National University of Computer & Emerging Sciences, Karachi

Fall-2023, School of Computing (BSCS, BSSE, BSAI)

Assignment # 2

Subject: Object Oriented Programming **Total Marks: 50**	**Post Date: 11th October 2023** **Due Date: 27th October 2023**
Course Instructor: Ms. Abeeha Sattar, Ms. Bakhtawer	

Instructions to be strictly followed.

1. **Each student should submit these files:**
   a. **A zip of all source files named as "A2-Q#[StudentID]" where # is the question number and Student ID is your ID.**
   b. **A DOC file where they copy code for each question and screen shot of the output. This document contains all the questions, answer codes and output in sequence. Name this document as "A2-[StudentID].docx".**
   c. **All the submissions will be made on Google Classroom.**
2. **Each output should have STUDENT ID and NAME of the student at the top.**
3. **It should be clear that your assignment would not get any credit if the assignment is submitted after the due date.**
4. **Zero grade for plagiarism (copy/ cheating) and late submissions.**

Task 1

A bank wants to design a system to manage different types of bank accounts for its customers. The bank offers two types of accounts: Savings Account and Checking Account. All accounts have a common set of features, such as the ability to deposit money, withdraw money, and check the account balance. However, each type of account has its own unique characteristics.

Base Class: Account

The **Account** class serves as the base class for all bank accounts. It has the following properties and methods:

- Properties:
    - **balance**: A double representing the current account balance.
- Constructor:
    - **Account(double initialBalance)**: Initializes the **balance** property with the provided initial balance. If the initial balance is negative, it sets the balance to 0 and displays an error message.
- Methods:
    - **void credit(double amount)**: Adds the given amount to the current balance.
    - **bool debit(double amount)**: Withdraws the specified amount from the account. Returns true if the debit is successful and false if the debit amount exceeds the account balance.
    - **double getBalance()**: Returns the current account balance.

Derived Class 1: SavingsAccount

The **SavingsAccount** class inherits from the **Account** class and adds the following properties and methods:

- Properties:
- **interestRate**: A double representing the interest rate (in percentage) associated with the account.
- Constructor:
    - **SavingsAccount(double initialBalance, double interestRate)**: Initializes the **balance** property using the base class constructor and sets the **interestRate**.

- Methods:
    - **double calculateInterest()**: Calculates and returns the amount of interest earned based on the current balance and interest rate.

Derived Class 2: CheckingAccount

The **CheckingAccount** class also inherits from the **Account** class and includes an additional property:

- Properties:
    - **transactionFee**: A double representing the fee charged per transaction.
- Constructor:
    - **CheckingAccount(double initialBalance, double transactionFee)**: Initializes the **balance** property using the base class constructor and sets the **transactionFee**.
- Method
    - **void credit(double amount)**: Overrides the base class method to subtract the transaction fee from the deposited amount.
    - **bool debit(double amount)**: Overrides the base class method to subtract the transaction fee from the withdrawn amount if the withdrawal is successful.

Your task is to implement the **Account**, **SavingsAccount**, and **CheckingAccount** classes as described above. Write a Java program that creates objects of each class, tests their member functions, and demonstrates the functionality of interest calculation for a **SavingsAccount**.

Task 2

A company pays its employees on a weekly basis. The employees are of four types:

1. **Salaried employees** are paid a fixed weekly salary regardless of the number of hours worked.

2. **Hourly employees** are paid by the hour and receive overtime pay (i.e., 1.5 times their hourly salary rate) for all hours worked more than 40 hours.

3. **Commission employees** are paid a percentage of their sales.

4. **Base-salaried commission employees** receive a base salary plus a percentage of their sales.

For the current pay period, the company has decided to reward salaried commission employees by adding 10% to their base salaries. The company wants to write an application that performs its payroll calculations polymorphically.

Task 3

The **Bee** class represents a bee in the hive. Its attributes might include the bee's age, gender, and current job within the hive (such as forager, nurse, or guard bee). The functions of the Bee class might include methods to set and get the bee's attributes, as well as methods to perform its assigned job.

If a Bee object is created by using the default constructor, it should always be a worker bee.

The **Hive** class represents a collection of bees living and working together. Its attributes might include the number of bees in the hive, the amount of honey stored in the hive, and the queen bee. The functions of the Hive class might include methods to add bees from to the hive, collect and store honey. The hive class also contains a member variable that stores the amount on honey in all hive objects.

Whenever a default Hive object is created, it should have a total of 10 Bees, and honey amount of 10 ml. However, in case of a custom amount, the custom number of bees should exist in the hive, along with the custom honey amount. Similarly, a default hive should have a default Queen Bee. The Queen Bee is a special type of bee who does not have any “job” (forager, nurse, guard, etc.), it would always be created as age 12, gender ‘F’ and job “Queen”.

Inside the parameterized constructor of your Hive class, make it so different bees are assigned different tasks. Try to make it so that there are equal number of bees for each task. A difference of one two bees per task is acceptable.

Now, in your main function, perform the following operations with your classes:

1. Dynamically allocate a queen bee of age 20, gender ‘F’ and job “Queen”
2. Create one default Hive object.
3. Create one Hive object with 20 bees, 20 honey amount, and the queen bee that you created in part 1.
4. Use the getter method of your Hive class to get the bees in the hives that you created in part 2 and 3, then print their jobs.
5. Display the total honey in both hives.

Task 4

Write a Java program that has a class named “Course”.

The class **Course** has the attributes course name, course code, class venue and credit hours. All are protected members. Set all these attributes with a parameterized constructor.

Derive a class “**Programming Fundamental Course**” that has an attribute: teacher name. For this class make a constructor and invoke the base class’s parameterized constructor. Set the teacher’s name in the constructor as well.

The derived class has a function Display that displays all the details of the course in the derived class.

In the main, create an object for Programming Fundamentals Course and display all the details, using the functions created.

Task 5

You are developing a library management system where users can borrow books. To make the borrowing process flexible, you decide to implement method overloading for handling different scenarios.

Requirements:

1. Create a class called **Library** with the following methods:

- **borrowBook(String bookName):** Method to borrow a book by its name.
- **borrowBook(String bookName, int numberOfCopies):** Method to borrow a specific number of copies of a book.
- **borrowBook(String bookName, String userName):** Method to borrow a book by a specific user.

2. All methods should print a message indicating the borrowing action.

3. Provide sample code to demonstrate the usage of the different overloaded methods, including borrowing books by name, borrowing multiple copies, and borrowing by a specific user.