatch` class.

```
public class WordMatch
{
     /** The secret string. */
     private String secret;

     /** Constructs a WordMatch object with the given secret string
       * of lowercase letters.
       */
     public WordMatch(String word)
     {
         /* implementation not shown */
     }

     /** Returns a score for guess, as described in part (a).
       * Precondition: 0 < guess.length() <= secret.length()
       */
     public int scoreGuess(String guess)
     { /* to be implemented in part (a) */ }

     /** Returns the better of two guesses, as determined by scoreGuess
       * and the rules for a tie-breaker that are described in part (b).
       * Precondition: guess1 and guess2 contain all lowercase letters.
       * guess1 is not the same as guess2.
       */
     public String findBetterGuess(String guess1, String guess2)
     { /* to be implemented in part (b) */ }
}
```

(a) Write the `WordMatch` method `scoreGuess`. To determine the score to be returned, `scoreGuess` finds the number of times that `guess` occurs as a substring of `secret` and then multiplies that number by the square of the length of `guess`. Occurrences of `guess` may overlap within `secret`.

Assume that the length of `guess` is less than or equal to the length of `secret` and that `guess` is not an

empty string.

The following examples show declarations of a `WordMatch` object. The tables show the outcomes of some possible calls to the `scoreGuess` method.

```
WordMatch game = new WordMatch("mississippi");
```

Value of `guess`	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of `guess`)	Return Value of `game.scoreGuess(guess)`
`"i"`	4	`4 * 1 * 1 = 4`	`4`
`"iss"`	2	`2 * 3 * 3 = 18`	`18`
`"issipp"`	1	`1 * 6 * 6 = 36`	`36`
`"mississippi"`	1	`1 * 11 * 11 = 121`	`121`

```
WordMatch game = new WordMatch("aaaabb");
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

Value of `guess`	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of `guess`)	Return Value of `game.scoreGuess(guess)`
`"a"`	4	`4 * 1 * 1 = 4`	4
`"aa"`	3	`3 * 2 * 2 = 12`	12
`"aaa"`	2	`2 * 3 * 3 = 18`	18
`"aabb"`	1	`1 * 4 * 4 = 16`	16
`"c"`	0	`0 * 1 * 1 = 0`	0

Complete the `scoreGuess` method.

```
/** Returns a score for guess, as described in part (a).
 * Precondition: 0 < guess.length() <= secret.length()
 */
public int scoreGuess(String guess)
```

(b) Write the `WordMatch` method `findBetterGuess`, which returns the better guess of its two `String` parameters, `guess1` and `guess2`. If the `scoreGuess` method returns different values for `guess1` and `guess2`, then the guess with the higher score is returned. If the `scoreGuess` method returns the same value for `guess1` and `guess2`, then the alphabetically greater guess is returned.

The following example shows a declaration of a `WordMatch` object and the outcomes of some possible calls to the `scoreGuess` and `findBetterGuess` methods.

```
WordMatch game = new WordMatch("concatenation");
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

Method Call	Return Value	Explanation
`game.scoreGuess("ten");`	`9`	`1 * 3 * 3`
`game.scoreGuess("nation");`	`36`	`1 * 6 * 6`
`game.findBetterGuess("ten", "nation");`	`"nation"`	Since `scoreGuess` returns `36` for `"nation"` and `9` for `"ten"`, the guess with the greater score, `"nation"`, is returned.
`game.scoreGuess("con");`	`9`	`1 * 3 * 3`
`game.scoreGuess("cat");`	`9`	`1 * 3 * 3`
`game.findBetterGuess("con", "cat");`	`"con"`	Since `scoreGuess` returns `9` for both `"con"` and `"cat"`, the alphabetically greater guess, `"con"`, is returned.

Complete method `findBetterGuess`.

Assume that `scoreGuess` works as specified, regardless of what you wrote in part (a). You must use `scoreGuess` appropriately to receive full credit.

```
/** Returns the better of two guesses, as determined by scoreGuess
 * and the rules for a tie-breaker that are described in part (b).
 * Precondition: guess1 and guess2 contain all lowercase letters.
 * guess1 is not the same as guess2.
 */
public String findBetterGuess(String guess1, String guess2)
```

Unit2_objectsQuestionsPartBJLC9_18_2023_

78. Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

2. This question involves analyzing words that are generated using the `getWord` method in the following `WordTester` class. You will write one method in the class.

```
public class WordTester
{
 /** Returns a generated word consisting of one or more letters */
 public static String getWord()
 {  /* implementation not shown */  }

 /** Determines whether fewer than 10 percent of n words examined start
 with
   *  firstLetter and are less than maxLength characters, as described
 in
  *  part (a)
  *  Precondition: firstLetter.length() = 1, maxLength > 0
  */
 public static boolean wordChecker(String firstLetter,
      int maxLength, int n)
 {  /* to be implemented in part (a) */  }

 // There may be variables and other methods that are not shown.
}
```

(a) Method `wordChecker` examines `n` words that are generated by calling the `getWord` method repeatedly. The `wordChecker` method returns `true` if fewer than 10 percent of the generated words meet both of the following criteria and returns `false` otherwise.

- The word begins with `firstLetter`.
- The length of the word is less than or equal to `maxLength`.

Complete method `wordChecker`.

```
/** Determines whether fewer than 10 percent of n words examined start
with
  *  firstLetter and are less than maxLength characters, as described in
 *  part (a)
 *  Precondition: firstLetter.length() = 1, maxLength > 0
 */
public static boolean wordChecker(String firstLetter,
     int maxLength, int n)
```

(b) A programmer wants to modify the `WordTester` class so that in the `wordChecker` method, the percentage checked for can vary between method calls. For example, in one call to `wordChecker`, the method might check whether fewer than 15 percent of the words examined start with the letter `"I"`, and in another call to `wordChecker`, the method might check whether fewer than 20 percent of the words examined start with the letter `"J"`. The programmer wants to implement this change without making any changes to the signature of the `wordChecker` method or overloading `wordChecker`.

Write a description of how you would change the `WordTester` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.