Unit 1 JLC 9_5_2023

1. In the code segment below, assume that the `int` variable `n` has been properly declared and initialized. The code segment is intended to print a value that is `1` more than twice the value of `n`.

```
/* missing code */
System.out.print(result);
```

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

I.
```
int result = 2 * n;
result = result + 1;
```

II.
```
int result = n + 1;
result = result * 2;
```

III.
```
int result = (n + 1) * 2;
```

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III

2. Consider the following code segment.

```
int a = 5;
int b = 8;
int c = 3;
System.out.println(a + b / c * 2);
```

What is printed as a result of executing this code?

(A) `2`

(B) `6`

(C) `8`

(D) `9`

(E) `14`

Unit 1 JLC 9_5_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.

1. This question involves a system to manage flights at an airport. Flights are represented by the following `Flight` class.

```
public class Flight
{
 /** Returns the number of passengers on the flight */
 public int getNumPassengers()
 {  /* implementation not shown */  }

 /** Returns the price of a seat on the flight. All seats on a flight
 have the same price.
   *   Postcondition: The value returned is greater than or equal to
 0.0.
  */
 public double getPrice()
 {  /* implementation not shown */  }

 /** Returns the capacity of the flight (the maximum number of
 passengers the flight can
  *  carry)
  */
 public int getCapacity()
 {  /* implementation not shown */  }

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

The flights into and out of an airport are stored in an `Airport` object, which contains a list of all such flights. You will write two methods of the Airport class.

```
public class Airport
{
```

Unit 1 JLC 9_5_2023

```
  /** A list of the flights into and out of this airport
    *   Guaranteed not to be null and to contain only non-null entries
   */
  private ArrayList<Flight> allFlights;

  /** Returns the revenue generated by all flights at the airport, as
  described in part (a) */
  public double getTotalRevenue()
  {  /* to be implemented in part (a) */  }

  /** Updates the list of flights by removing certain flights and returns
  the total number of
    *   passengers whose flights were removed, as described in part (b)
   */
  public int updateFlights()
  {  /* to be implemented in part (b) */  }

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

(a) Write the `Airport` method `getTotalRevenue`. The method returns the total revenue for all flights into and out of the airport. Revenue for a flight is the product of the number of passengers on the flight and the price of a seat on the flight. All seats on a flight have the same price.

Some flights sell more seats than there is capacity for, since passengers sometimes cancel or modify reservations. If there are more passengers on a flight than the flight has capacity for, the revenue for the flight is the product of the capacity and the price of a seat on the flight.

For example, assume that `capitalHub` has been declared as an `Airport` object and `ArrayList allFlights` contains the following flights.

Number of Passengers: `25` Price: `50.00` Capacity: `30`	Number of Passengers: `10` Price: `100.50` Capacity: `60`	Number of Passengers: `50` Price: `200.00` Capacity: `40`	Number of Passengers: `20` Price: `100.00` Capacity: `120`

The following table shows the revenue that would be generated by each flight in `allFlights`.

Number of Passengers	Price of a Seat	Capacity	Revenue Calculation
25	\$50.00	30	$25 \times \$50.00 = \$1{,}250.00$
10	\$100.50	60	$10 \times \$100.50 = \$1{,}005.00$
50	\$200.00	40	$40 \times \$200.00 = \$8{,}000.00$
20	\$100.00	120	$20 \times \$100.00 = \$2{,}000.00$

The call `capitalHub.getTotalRevenue()` should return the value `12255.0`.

Complete method `getTotalRevenue`.

```
/** Returns the revenue generated by all flights at the airport, as
described in part (a) */
public double getTotalRevenue()
```

(b) Write the `Airport` method `updateFlights`. The method removes from the `ArrayList allFlights` any flight where the number of passengers is less than 20 percent of the total capacity. The method should return the total number of passengers whose flight was removed.

For example, assume that `capitalHub` has been declared as an `Airport` object and `ArrayList allFlights` contains the following flights.

Number of Passengers: `25` Price: `50.00` Capacity: `30`	Number of Passengers: `10` Price: `100.50` Capacity: `60`	Number of Passengers: `50` Price: `200.00` Capacity: `40`	Number of Passengers: `20` Price: `100.00` Capacity: `120`

The call `capitalHub.updateFlights()` should return the value `30`, and after the method finished executing, the `ArrayList allFlights` should contain the following two flights.

Number of Passengers: `25` Price: `50.00` Capacity: `30`	Number of Passengers: `50` Price: `200.00` Capacity: `40`

Complete method `updateFlights`.

```
/** Updates the list of flights by removing certain flights and returns
the total number of
  *   passengers whose flights were removed, as described in part (b)
 */
public int updateFlights()
```

(c) A programmer would like to add a method called `getMostLuggageCapacity`, which returns a `Flight` object that can carry the greatest amount of luggage, by weight.

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

Make sure to include the following in your response.

Write the method header for the `getMostLuggageCapacity` method.
Identify any new or modified variables, constructors, or methods aside from the `getMostLuggageCapacity` 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 `getMostLuggageCapacity` method. **Do not write the program code for this change.**

4. Consider the following class declarations.

```
public class Alpha
{
     private int answer()
     {
         return 10;
     }
}

public class Beta
{
     public double sample()
     {
         Alpha item = new Alpha();
         double temp = item.answer();
         return temp * 2.0;
     }
}
```

Which of the following best describes why an error occurs when the classes are compiled?

(A) The class `Alpha` does not have a defined constructor.

(B) The class `Alpha` must be declared as a subclass of `Beta`.

(C) The class `Beta` must be declared as a subclass of `Alpha`.

(D) The `answer` method cannot be accessed from a class other than `Alpha`.

(E) The result of the method call `item.answer()` cannot be assigned to a variable of type `double`.

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 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.

6. **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 APLine is a line defined by the equation $ax + by + c = 0$, where a is not equal to zero, b is not equal to zero, and a, b, and c are all integers. The slope of an APLine is defined to be the double value $-a\ b/$. A point (represented by integers x and y) is on an APLine if the equation of the APLine is satisfied when those x and y values are substituted into the equation. That is, a point represented by x and y is on the line if $ax + by + c$ is equal to 0. Examples of two APLine equations are shown in the following table.

Equation	Slope ($-a$ / b)	Is point (5, -2) on the line?
$5x + 4y - 17 = 0$	-5 / 4 = -1.25	Yes, because $5(5) + 4(-2) + (-17) = 0$
$-25x + 40y + 30 = 0$	25 / 40 = 0.625	No, because $-25(5) + 40(-2) + 30 \neq 0$

Assume that the following code segment appears in a class other than APLine. The code segment shows an example of using the APLine class to represent the two equations shown in the table.

```
APLine line1 = new APLine(5, 4, -17);
double slope1 = line1.getSlope();         // slope1 is assigned -1.25
boolean onLine1 = line1.isOnLine(5, -2); // true because 5(5) + 4(-2) + (-17) = 0

APLine line2 = new APLine(-25, 40, 30);
double slope2 = line2.getSlope();         // slope2 is assigned 0.625
boolean onLine2 = line2.isOnLine(5, -2); // false because -25(5) + 40(-2) + 30 ≠ 0
```

Write the APLine class. Your implementation must include a constructor that has three integer parameters that represent a, b, and c, in that order. You may assume that the values of the parameters representing a and b are not zero. It must also include a method getSlope that calculates and returns the slope of the line, and a method isOnLine that returns true if the point represented by its two parameters (x and y, in that order) is on the APLine and returns false otherwise. Your class must produce the indicated results when invoked by the code segment given above. You may ignore any issues related to integer overflow.

7. In the code segment below, assume that the `int` variables `a` and `b` have been properly declared and initialized.

```
int c = a;
int d = b;
c += 3;
d--;
double num = c;
num /= d;
```

Which of the following best describes the behavior of the code segment?

(A) The code segment stores the value of `(a + 3) / b` in the variable `num`.

(B) The code segment stores the value of `(a + 3) / (b - 1)` in the variable `num`.

(C) The code segment stores the value of `(a + 3) / (b - 2)` in the variable `num`.

(D) The code segment stores the value of `(a + 3) / (1 - b)` in the variable `num`.

(E) The code segment causes a runtime error in the last line of code because `num` is type `double` and `d` is type `int`.

8. Consider the following code segment, which is intended to find the average of two positive integers, `x` and `y`.

```
int x;
int y;
int sum = x + y;
double average = (double) (sum / 2);
```

Which of the following best describes the error, if any, in the code segment?

(A) There is no error, and the code works as intended.

(B) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be less than the expected result for even values of `sum`.

(C) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be greater than the expected result for even values of `sum`.

(D) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be less than the expected result for odd values of `sum`.

(E) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be greater than the expected result for odd values of `sum`.

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.

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.

10. Consider the following static method.

```
public static int calculate(int x)

{

 x = x + x;

 x = x + x;

 x = x + x;



 return x;

}
```

Which of the following can be used to replace the body of calculate so that the modified version of calculate will return the same result as the original version for all x ?

(A) return 3 + x;

(B) return 3 * x;

(C) return 4 * x;

(D) return 6 * x;

(E) return 8 * x;

11. Consider the following static method.

```
public static int calculate(int x)
{
  x = x + x;
  x = x + x;
  x = x + x;
  return x;
}
```

Which of the following can be used to replace the body of calculate so that the modified version of calculate will return the same result as the original version for all x ?

(A) return 2 * x;

(B) return 4 * x;

(C) return 8 * x;

(D) return 3 * calculate(x);

(E) return x + calculate(x - 1);

12. Consider the following code segment.

```
double num = 9 / 4;
System.out.print(num);
System.out.print(" ");
System.out.print((int) num);
```

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

(A) `2 2`

(B) `2.0 2`

(C) `2.0 2.0`

(D) `2.25 2`

(E) `2.25 2.0`

13. Which of the following expressions evaluate to `3.5` ?

I. `(double) 2 / 4 + 3`
II. `(double) (2 / 4) + 3`
III. `(double) (2 / 4 + 3)`

(A) I only
(B) III only
(C) I and II only
(D) II and III only
(E) I, II, and III

14. Consider the following code segment.

```
double x = (int) (5.5 - 2.5);
double y = (int) 5.5 - 2.5;
System.out.println(x - y);
```

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

(A) `-1.0`
(B) `-0.5`
(C) `0.0`
(D) `0.5`
(E) `1.0`

15. Consider the following code segment.

```
int w = 1;
int x = w / 2;
double y = 3;
int z = (int) (x + y);
```

Which of the following best describes the results of compiling the code segment?

(A) The code segment compiles without error.
(B) The code segment does not compile, because the `int` variable `x` cannot be assigned the result of the operation `w / 2`.
(C) The code segment does not compile, because the integer value `3` cannot be assigned to the `double` variable `y`.
(D) The code segment does not compile, because the operands of the addition operator cannot be of different types `int` and `double`.
(E) The code segment does not compile because the result of the addition operation is of type `double` and cannot be cast to an `int`.

16. Consider the following code segment.

```
double x = 4.5;
int y = (int) x * 2;
System.out.print(y);
```

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

Unit 1 JLC 9_5_2023

(A) `8`

(B) `8.0`

(C) `9`

(D) `9.0`

(E) `10`

Unit 1 JLC 9_5_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)
     {
```

Unit 1 JLC 9_5_2023

```
        /* implementation not shown */
    }

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

17. **(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)
```

---

18. The code segment below is intended to calculate the circumference `c` of a circle with the diameter `d` of `1.5`. The circumference of a circle is equal to its diameter times `pi`.

```
/* missing declarations */
c = pi * d;
```

Which of the following variable declarations are most appropriate to replace `/* missing declarations */` in this code segment?

Unit 1 JLC 9_5_2023

(A)
```
int pi = 3.14159;
int d = 1.5;
final int c;
```

(B)
```
final int pi = 3.14159;
int d = 1.5;
int c;
```

(C)
```
final double pi = 3.14159;
double d = 1.5;
double c;
```

(D)
```
double pi = 3.14159;
double d = 1.5;
final double c = 0.0;
```

(E)
```
final double pi = 3.14159;
final double d = 1.5;
final double c = 0.0;
```

19. Consider the following code segment.

```
int a = 5;
int b = 4;
int c = 2;
a *= 3;
b += a;
b /= c;
System.out.print(b);
```

What is printed when the code segment is executed?

(A) 2

(B) 4

(C) 9

(D) 9.5

(E) 19

20. Consider the following method.

```
public int getTheResult(int n)

{

int product = 1;

for (int number = 1; number < n; number++)

{

if (number % 2 == 0)

product *= number;

}

return product;

}
```

What value is returned as a result of the call getTheResult(8) ?

(A) 48
(B) 105
(C) 384
(D) 5040
(E) 40320

21. Consider the following code segment.

```
int x = 5;
int y = 6;
/* missing code */
z = (x + y) / 2;
```

Which of the following can be used to replace `/* missing code */` so that the code segment will compile?

I. `int z = 0;`
II. `int z;`
III. `boolean z = false;`

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

22. A code segment (not shown) is intended to determine the number of players whose average score in a game exceeds `0.5`. A player's average score is stored in `avgScore`, and the number of players who meet the criterion is stored in the variable `count`.

Which of the following pairs of declarations is most appropriate for the code segment described?

(A)
```
double avgScore;
boolean count;
```

(B)
```
double avgScore;
double count;
```

(C)
```
double avgScore;
int count;
```

(D)
```
int avgScore;
boolean count;
```

(E)
```
int avgScore;
int count;
```

23. **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 grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is an integer value that represents a shade of gray. In this question, pixel values can be in the range from 0 through 255, inclusive. A black pixel is represented by 0, and a white pixel is represented by 255.

The declaration of the GrayImage class is shown below. You will write two unrelated methods of the GrayImage class.

```
public class GrayImage
{
  public static final int BLACK = 0;
  public static final int WHITE = 255;

  /** The 2-dimensional representation of this image. Guaranteed not to be null.
   *  All values in the array are within the range [BLACK, WHITE], inclusive.
   */
  private int[][] pixelValues;

  /** @return the total number of white pixels in this image.
   *  Postcondition: this image has not been changed.
   */
  public int countWhitePixels()
  {  /* to be implemented in part (a) */  }

  /** Processes this image in row-major order and decreases the value of each pixel at
   *  position (row, col) by the value of the pixel at position (row + 2, col + 2) if it exists.
   *  Resulting values that would be less than BLACK are replaced by BLACK.
   *  Pixels for which there is no pixel at position (row + 2, col + 2) are unchanged.
   */
  public void processImage()
  {  /* to be implemented in part (b) */  }
}
```

a. Write the method countWhitePixels that returns the number of pixels in the image that contain the value WHITE. For example, assume that pixelValues contains the following image.

	0	1	2	3	4
0	**255**	184	178	84	129
1	84	**255**	**255**	130	84
2	78	**255**	0	0	78
3	84	130	**255**	130	84

A call to countWhitePixels method would return 5 because there are 5 entries (shown in boldface) that have the value WHITE.

Complete method countWhitePixels below.

```
/** @return the total number of white pixels in this image.
 *  Postcondition: this image has not been changed.
 */
public int countWhitePixels()
```

b. Write the method processImage that modifies the image by changing the values in the instance variable pixelValues according to the following description. The pixels in the image are processed one at a time in row-major order. Row-major order processes the first row in the array from left to right and then processes the second row from left to right, continuing until all rows are processed from left to right. The first index of pixelValues represents the row number, and the second index represents the column number.

The pixel value at position (row, col) is decreased by the value at position (row + 2, col + 2) if such a position exists. If the result of the subtraction is less than the value BLACK, the pixel is assigned the value of BLACK. The values of the pixels for which there is no pixel at position (row + 2, col + 2) remain unchanged. You may assume that all the original values in the array are within the range [BLACK, WHITE], inclusive.

The following diagram shows the contents of the instance variable pixelValues before and after a call to processImage. The values shown in boldface represent the pixels that could be modified in a grayscale image with 4 rows and 5 columns.

Before Call to `processImage`

	0	1	2	3	4
0	**221**	**184**	**178**	84	135
1	**84**	**255**	**255**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

After Call to `processImage`

	0	1	2	3	4
0	**221**	**184**	**100**	84	135
1	**0**	**125**	**171**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

Information repeated from the beginning of the question

```
public class GrayImage

public static final int BLACK = 0
public static final int WHITE = 255
private int[][] pixelValues
public int countWhitePixels()
public void processImage()
```

Complete method processImage below.

```
/** Processes this image in row-major order and decreases the value of each pixel at
 *  position (row, col) by the value of the pixel at position (row + 2, col + 2) if it exists.
 *  Resulting values that would be less than BLACK are replaced by BLACK.
 *  Pixels for which there is no pixel at position (row + 2, col + 2) are unchanged.
 */
public void processImage()
```

24. Consider the following code segment.

```
int a = 5;
int b = 2;
double c = 3.0;
System.out.println(5 + a / b * c - 1);
```

What is printed when the code segment is executed?

(A) `0.666666666666667`

(B) `9.0`

(C) `10.0`

(D) `11.5`

(E) `14.0`

Unit 1 JLC 9_5_2023

25. This question involves reasoning about a simulation of a frog hopping in a straight line. The frog attempts to hop to a goal within a specified number of hops. The simulation is encapsulated in the following FrogSimulation class. You will write two of the methods in this class.

```
public class FrogSimulation
{
   /** Distance, in inches, from the starting position to the goal. */
   private int goalDistance;

   /** Maximum number of hops allowed to reach the goal. */
   private int maxHops;


   /** Constructs a FrogSimulation where dist is the distance, in inches, from the starting
    *  position to the goal, and numHops is the maximum number of hops allowed to reach the goal.
    *  Precondition: dist > 0; numHops > 0
    */
   public FrogSimulation(int dist, int numHops)
   {
      goalDistance = dist;
      maxHops = numHops;
   }


   /** Returns an integer representing the distance, in inches, to be moved when the frog hops.
    */
   private int hopDistance()
   {  /* implementation not shown */  }


   /** Simulates a frog attempting to reach the goal as described in part (a).
    *  Returns true if the frog successfully reached or passed the goal during the simulation;
    *          false otherwise.
    */
   public boolean simulate()
   {  /* to be implemented in part (a) */  }


   /** Runs num simulations and returns the proportion of simulations in which the frog
    *  successfully reached or passed the goal.
    *  Precondition: num > 0
    */
   public double runSimulations(int num)
   {  /* to be implemented in part (b) */  }
}
```

a. Write the simulate method, which simulates the frog attempting to hop in a straight line to a goal from the frog's starting position of 0 within a maximum number of hops. The method returns true if the frog successfully reached the goal within the maximum number of hops; otherwise, the method returns false.

   The FrogSimulation class provides a method called hopDistance that returns an integer representing the distance (positive or negative) to be moved when the frog hops. A positive distance represents a move toward the goal. A negative distance represents a move away from the goal. The returned distance may vary from call to call. Each time the frog hops, its position is adjusted by the value returned by a call to the hopDistance method.

   The frog hops until one of the following conditions becomes true:

   • The frog has reached or passed the goal.

   • The frog has reached a negative position.

   • The frog has taken the maximum number of hops without reaching the goal.

The following example shows a declaration of a FrogSimulation object for which the goal distance is 24 inches and the maximum number of hops is 5. The table shows some possible outcomes of calling the simulate method.

FrogSimulation sim = new FrogSimulation(24, 5);

	Values returned by `hopDistance()`	Final position of frog	Return value of `sim.simulate()`
Example 1	`5, 7, -2, 8, 6`	24	`true`
Example 2	`6, 7, 6, 6`	25	`true`
Example 3	`6, -6, 31`	31	`true`
Example 4	`4, 2, -8`	-2	`false`
Example 5	`5, 4, 2, 4, 3`	18	`false`

Class information for this question

```
public class FrogSimulation

private int goalDistance
private int maxHops

private int hopDistance()
public boolean simulate()
public double runSimulations(int num)
```

Complete method simulate below. You must use hopDistance appropriately to receive full credit.

/** Simulates a frog attempting to reach the goal as described in part (a). * Returns true if the frog successfully reached or passed the goal during the simulation; * false otherwise. */

public boolean simulate()

b. Write the runSimulations method, which performs a given number of simulations and returns the proportion of simulations in which the frog successfully reached or passed the goal. For example, if the parameter passed to runSimulations is 400, and 100 of the 400 simulate method calls returned true, then the runSimulations method should return 0.25.

Unit 1 JLC 9_5_2023

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

```
/** Runs num simulations and returns the proportion of simulations in which the frog
 *  successfully reached or passed the goal.
 *  Precondition: num > 0
 */
public double runSimulations(int num)
```

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
```

Unit 1 JLC 9_5_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.
}
```

26. **(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()
```

27. Assume that `x` and `y` are variables of type `int`. Which of the following Java expressions never results in a division by zero?

(A) `(y / x) == 0`

(B) `((y / x) == 0) && (x != 0)`

(C) `((y / x) == 0) || (x != 0)`

(D) `(x != 0) && ((y / x) == 0)`

(E) `(x != 0) || ((y / x) == 0)`

28. 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 mathematical sequence is an ordered list of numbers. This question involves a sequence called a *hailstone sequence*. If n is the value of a term in the sequence, then the following rules are used to find the next term, if one exists.

- If n is 1, the sequence terminates.
- If n is even, then the next term is $\frac{n}{2}$.
- If n is odd, then the next term is $3n + 1$.

For this question, assume that when the rules are applied, the sequence will eventually terminate with the term $n = 1$.

The following are examples of hailstone sequences.

Example 1: 5, 16, 8, 4, 2, 1

- The first term is 5, so the second term is $5 * 3 + 1 = 16$.
- The second term is 16, so the third term is $\frac{16}{2} = 8$.
- The third term is 8, so the fourth term is $\frac{8}{2} = 4$.
- The fourth term is 4, so the fifth term is $\frac{4}{2} = 2$.
- The fifth term is 2, so the sixth term is $\frac{2}{2} = 1$.
- Since the sixth term is 1, the sequence terminates.

Example 2: 8, 4, 2, 1

- The first term is 8, so the second term is $\frac{8}{2} = 4$.
- The second term is 4, so the third term is $\frac{4}{2} = 2$.
- The third term is 2, so the fourth term is $\frac{2}{2} = 1$.
- Since the fourth term is 1, the sequence terminates.

The `Hailstone` class, shown below, is used to represent a hailstone sequence. You will write three methods in the `Hailstone` class.

```
public class Hailstone
{
     /** Returns the length of a hailstone sequence that starts with n,
       * as described in part (a).
       * Precondition: n > 0
       */
     public static int hailstoneLength(int n)
```

```
    { /* to be implemented in part (a) */ }
  /** Returns true if the hailstone sequence that starts with n is
considered long
      * and false otherwise, as described in part (b).
      * Precondition: n > 0
      */
    public static boolean isLongSeq(int n)
    { /* to be implemented in part (b) */ }
  /** Returns the proportion of the first n hailstone sequences that are
considered long,
      * as described in part (c).
      * Precondition: n > 0
      */
    public static double propLong(int n)
    { /* to be implemented in part (c) */ }
  // There may be instance variables, constructors, and methods not
shown.
}
```

(a) The length of a hailstone sequence is the number of terms it contains. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) has a length of 6 and the hailstone sequence in example 2 (8, 4, 2, 1) has a length of 4.

Write the method `hailstoneLength(int n)`, which returns the length of the hailstone sequence that starts with `n`.

```
/** Returns the length of a hailstone sequence that starts with n,
 * as described in part (a).
 * Precondition: n > 0
 */
public static int hailstoneLength(int n)
```

Class information for this question

```
public class Hailstone
public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```

(b) A hailstone sequence is considered long if its length is greater than its starting value. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) is considered long because its length (6) is greater than its starting value (5). The hailstone sequence in example 2 (8, 4, 2, 1) is not considered long because its length (4) is less than or equal to its starting value (8).

Write the method `isLongSeq(int n)`, which returns `true` if the hailstone sequence starting with `n` is considered long and returns `false` otherwise. Assume that `hailstoneLength` works as intended, regardless of what you wrote in part (a). You must use `hailstoneLength` appropriately to receive full credit.

```
/** Returns true if the hailstone sequence that starts with n is
considered long
 * and false otherwise, as described in part (b).
 * Precondition: n > 0
 */
public static boolean isLongSeq(int n)
```

(c) The method `propLong(int n)` returns the proportion of long hailstone sequences with starting values between 1 and `n`, inclusive.

Consider the following table, which provides data about the hailstone sequences with starting values between 1 and 10, inclusive.

Starting Value	Terms in the Sequence	Length of the Sequence	Long?
1	1	1	No
2	2, 1	2	No
3	3, 10, 5, 16, 8, 4, 2, 1	8	Yes
4	4, 2, 1	3	No
5	5, 16, 8, 4, 2, 1	6	Yes
6	6, 3, 10, 5, 16, 8, 4, 2, 1	9	Yes
7	7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	17	Yes
8	8, 4, 2, 1	4	No
9	9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	20	Yes
10	10, 5, 16, 8, 4, 2, 1	7	No

The method call `Hailstone.propLong(10)` returns `0.5`, since 5 of the 10 hailstone sequences shown in the table are considered long.

Write the `propLong` method. Assume that `hailstoneLength` and `isLongSeq` work as intended, regardless of what you wrote in parts (a) and (b). You must use `isLongSeq` appropriately to receive full credit.

```
/** Returns the proportion of the first n hailstone sequences that are
considered long,
```

```
 * as described in part (c).
 * Precondition: n > 0
 */
public static double propLong(int n)
```

Class information for this question

```
public class Hailstone
public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```

29. Consider the following code segment.

```
System.out.print("Hello System.out.println");
System.out.print("!!!");
```

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

(A) `Hello!!!`

(B) `Hello System.out.println!!!`

(C)
```
Hello
!!!
```

(D)
```
Hello System.out.println
!!!
```

(E) Nothing is printed because the text `"System.out.println"` cannot appear inside a print statement.

30. 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 a system to manage hotel rooms available for occupancy. The rooms are represented by the following `HotelRoom` class.

```
public class HotelRoom
{
 /** Returns the number of beds in the room */
 public int getNumBeds()
 {  /* implementation not shown */  }

 /** Returns the room number of the room */
 public int getRoomNumber()
 {  /* implementation not shown */  }

 /** Returns true if the room is a suite and returns false otherwise */
 public boolean isSuite()
 {  /* implementation not shown */  }

 /** Returns true if the room is available for occupancy and returns
 false otherwise
  */
 public boolean isAvailable()
 {  /* implementation not shown */  }

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

Information about a hotel is stored in a `Hotel` object, which contains an array of rooms at the hotel. You will write two methods of the `Hotel` class.

```
public class Hotel
{
```

```
    /** An array of all rooms at the hotel
      *  Guaranteed not to be null and to contain only non-null entries
     */
    private HotelRoom[] rooms;

    /** Returns the room number of an available room at the hotel, as
    described in part (a) */
    public int chooseRoom(int numBeds, boolean wantSuite)
    {  /* to be implemented in part (a) */  }

    /** Returns the number of staff members needed to clean hotel rooms, as
    described in
     *  part (b)
     */
    public int numberOfStaff()
    {  /* to be implemented in part (b) */  }

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

(a) Write the `Hotel` method `chooseRoom`. The method returns the room number for a room in the hotel that meets both of the following criteria.

- The room has at least `numBeds` beds and is available for occupancy.
- The room type (whether or not the room is a suite) matches the value specified by the `wantSuite` parameter.

If a room that meets the criteria is found, its room number is returned. Otherwise, the method returns `-1`. If more than one room meets the criteria, the room number of any matching room may be returned.

Complete method `chooseRoom`.

```
/** Returns the room number of an available room at the hotel, as
described in part (a) */
public int chooseRoom(int numBeds, boolean wantSuite)
```

(b) Write the `Hotel` method `numberOfStaff`. The method returns the number of staff members required to clean all the occupied (not available) rooms in the hotel. The number of staff members is calculated as follows.

- Two staff members are needed to clean each occupied suite.
- One staff member is needed to clean each group of three occupied non-suite rooms. One additional staff member is required if there are one or two occupied non-suite rooms not included in a group of three.

The following table shows some examples of the calculation of staff required to clean occupied rooms.

Number of Suites	Number of Non-Suites	numberOfStaff Return Value	Explanation
3	3	7	The hotel needs 6 staff members for the suites and 1 staff member for the non-suites.
1	4	4	The hotel needs 2 staff members for the suite and 2 staff members for the non-suites (1 staff member for the first 3 non-suites and 1 staff member for the remaining non-suite).
0	8	3	The hotel needs no staff members for the suites and 3 staff members for the non-suites (2 staff members for the first 6 non-suites and 1 staff member for the remaining 2 non-suites).
5	0	10	The hotel needs 10 staff members for the suites and no staff members for the non-suites.

Complete method `numberOfStaff`.

```
/** Returns the number of staff members needed to clean hotel rooms, as
described in
 *  part (b)
 */
public int numberOfStaff()
```

(c) A programmer would like to add a method called `getAllAffordableAvailableRooms`, which returns a list of all rooms that are available for occupancy and cost less than a given amount.

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

Make sure to include the following in your response.

- Write the method header for the `getAllAffordableAvailableRooms` method.
- Identify any new or modified variables, constructors, or methods aside from the `getAllAffordableAvailableRooms` 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 `getAllAffordableAvailableRooms` method. **Do not write the program code for this change.**

31. 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?

(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);`

32. Consider the following code segment.

```
double firstDouble = 2.5;
int firstInt = 30;
int secondInt = 5;
double secondDouble = firstInt - secondInt / firstDouble + 2.5;
```

What value will be assigned to `secondDouble` when the code segment is executed?

(A) `5.0`

(B) `12.5`

(C) `25.5`

(D) `29.0`

(E) `30.5`

33. The following code segment is intended to interchange the values of the `int` variables `x` and `y`. Assume that `x` and `y` have been properly declared and initialized.

```
int temp = x;
/* missing code */
```

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

(A)
```
x = y;
x = temp;
```

(B)
```
x = y;
y = temp;
```

(C)
```
y = x;
x = temp;
```

(D)
```
y = x;
temp = y;
```

(E)
```
y = x;
temp = x;
```

34. Consider the following method.

```
public static int mystery(boolean a, boolean b, boolean c)
{
     int answer = 7;

     if (!a)
     {
         answer += 1;
     }

     if (b)
     {
         answer += 2;
     }

     if (c)
     {
         answer += 4;
     }

     return answer;
}
```

Which of the following method calls will return the value `11` ?

(A) `mystery(true, true, true)`

(B) `mystery(true, false, true)`

(C) `mystery(false, true, false)`

(D) `mystery(false, false, true)`

(E) `mystery(false, false, false)`

35. Consider the following code segment.

```
num += num;
num *= num;
```

Assume that `num` has been previously declared and initialized to contain an integer value. Which of the following best describes the behavior of the code segment?

(A) The value of `num` is two times its original value.

(B) The value of `num` is the square its original value.

(C) The value of `num` is two times the square of its original value.

(D) The value of `num` is the square of twice its original value.

(E) It cannot be determined without knowing the initial value of `num`.

36. Consider the following code segment, which is intended to print the digits of the two-digit `int` number `num` in reverse order. For example, if `num` has the value `75`, the code segment should print `57`. Assume that `num` has been properly declared and initialized.

```
/* missing code */
System.out.print(onesDigit);
System.out.print(tensDigit);
```

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

(A)
```
int onesDigit = num % 10;
int tensDigit = num / 10;
```

(B)
```
int onesDigit = num / 10;
int tensDigit = num % 10;
```

(C)
```
int onesDigit = 10 / num;
int tensDigit = 10 % num;
```

(D)
```
int onesDigit = num % 100;
int tensDigit = num / 100;
```

(E)
```
int onesDigit = num / 100;
int tensDigit = num % 100;
```

37. Consider the following code segment.

```
int a = 3 + 2 * 3;
int b = 4 + 3 / 2;
int c = 7 % 4 + 3;
double d = a + b + c;
```

What is the value of `d` after the code segment is executed?

(A) `14.0`

(B) `18.0`

(C) `20.0`

(D) `20.5`

(E) `26.0`

38. Which of the following expressions evaluate to `7` ?

I. `9 + 10 % 12`
II. `(9 + 10) % 12`
III. `9 - 2 % 12`

(A) I only

(B) II only

(C) I and III

(D) II and III

(E) I, II, and III

39. Consider the following code segment.

```
int x = 5;
x += 6 * 2;
x -= 3 / 2;
```

What value is stored in `x` after the code segment executes?

(A) `-1.5`

(B) `1`

(C) `9`

(D) `15.5`

(E) `16`

40. Consider the following code segment, where `k` and `count` are properly declared and initialized `int` variables.

```
k++;
k++;
count++;
k--;
count++;
k--;
```

Which of the following best describes the behavior of the code segment?

(A) The code segment leaves both `k` and `count` unchanged.

(B) The code segment increases both `k` and `count` by `2`.

(C) The code segment increases `k` by `4` and `count` by `2`.

(D) The code segment leaves `k` unchanged and increases `count` by `2`.

(E) The code segment increases `k` by `2` and leaves `count` unchanged.

41. Consider the following code segment.

```
int a = 4;
int b = 5;
a++;
b++;
int c = a + b;
a -= 1;
System.out.println(a + c);
```

What is printed when the code segment is executed?

(A) `9`

(B) `10`

(C) `14`

(D) `15`

(E) `25`

42. Consider the following code segment.

```
System.out.print("AP");
System.out.println();
System.out.println("CS");
System.out.print("A");
```

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

(A)
```
APCSA
```

(B)
```
APCS
A
```

(C)
```
AP
CSA
```

(D)
```
AP
CS
A
```

(E)
```
AP

CS
A
```

43. Consider the following code segment.

```
System.out.print(I do not fear computers. );   // Line 1
System.out.println(I fear the lack of them.);  // Line 2
System.out.println(--Isaac Asimov);            // Line 3
```

The code segment is intended to produce the following output but may not work as intended.

```
I do not fear computers. I fear the lack of them.
--Isaac Asimov
```

Which change, if any, can be made so that the code segment produces the intended output?

(A) In line 1, `print` should be changed to `println`.

(B) In lines 2 and 3, `println` should be changed to `print`.

(C) The statement `System.out.println()` should be inserted between lines 2 and 3.

(D) In lines 1, 2, and 3, the text that appears in parentheses should be enclosed in quotation marks.

(E) No change is needed; the code segment works correctly as is.

44. Consider the following code segment.

```
System.out.print(*);     // Line 1
System.out.print("*");   // Line 2
System.out.println();    // Line 3
System.out.println("*"); // Line 4
```

The code segment is intended to produce the following output, but may not work as intended.

```
**
*
```

Which line of code, if any, causes an error?

(A) Line 1
(B) Line 2
(C) Line 3
(D) Line 4
(E) The code segment works as intended.

45. Consider the following code segment.

```
System.out.print("*");
System.out.println("**");
System.out.println("***");
System.out.print("****");
```

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

(A)
```
*
**
***
****
```

(B)
```
*
**
*******
```

(C)
```
*
*****
****
```

(D)
```
***
***
****
```

(E)
```
***
*******
```

46. Consider the following code segment.

```
System.out.print("One");   // Line 1
System.out.print("Two");   // Line 2
System.out.print("Three"); // Line 3
System.out.print("Four");  // Line 4
```

The code segment is intended to produce the following output, but does not work as intended.

```
OneTwo
ThreeFour
```

Which of the following changes can be made so that the code segment produces the intended output?

(A) Changing `print` to `println` in line 1 only

(B) Changing `print` to `println` in line 2 only

(C) Changing `print` to `println` in line 3 only

(D) Changing `print` to `println` in lines 2 and 3 only

(E) Changing `print` to `println` in lines 1, 2, 3, and 4

47. What is printed as a result of executing the following statement?

```
System.out.println(404 / 10 * 10 + 1);
```

(A) 4

(B) 5

(C) 41

(D) 401

(E) 405

48. Consider the following code segment.

```
for (int k = 0; k < 9; k = k + 2)
{
     if ((k % 2) != 0)
     {
          System.out.print(k + " ");
     }
}
```

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

(A) `0 2 4 6 8 10`

(B) `0 2 4 6 8`

(C) `1 3 5 7 9`

(D) `1 3 5 7`

(E) Nothing is printed.

49. Consider the following code segment.

```
double a = 1.1;

double b = 1.2;


if ((a + b) * (a - b) != (a * a) - (b * b))

{

  System.out.println("Mathematical error!");

}
```

Which of the following best describes why the phrase "Mathematical error!" would be printed?

(Remember that mathematically $(a + b) * (a - b) = a^2 - b^2$.)

(A) Precedence rules make the if condition true.

(B) Associativity rules make the if condition true.

(C) Roundoff error makes the if condition true.

(D) Overflow makes the if condition true.

(E) A compiler bug or hardware error has occurred.

50. 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.

51. 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.

52. The following code segment is intended to round `val` to the nearest integer and print the result.

```
double val = -0.7;
int roundedVal = (int) (val + 0.5);
System.out.println(roundedVal);
```

Which of the following best describes the behavior of the code segment?

Unit 1 JLC 9_5_2023

(A) The code segment works as intended.

(B) The code segment does not work as intended because `val` and `roundedVal` should be declared as the same data type.

(C) The code segment does not work as intended because the expression `(val + 0.5)` should be cast to a `double` instead of an `int`.

(D) The code segment does not work as intended because `val` should be cast to an `int` before `0.5` is added to it.

(E) The code segment does not work as intended because the expression `(int) (val + 0.5)` rounds to the nearest integer only when `val` is positive.

53. 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 manufacturer wants to keep track of the average of the ratings that have been submitted for an item using a running average. The algorithm for calculating a running average differs from the standard algorithm for calculating an average, as described in part (a).

A partial declaration of the `RunningAverage` class is shown below. You will write two methods of the `RunningAverage` class.

```
public class RunningAverage
{
     /** The number of ratings included in the running average. */
     private int count;

     /** The average of the ratings that have been entered. */
     private double average;

     // There are no other instance variables.

     /** Creates a RunningAverage object.
       * Postcondition: count is initialized to 0 and average is
       * initialized to 0.0.
       */
     public RunningAverage()
     { /* implementation not shown */ }

     /** Updates the running average to reflect the entry of a new
       * rating, as described in part (a).
       */
     public void updateAverage(double newVal)
     { /* to be implemented in part (a) */ }

     /** Processes num new ratings by considering them for inclusion
       * in the running average and updating the running average as
       * necessary. Returns an integer that represents the number of
       * invalid ratings, as described in part (b).
       * Precondition: num > 0
       */
```

```
    public int processNewRatings(int num)
    { /* to be implemented in part (b) */ }

    /** Returns a single numeric rating. */
    public double getNewRating()
    { /* implementation not shown */ }
}
```

(a) Write the method `updateAverage`, which updates the `RunningAverage` object to include a new rating. To update a running average, add the new rating to a calculated total, which is the number of ratings times the current running average. Divide the new total by the incremented count to obtain the new running average.

For example, if there are 4 ratings with a current running average of 3.5, the calculated total is 4 times 3.5, or 14.0. When a fifth rating with a value of 6.0 is included, the new total becomes 20.0. The new running average is 20.0 divided by 5, or 4.0.

Complete method `updateAverage`.

```
/** Updates the running average to reflect the entry of a new
 * rating, as described in part (a).
 */
public void updateAverage(double newVal)
```

(b) Write the `processNewRatings` method, which considers `num` new ratings for inclusion in the running average. A helper method, `getNewRating`, which returns a single rating, has been provided for you.

The running average must only be updated with ratings that are greater than or equal to zero. Ratings that are less than 0 are considered invalid and are not included in the running average.

The `processNewRatings` method returns the number of invalid ratings. See the table below for three examples of how calls to `processNewRatings` should work.

Unit 1 JLC 9_5_2023

Statement	Ratings Generated	processNewRatings Return Value	Comments
`processNewRatings(2)`	`2.5, 4.5`	`0`	Both new ratings are included in the running average.
`processNewRatings(1)`	`-2.0`	`1`	No new ratings are included in the running average.
`processNewRatings(4)`	`0.0, -2.2, 3.5, -1.5`	`2`	Two new ratings (`0.0` and `3.5`) are included in the running average.

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

```
/** Processes num new ratings by considering them for inclusion
 * in the running average and updating the running average as
 * necessary. Returns an integer that represents the number of
 * invalid ratings, as described in part (b).
 * Precondition: num > 0
 */
public int processNewRatings(int num)
```

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.

3. This question involves a system to manage a scientist's experimental trials in a lab. Trials are represented by the following `Trial` class.

```
public class Trial
{
 /** Returns the number of the trial */
 public int getTrialNum()
 {  /* implementation not shown */  }

 /** Returns the temperature reading of this trial
   *   Postcondition: The value returned is greater than or equal to
 0.0.
  */
 public double getTemp()
 {  /* implementation not shown */  }

 /** Returns the compound (for example, "alkynes", "ketones", or
 "ethers")
  *   used in this trial
  */
 public String getCompound()
 {  /* implementation not shown */  }

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

A scientist's trials are stored in a `ScienceExperiment` object, which contains a list of the scientist's trials. You will write two methods in this class.

```
public class ScienceExperiment
{
```

```
/** A list containing the scientist's trials
  *  Guaranteed not to be null and to contain only non-null entries
 */
private ArrayList<Trial> trialList;

/** Returns the average of all trial temperatures for a given compound
if there are any;
 *  otherwise, returns -1.0
 */
public double getCompoundAvg(String comp)
{  /* to be implemented in part (a) */  }

/** Returns a list of compounds used in the trials
  *  Postcondition: There are no duplicates in the returned list.
 */
public ArrayList<String> getCompoundList()
{  /* implementation not shown */  }

/** Returns a compound with the highest average temperature, as
described in part (b)
  *  Precondition: There is at least one entry in the list of trials.
 */
public String getCompoundWithHighestAvg()
{  /* to be implemented in part (b) */  }

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

(a) Write the `ScienceExperiment` method `getCompoundAvg`. The method returns the average (arithmetic mean) temperature of trials that use a given compound. If there are no trials that use the given compound, the method returns `-1.0`.

For example, assume that `experiment1` has been declared as a `ScienceExperiment` object and contains `Trial` objects with the following return values from the `getTrialNum`, `getTemp`, and `getCompound` methods.

`getTrialNum()` **Return Value**	`getTemp()` **Return Value**	`getCompound()` **Return Value**
`1`	`70.0`	`"alkynes"`
`27`	`100.0`	`"ketones"`
`29`	`80.0`	`"ketones"`

The following table shows the average temperature that would be computed for each compound.

Method Call	Return Value
`experiment1.getCompoundAvg("alkynes")`	`70.0`
`experiment1.getCompoundAvg("ketones")`	`90.0`
`experiment1.getCompoundAvg("thiol")`	`-1.0`

Complete method `getCompoundAvg`.

```
/** Returns the average of all trial temperatures for a given compound if
there are any;
 *  otherwise, returns -1.0
 */
public double getCompoundAvg(String comp)
```

(b) Write the `ScienceExperiment` method `getCompoundWithHighestAvg`. The method returns a compound that has the highest average temperature. If there is more than one compound with the highest average temperature, any one of those compounds may be returned.

For example, assume that `experiment1` has been declared as a `ScienceExperiment` object and contains `Trial` objects with the following return values from the `getTrialNum`, `getTemp`, and `getCompound` methods.

`getTrialNum()` **Return Value**	`getTemp()` **Return Value**	`getCompound()` **Return Value**
`1`	`70.0`	`"alkynes"`
`27`	`100.0`	`"ketones"`
`35`	`90.0`	`"ethers"`
`29`	`80.0`	`"ketones"`

The following table shows the average temperature for each compound.

Method Call	Return Value
`experiment1.getCompoundAvg("alkynes")`	`70.0`
`experiment1.getCompoundAvg("ketones")`	`90.0`
`experiment1.getCompoundAvg("ethers")`	`90.0`

The call `experiment1.getCompoundWithHighestAvg()` would return either `"ketones"` or `"ethers"` because each has the highest average temperature of `90.0`.

A helper method, `getCompoundList`, has been provided. The `getCompoundList` method returns an `ArrayList` of `String` objects representing the compounds that are used in the trials.

Assume that `getCompoundAvg` works as specified, regardless of what you wrote for part (a). You must use `getCompoundAvg` and `getCompoundList` appropriately to receive full credit.

Complete method `getCompoundWithHighestAvg`.

```
/** Returns one of the compounds with the highest average temperature, as
described in part (b)
  *  Precondition: There is at least one entry in the list of trials.
 */
public String getCompoundWithHighestAvg()
```

(c) A programmer would like to add a method called `getCompoundAmountUsed`, which returns a `double` that represents the total amount of a given compound used in all trials in the experiment.

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

Make sure to include the following in your response.

- Write the method header for the `getCompoundAmountUsed` method.
- Identify any new or modified variables, constructors, or methods aside from the `getCompoundAmountUsed` 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 `getCompoundAmountUsed` method. **Do not write the program code for this change.**

55. Which of the following statements stores the value `3` in `x` ?

(A) `int x = 4 / 7;`

(B) `int x = 7 / 3;`

(C) `int x = 7 / 4;`

(D) `int x = 5 % 8;`

(E) `int x = 8 % 5;`

Directions: Select the choice that best fits each statement. The following question(s) refer to the following incomplete class declaration.

```
public class TimeRecord
{
  private int hours;
  private int minutes; // 0 ≤ minutes < 60
  /** Constructs a TimeRecord object.
   *  @param h the number of hours
   *         Precondition: h ≥ 0
   *  @param m the number of minutes
   *         Precondition: 0 ≤ m < 60
   */
  public TimeRecord(int h, int m)
  {
    hours = h;
    minutes = m;
  }

  /** @return the number of hours
   */
  public int getHours()
  { /* implementation not shown */ }

  /** @return the number of minutes
   *  Postcondition: 0 ≤ minutes < 60
   */
  public int getMinutes()
  { /* implementation not shown */ }

  /** Adds h hours and m minutes to this TimeRecord.
   *  @param h the number of hours
   *         Precondition: h ≥ 0
   *  @param m the number of minutes
   *         Precondition: m ≥ 0
   */
  public void advance(int h, int m)
  {
    hours = hours + h;
    minutes = minutes + m;
    /* missing code */
  }
  // Other methods not shown
}
```

56. Which of the following can be used to replace / * *missing code* * / so that advance will correctly update the time?

(A) minutes = minutes % 60;

(B) minutes = minutes + hours % 60;

(C) hours = hours + minutes / 60;
minutes = minutes % 60;

(D) hours = hours + minutes % 60;
minutes = minutes / 60;

(E) hours = hours + minutes / 60;

57. 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.