PARAMASIVAM A 24BCS198

Experiment Number : 06 **Date: 05/02/2026**

ABSTRACT CLASSES

Aim:

To understand and implement inheritance concepts in Java.

PRE LAB EXERCISE

QUESTIONS

1. **What is an abstract class?**

   • An abstract class is a class declared using the abstract keyword.

   • It may contain both abstract methods and non-abstract (concrete) methods.

   • It cannot be instantiated (object cannot be created).

2. **Why are abstract methods used?**

   • Abstract methods do not have a method body.

   • They force child classes to provide their own implementation.

   • They help in achieving abstraction in Java.

3. **Difference between abstract class and interface.**

   Abstract Class:
   • Can have abstract and non-abstract methods
   • Supports constructors and instance variables
   • A class can extend only one abstract class
   Interface:
   • Contains only abstract methods (and default/static methods in newer Java)
   • Does not support constructors
   • A class can implement multiple interfaces

IN LAB EXERCISE

Objective:

To implement abstract class and demonstrate abstraction.

PROGRAMS:

1.University System

Scenario:
A university has different types of courses: Online, Offline, and Hybrid. Each course has a getDetails() method.

PARAMASIVAM A 24BCS198

Question:
Create an abstract class Course with abstract method getDetails(). Implement OnlineCourse, OfflineCourse, and HybridCourse classes.

Code:

```
abstract class Course {
    abstract void getDetails();
}


class OnlineCourse extends Course {
    void getDetails() {
        System.out.println("Online Course: Attend via Internet");
    }
}


class OfflineCourse extends Course {
    void getDetails() {
        System.out.println("Offline Course: Attend in classroom");
    }
}


class HybridCourse extends Course {
    void getDetails() {
        System.out.println("Hybrid Course: Combination of online and offline");
    }
}


public class Main {
    public static void main(String[] args) {
        Course c1 = new OnlineCourse();
        Course c2 = new OfflineCourse();

        Course c3 = new HybridCourse();


        c1.getDetails();

        c2.getDetails();

        c3.getDetails();

    }

}
```

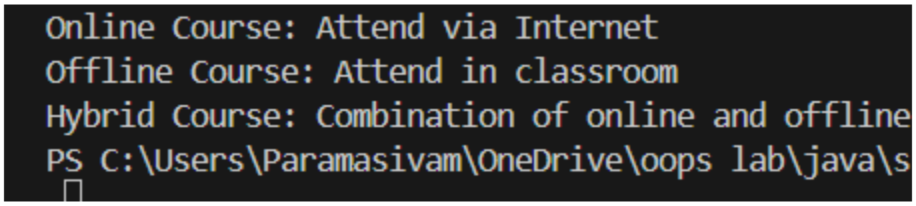
Output:

Online Course: Attend via Internet

Offline Course: Attend in classroom

Hybrid Course: Combination of online and offline

OUTPUT:



```
Online Course: Attend via Internet
Offline Course: Attend in classroom
Hybrid Course: Combination of online and offline
PS C:\Users\Paramasivam\OneDrive\oops lab\java\s
```

2. Employee Payroll System

Scenario:
A company has different types of employees — Regular and Contract. All employees have a salary, but the calculation differs for each type.

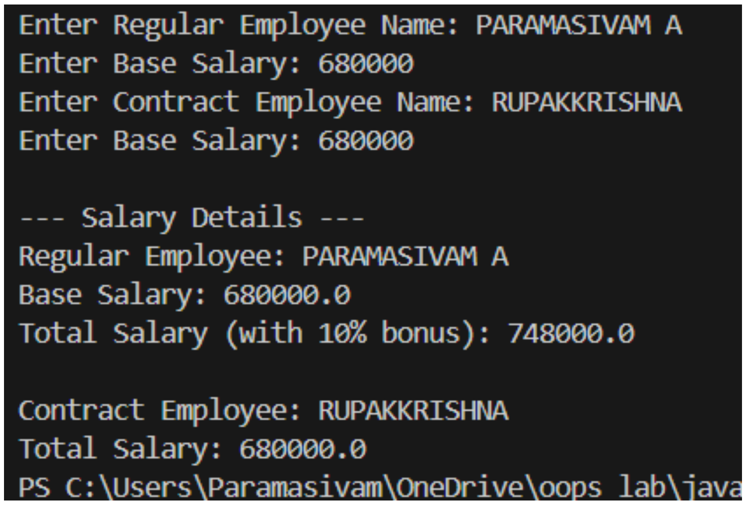
Question:
Design an abstract class Employee with an abstract method calculateSalary(). Implement subclasses RegularEmployee and ContractEmployee to calculate salary differently.

Code:

```
import java.util.Scanner;

abstract class Employee {

    String name;

    double baseSalary;
    // Abstract method to calculate total salary
    abstract void calculateSalary();
}

class RegularEmployee extends Employee {
    double bonusRate = 0.1; // 10% bonus

    void calculateSalary() {
        double totalSalary = baseSalary + (baseSalary * bonusRate);
        System.out.println("Regular Employee: " + name);
        System.out.println("Base Salary: " + baseSalary);
        System.out.println("Total Salary (with 10% bonus): " + totalSalary);
    }
}

class ContractEmployee extends Employee {
    void calculateSalary() {
        System.out.println("Contract Employee: " + name);
        System.out.println("Total Salary: " + baseSalary);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input for Regular Employee
        System.out.print("Enter Regular Employee Name: ");
        String regName = sc.nextLine();
        System.out.print("Enter Base Salary: ");
        double regSalary = sc.nextDouble();
        sc.nextLine(); // Consume newline

        // Input for Contract Employee
        System.out.print("Enter Contract Employee Name: ");
        String conName = sc.nextLine();
        System.out.print("Enter Base Salary: ");
        double conSalary = sc.nextDouble();

        // Create objects
        Employee e1 = new RegularEmployee();
        e1.name = regName;
        e1.baseSalary = regSalary;

        Employee e2 = new ContractEmployee();
        e2.name = conName;
        e2.baseSalary = conSalary;

        System.out.println("\n--- Salary Details ---");
        e1.calculateSalary();
        System.out.println();
        e2.calculateSalary();

        sc.close();
    }
}
```

Output:

```
Enter Regular Employee Name: Ram

Enter Base Salary: 30000

Enter Contract Employee Name: Ravi

Enter Base Salary: 20000


--- Salary Details ---

Regular Employee: Anitha

Base Salary: 30000.0

Total Salary (with 10% bonus): 33000.0


Contract Employee: Ravi

Total Salary: 20000.0
```



```
Enter Regular Employee Name: PARAMASIVAM A
Enter Base Salary: 680000
Enter Contract Employee Name: RUPAKKRISHNA
Enter Base Salary: 680000

--- Salary Details ---
Regular Employee: PARAMASIVAM A
Base Salary: 680000.0
Total Salary (with 10% bonus): 748000.0

Contract Employee: RUPAKKRISHNA
Total Salary: 680000.0
PS C:\Users\Paramasivam\OneDrive\oops lab\java
```

3.Banking System

Scenario:

A bank has different types of accounts: Savings and Current. Both accounts need a method to calculate interest, but the calculation differs for each account type.

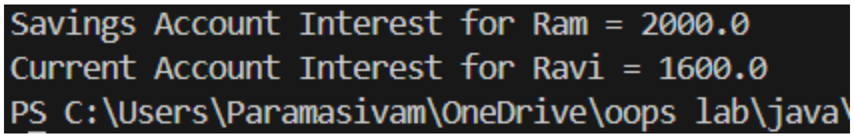
Question:

Use an abstract class BankAccount with an abstract method calculateInterest() and implement it in SavingsAccount and CurrentAccount classes.

Code

```
abstract class BankAccount {
    String accountHolder;
    double balance;

    BankAccount(String name, double bal) {
        accountHolder = name;
        balance = bal;
    }

    abstract void calculateInterest();  // Abstract method
}

class SavingsAccount extends BankAccount {
    double interestRate = 0.04; // 4% interest

    SavingsAccount(String name, double bal) {
        super(name, bal);
    }

    void calculateInterest() {
        double interest = balance * interestRate;
        System.out.println("Savings Account Interest for " + accountHolder + " = " + interest);
    }
}

class CurrentAccount extends BankAccount {
    double interestRate = 0.02; // 2% interest
    CurrentAccount(String name, double bal) {
        super(name, bal);
    }

    void calculateInterest() {
        double interest = balance * interestRate;
        System.out.println("Current Account Interest for " + accountHolder + " = " + interest);
    }
}

public class Main {
    public static void main(String[] args) {
        BankAccount acc1 = new SavingsAccount("Ram", 50000);
        BankAccount acc2 = new CurrentAccount("Ravi", 80000);

        acc1.calculateInterest();
        acc2.calculateInterest();
    }
}
```

OUTPUT:



```
Savings Account Interest for Ram = 2000.0
Current Account Interest for Ravi = 1600.0
PS C:\Users\Paramasivam\OneDrive\oops lab\java\
```

Output

Savings Account Interest for Ram = 2000.0

Current Account Interest for Ravi = 1600.0

POST LAB EXERCISE

PARAMASIVAM A 24BCS198

1. **How is an abstract class different from a regular class?**

   • An abstract class may contain abstract methods, a regular class cannot.

   • An abstract class cannot be instantiated, a regular class can be instantiated.

   • Abstract classes are used for abstraction, while regular classes provide full implementation.

2. **Can you create an object of an abstract class? Why or why not?**

   • No, you cannot create an object of an abstract class.

   • It may contain unimplemented (abstract) methods.

   • Objects are created only for fully implemented classes.

3. **What happens if a subclass does not implement an abstract method?**

   • The subclass must implement all inherited abstract methods.

   • If it does not, the subclass itself becomes abstract.

   • Otherwise, the program will result in a compile-time error.

4. **Can an abstract class exist without any abstract methods?**

   • Yes, an abstract class can exist without abstract methods.

   • It is declared abstract to prevent object creation.

   • It can be used as a base class for inheritance.

5. **Can an abstract class extend another abstract class?**

   • Yes, an abstract class can extend another abstract class.

   • It may or may not implement the inherited abstract methods.

   • Implementation can be done in a concrete subclass later.

Result:

Thus the abstract classes and methods were implemented and executed successfully.

ASSESSMENT

Description	**Max Marks**	**Marks Awarded**
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		