Spec Sheet for Chemistry Calculator Program

Purpose

The purpose of this program is to calculate the molar mass of a given chemical compound based on its chemical formula. This utility will help students, chemists, and educational professionals quickly determine the molar mass necessary for various chemical calculations and experiments.

Scope

- Target Users: Students, educators, and professionals in chemistry.
- Platform: Desktop and command-line interface.
- Language: Java.

Requirements

- Functional Requirements:
  - The program must accept a chemical formula as input from the user.
  - The program must calculate the molar mass by parsing the input formula and summing the atomic masses of the constituent elements.
  - The program must handle errors such as unrecognized elements and display appropriate messages.
  - The program should ignore case sensitivity for element symbols.

- Non-Functional Requirements:
  - Performance: The program should perform calculations within seconds.
  - Usability: The program should be easy to use, with clear prompts and error messages.
  - Maintainability: The code should be well-documented and easy to extend with additional elements or functionalities.

Design Constraints

- The program will only handle non-parenthetical chemical formulas (e.g., no handling of nested elements like Ca(OH)2).
- The program will run in a command-line interface without a graphical user interface.

System Architecture

- Input/Output: The program will use standard input and output for interaction.
- Data Management: Element weights will be stored in a static HashMap initialized at runtime.
- Error Handling: The program will catch and handle exceptions related to input errors, such as invalid formulas or non-existent elements.

User Interface

- Input: User is prompted to enter a chemical formula.
- Output: Displays the calculated molar mass or an error message.

Technologies Used

- Core Technology: Java.
- Libraries: java.util for HashMap and Scanner, java.util.regex for pattern matching.

Development and Deployment Environment

- IDE: Eclipse, or any Java IDE.
- Java Version: Java SE 8 or higher.
- Testing: Unit tests using JUnit to cover various chemical formulas and edge cases.
- Deployment: No need at this time.

Testing Strategy

- Test cases will be written to ensure that all elements are correctly recognized and calculated.
- Boundary tests for chemical formulas with maximum and minimum valid lengths.
- Error handling tests to ensure graceful handling of unsupported or incorrectly entered formulas.

Maintenance and Support

- The program will be maintained by the initial development team with periodic updates for additional elements and improvements in parsing algorithms.
- User feedback will be collected for future enhancements.