Basic Questions

Part 1: Data Types, Variables, and Arithmetic Operations

1. Declare variables of different data types (int, double, boolean, char) and perform various arithmetic operations on them.
2. Write a program that calculates the area and perimeter of a rectangle, given the length and width as input.
3. Implement a program that converts temperature from Celsius to Fahrenheit and vice versa.

Part 2: Classes and Methods

1. Create a Person class with the following properties: name, age, and gender. Implement methods to set and get these properties.
2. Write a BankAccount class with the following properties: accountNumber, balance, and owner. Implement methods to deposit, withdraw, and check the account balance.
3. Create a Circle class with the following properties: radius. Implement methods to calculate the area and circumference of the circle.

Part 3: Conditional Statements and Loops

1. Write a program that takes a number as input and determines whether it is even or odd using an if-else statement.
2. Implement a program that uses a switch-case statement to determine the day of the week based on a numeric input (1 for Monday, 2 for Tuesday, etc.).
3. Create a program that uses a for loop to print the first 10 multiples of a given number.
4. Write a program that uses a while loop to calculate the factorial of a given number.

Part 4: Putting It All Together

Create a Student class with the following properties: name, grade, and age. Implement the following methods:

setName(String name): Sets the student's name.
getGrade(): Returns the student's grade.
incrementAge(): Increments the student's age by 1.
printStudentInfo(): Prints the student's name, grade, and age.

In the main method, create an array of Student objects and perform the following tasks:

Prompt the user to enter the number of students.

Create the Student objects and populate their information.

Use a for loop to iterate through the array and call the printStudentInfo() method for each student.

Find the student with the highest grade and print their information.

Note: Remember to include appropriate comments and documentation throughout your code to make it more readable and maintainable.

Case Study Based Questions

Problem 1: Currency Converter

Write a program that can convert between different currencies. The program should have the following features:

- Prompt the user to enter the amount they want to convert.
- Provide a list of available currencies (e.g., USD, EUR, JPY, GBP, INR) and ask the user to select the source and target currencies.
- Implement the currency conversion logic using appropriate exchange rates.
- Display the converted amount to the user.

Problem 2: Temperature Analyzer

Create a program that analyzes temperature data. The program should have the following features:

- Allow the user to input a series of temperature readings (in Celsius).
- Calculate and display the average temperature.
- Determine and display the highest and lowest temperatures.
- Categorize the temperatures into "cold," "mild," and "hot" based on predefined temperature ranges.

Problem 3: Student Grade Management

Develop a program to manage student grades. The program should have the following features:

- Create a Student class with properties for name, student ID, and grades (in an array).
- Implement methods to:
- Add a new student
- Update a student's grades
- Calculate the average grade for a student
- Find the student with the highest average grade

Prompt the user to perform various operations, such as adding a new student, updating a student's grades, and displaying the student with the highest average grade.

Problem 4: Fibonacci Sequence Generator

Write a program that generates the Fibonacci sequence. The program should have the following features:

- Prompt the user to enter the number of Fibonacci numbers to generate.
- Use a loop to calculate and display the Fibonacci sequence up to the specified number of terms.
- Optionally, allow the user to choose whether to display the sequence using a for loop, while loop, or recursive method.

Problem 5: Palindrome Checker

Develop a program that checks whether a given string is a palindrome. The program should have the following features:

- Prompt the user to enter a string.
- Implement a method to determine if the string is a palindrome (a word, phrase, number, or other sequence of characters that reads the same backward as forward).
- Display the result (whether the input is a palindrome or not) to the user.

Problem 6: BMI Calculator

Create a program that calculates a person's Body Mass Index (BMI) and provides an interpretation of the result. The program should have the following features:

- Prompt the user to enter their height (in meters) and weight (in kilograms).
- Calculate the BMI using the formula: BMI = weight / (height * height).
- Determine the BMI category (underweight, normal, overweight, or obese) based on the calculated BMI.
- Display the user's BMI value and the corresponding BMI category.

Problem 7: Leap Year Checker

Write a program that checks whether a given year is a leap year. The program should have the following features:

- Prompt the user to enter a year.
- Implement the logic to determine if the year is a leap year (a year divisible by 4, except for years divisible by 100, unless they are also divisible by 400).
- Display whether the entered year is a leap year or not.

Problem 8: Coin Toss Simulator

Develop a program that simulates a coin toss. The program should have the following features:

- Allow the user to choose the number of coin tosses to perform.
- Randomly generate the outcome of each coin toss (heads or tails).
- Keep track of the number of heads and tails.
- Display the results, including the number of heads and tails, as well as the percentage of each outcome.

Problem 9: Password Validator

Create a program that validates the strength of a user's password. The program should have the following features:

- Prompt the user to enter a password.
- Implement a set of rules to check the password's strength, such as minimum length, inclusion of uppercase and lowercase letters, digits, and special characters.
- Provide feedback to the user on the password's strength (e.g., weak, medium, strong).
- Allow the user to try again if the password is not strong enough.

Problem 10: Grocery List Manager

Develop a program that helps users manage their grocery list. The program should have the following features:

- Allow the user to add items to the grocery list.
- Provide the ability to mark items as purchased or remove them from the list.
- Display the current grocery list, including the purchased and remaining items.
- Implement a search functionality to find specific items on the list.
- Optionally, allow the user to save and load the grocery list to/from a file.