GIRI's Tech Hub, Pune

Programming (Machine) Test

Batch: June-2024 **Date: 12/11/2024**

Time: 10:00 to 12:00 Pm

Instructions: **Total:- 05 Marks**

1. Solve any four questions.
2. Input should be from user.
3. Indentation and comments mandatory.
4. Each program 1 Marks and 1 Marks Comments.

Q1. Write a POJO class Product with:
String productid, String productName, double price, int quantity ;
In the Product class : Include getters and setters for each field.
Create a constructor to initialize the productId, productName, price, and quantity.
Override the toString() method to print out the product details.
In the main method, create a few Product instances, modify their attributes using setters, and print the details.

Perform this operation :

- **Calculate total bill of all products.**
- **Find Product with Highest Quantity.**
- **Apply Discount to All Products.**

Q2. Create a class Employee with the following attributes:
int id , String name , double salary.
Create a parameterized constructor that initializes the id, name, and salary fields.
calculateBonus(): Write a method calculateBonus() that returns a bonus amount based on the employee's salary as follows:
If salary is greater than or equal to 100,000, bonus is 15% of salary.
If salary is between 50,000 (inclusive) and 100,000, bonus is 10% of salary.
If salary is less than 50,000, bonus is 5% of salary.
displayEmployeeDetails(): Write a method displayEmployeeDetails() that displays the employee's id, name, salary, calculated bonus, and total compensation (salary + bonus).
Create an Employee object using the parameterized constructor.
Call calculateBonus() and displayEmployeeDetails() to show the employee's information along with the bonus and total compensation.

Q3. Create a class BankAccount with the following attributes:

- **String accountNumber , double balance ;**

Methods:

1. **deposit(double amount): Adds the specified amount to the balance. If the amount is less than or equal to zero, display an error message.**
2. **withdraw(double amount): Deducts the specified amount from the balance. If the amount is greater than the balance, display an error message indicating insufficient funds. If amount is less than or equal to zero, display an error message for invalid withdrawal amount.**

Create a subclass SavingsAccount that inherits from BankAccount and adds an attribute:

- **double interestRate - to store the interest rate in percentage.**

applyInterest(): Calculates the interest based on the interestRate and adds it to the balance.

- Instantiate a SavingsAccount object with an account number, initial balance, and interest rate.

- Perform the following operations in sequence:

- **Deposit an amount into the account.**
- **Attempt a withdrawal with sufficient funds.**
- **Attempt a withdrawal with an amount greater than the balance (to test the error message).**
- **Apply interest to the account.**
- **Display the final balance and all the transaction details.**

Q4. Write an abstract class Appliance with:

Attributes: String brand, double power;

Constructor: Initialize brand and power.

Abstract Method:

abstract void turnOn() - This method should print a message indicating that the appliance is turned on.

Create a concrete subclass WashingMachine with:

Additional attribute int capacity. A constructor to initialize brand, power, and capacity.

Implementation of turnOn() to print a message about starting the washing machine with the specific capacity.

In the main method, create an instance of WashingMachine and call the turnOn() method.

Q5. Create a class Book with the following attributes:

Attributes : String title , String author , double price , int stockQuantity ;

Add the following:

1. **Parameterized Constructor:**
   - **Initialize all attributes: title, author, price, and stockQuantity.**
   - **If any of the parameters have invalid values (e.g., price is negative, stockQuantity is negative), set default values (e.g., price should default to 0.0 and stockQuantity should default to 0).**
2. **Overloaded Constructor:**
   - **Create an overloaded constructor that initializes only title and author, with default values for price and stockQuantity (e.g., price = 0.0, stockQuantity = 10).**
3. **Methods:**
   - **displayBookInfo(): Display the details of the book.**
   - **updateStock(int quantity): Update the stockQuantity based on the quantity provided. If the quantity is negative, display an error message.**
   - **applyDiscount(double discountPercentage): Apply a discount to the book's price. The discount should be a percentage (e.g., 10 means 10% off). Ensure the price never becomes negative after the discount.**
   - **getDiscountedPrice(): Return the discounted price.**
4. **Validation:**
   - **Ensure that the price and stock quantity are always valid by implementing a validation method. If invalid input is provided during object creation, the system should handle it gracefully by setting the invalid attributes to default values.**
5. **Main Method:**
   - **In the main method, create multiple Book instances using both constructors, apply discounts to them, update stock quantities, and display their updated details. You can use the displayBookInfo() method to print out the details of the books.**

Expected Functionality:

- **The program should handle invalid inputs (like negative prices or stock quantities).**
- **After applying discounts, the price should reflect the change, and the discounted price should not fall below zero.**
- **The stock quantity should be updated based on user input, and errors should be caught if an invalid quantity (negative number) is given.**
- **Finally, when displaying the details of a book, the updated price and stock should be shown.**

Q6. You need to create an interface Payable that has a method calculateSalary(). Then, you will create two classes, FullTimeEmployee and PartTimeEmployee, that implement the Payable interface.

Requirements:

1. **Interface Payable:**
    - **Method calculateSalary(): This method should be used to calculate and return the salary of an employee.**
2. **Class FullTimeEmployee:**
    - **Attributes:**
        - **String name**
        - **double monthlySalary**
    - **Constructor to initialize name and monthlySalary.**
    - **Implements calculateSalary() method to return the monthly salary.**
3. **Class PartTimeEmployee:**
    - **Attributes:**
        - **String name**
        - **double hourlyRate**
        - **int hoursWorked**
    - **Constructor to initialize name, hourlyRate, and hoursWorked.**
    - **Implements calculateSalary() method to return the total salary based on hourly rate and hours worked.**
4. **Main Class:**
    - **In the main class, create instances of both FullTimeEmployee and PartTimeEmployee.**
    - **Use polymorphism to call the calculateSalary() method of each employee.**

-----ALL THE BEST-----