PHM Technology

Decisions better made

PHM Technology (PHMT) specializes in developing advanced engineering analysis software to optimize the design and sustainment of complex engineering systems in mission-critical and safety-critical industries. Our flagship product, the Maintenance Aware Design Ecosystem (MADE), is at the forefront of technological innovation, offering comprehensive solutions that enhance reliability, reduce downtime, and improve overall operational efficiency.

We are expanding our development team in Melbourne to ensure MADE remains a cutting-edge solution that meets the evolving needs of our clients. By joining us, you will collaborate with leading engineers and industry experts, working on challenging projects that have a tangible impact on the future of engineering. You'll be part of a dynamic team that serves global clients in the aerospace, automotive, and defense sectors, contributing to advancements in these high-stakes industries.

Our development team is dedicated to creating innovative modeling concepts, sophisticated mathematical simulations, and decision support tools, all seamlessly integrated with ERP and PLM systems. We foster a collaborative environment that encourages creativity, continuous learning, and the sharing of ideas.

As part of the shortlisting application process for a development role at PHM Technology, candidates are required to complete this Applicant Test. This assessment is designed to evaluate your technical proficiency and problem-solving abilities, ensuring that you are well-equipped to contribute to our mission of delivering top-tier engineering solutions.

1

Core Java

Develop the required Java class(es) to represent the following data within an application.

Note: You may assume that classes related to components already exist. Please focus exclusively on the Maintenance Action.

Users need the ability to create Maintenance Activities, which represent different types of maintenance that may be performed on a component. Each Maintenance Action should record the following information:

1. **Action ID:** A unique identifier for the Maintenance Action.
2. **Name:** A descriptive name for the Maintenance Action.
3. **Maintenance Type:** The type of maintenance being performed. This can be one of the following:
    - Repair
    - Replace
    - Service
    - Lubrication
4. **Mean Time Between Maintenance (MTBM):** The average time in hours between executions of the Maintenance Action.
5. **Maintenance Action Tasks:** One or more Tasks representing the steps required to complete the Maintenance Action.

Each Maintenance Task must include the following variables:

1. **Task ID:** A unique identifier for the task within the Maintenance Action.
2. **Name:** A descriptive name for the task.
3. **Duration:** The expected time required to complete the task.
4. **Personnel Type:** The category of personnel needed to perform the task.
5. **Hourly Rate:** The hourly rate charged for the personnel.
6. **Number of Personnel:** The average number of personnel required for the task.

Please ensure your classes are well-structured and adhere to best coding practices.

2

File IO

1. Develop a Java Program to Read and Display Text from a File

Create a Java application that reads the contents of a text file and displays them in the console output. The text file shall contain the following line:

```
A SQL query goes into a bar, walks up to two tables and asks,
Can I join you?
```

Requirements:

- **File Reading:** Implement file input operations to read the text file.
- **Error Handling:** Include exception handling to manage potential issues such as the file not being found or access permissions being denied.
- **Resource Management:** Ensure all file streams are properly closed after operations to prevent resource leaks.
- **Output:** Display the exact contents of the file in the console.

2. Develop a Java GUI Application to Save User Input to a File

Design a Java application with a graphical user interface (GUI) that allows users to enter text and save it to a text file on their disk.

Features:

- **User Input Area:** Provide a text area where users can type their content.
- **Save Functionality:** Include a "Save" button that, when clicked, opens a file chooser dialog. This enables the user to select the destination and filename for saving the text file.
- **User Feedback:** After attempting to save the file, display a message indicating whether the operation was successful or if an error occurred.
- **Error Handling:** Implement exception handling for potential issues like IOExceptions to ensure the application remains stable during errors.

Requirements:

- **GUI Components:** Utilize standard Java libraries for GUI development (e.g., Swing, SWT or JavaFX) to create an intuitive and user-friendly interface.
- **File Writing:** Implement file output operations to write the user's input to the selected file.
- **Resource Management:** Ensure all file streams are properly closed after writing.
- **Validation:** Validate user input and handle cases where the user cancels the file save operation.

Additional Notes:

- **Best Practices:** Follow coding standards and best practices, including code readability, commenting, and modular design.
- **Testing:** Thoroughly test the application to handle various scenarios, such as empty input, canceled file dialogs, and file permission issues.
- **User Experience:** Ensure that error messages are clear and provide guidance on how to resolve issues.

3

Tree Data Structure

1. Develop a Java Program to Search a Tree for a Corresponding Character

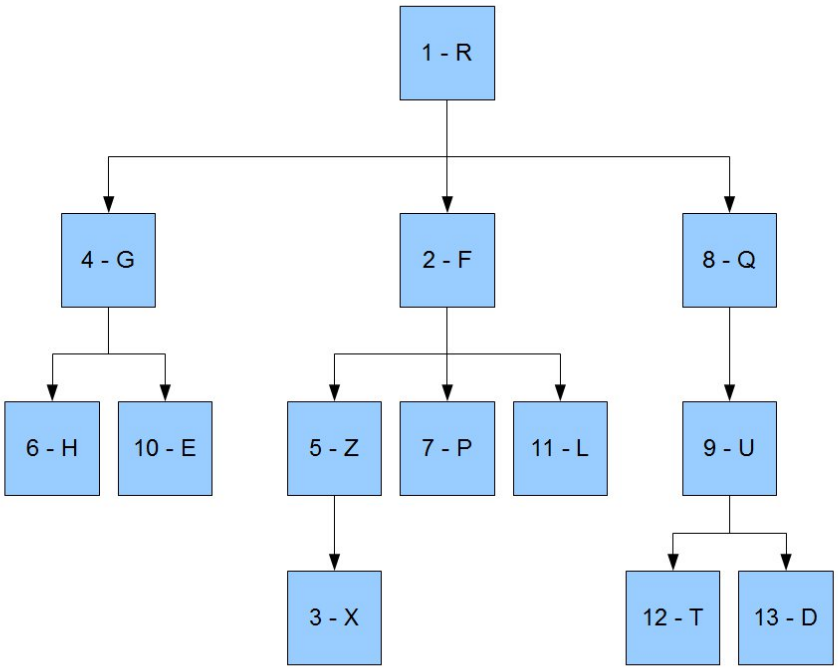
Create a Java program that constructs a tree data structure as shown in the provided diagram.

Each **node** of the tree contains:

- An **integer**
- A **character variable**

Requirements:

- **Tree Construction:**
  - Build the tree according to the diagram.
  - Ensure each leaf node stores both an integer and a corresponding character.
- **Recursive Search:**
  - Implement a recursive algorithm that accepts an integer input from the user.
  - Search the tree to find the leaf node with the matching integer.
  - Output the corresponding character to the console.
- **Error Handling:**
  - If the integer is not found in any leaf node, display an appropriate message to the user.



4

Math & Problem Solving

1. Complete the Equation with Given Inputs

Given the partial equation below and the input values **1, 2, 3, and 4**, fill in the missing section (indicated by "..."):

```
                          (1)  +  (2)  +  (3)  +  (4)  -
(1 x 2)  -  (1 x 3)  -  (1 x 4)  -  (2 x 3)  -  (2 x 4)  -  (3 x 4)  +
( ..................................................................... )  -
                              ( 1 x 2 x 3 x 4 )
```

Task: Determine the missing part of the equation based on the pattern shown.

2. Develop a Java Program to Calculate the Result

Create a Java program that performs the following:

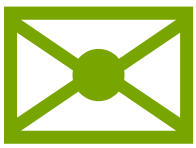
Requirements:

- **Input Handling:**
  - Accepts a collection of double values from the user.
  - Supports a maximum of **100,000** input elements.
- **Equation Formulation:**
  - Constructs an equation based on the input values, following the pattern demonstrated in the example.
  - The equation should systematically combine the input values using addition and subtraction of their products.
- **Calculation:**
  - Computes the result of the formulated equation.
  - Ensures accurate calculations even with a large number of inputs.
- **Output:**
  - Displays the calculated result in the console.
  - Writes the fully expanded equation to a log file for validation purposes.
- **Performance Optimization:**
  - Manages memory efficiently to prevent out-of-memory errors during execution.
  - Optimizes the algorithm to handle large datasets within reasonable time constraints.

Note: The equation utilizes concepts from set theory, specifically involving the inclusion-exclusion principle applied to the products of the input values.

Example:

- **Input:** 2.0, 4.0, 6.0
- **Equation:** (2.0) + (4.0) + (6.0) - (2.0 × 4.0) - (2.0 × 6.0) - (4.0 × 6.0) + (2.0 × 4.0 × 6.0)
- **Calculated Result:** 16.0



Submit Applicant Test

How to Submit Your Applicant Test:

1. **Prepare Your Files:**
   - Ensure all your Java source files and any accompanying documents required for the test are complete and properly organized.
   - Verify that your code is well-documented and follows best coding practices.

1. **Create a ZIP Archive:**
   - Compress all the necessary files into a single ZIP file.
   - Recommended naming convention: YourName_PHMT_ApplicantTest.zip

1. **Submit via Email:**
   - Send the ZIP archive as an email attachment.
   - Address the email to: **jobs@phmtechnology.com**
   - Use the subject line: PHMT Applicant Test Submission - [Your Name]