**Experiment Number : 06** **Date:**

## ABSTRACT CLASSES

**Aim:**

To understand and implement inheritance concepts in Java.

**PRE LAB EXERCISE**

**QUESTIONS**

- ✓ **What is an abstract class?**

  An abstract class is a class that cannot be instantiated and may contain abstract and non-abstract methods.

- ✓ **Why are abstract methods used?**

  Abstract methods are used to define methods without implementation that must be implemented by subclasses.

- ✓ **Difference between abstract class and interface.**

  An abstract class can have constructors and instance variables, while
  an interface supports multiple inheritance and contains only abstract methods.

**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.

**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

```
Online Course: Attend via Internet
Offline Course: Attend in classroom
Hybrid Course: Combination of online and offline
```

**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

```
--- Salary Details ---
Regular Employee: JAI
Base Salary: 20000.0
Total Salary (with 10% bonus): 22000.0

Contract Employee: RAM
Total Salary: 40000.0
```

**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;
```

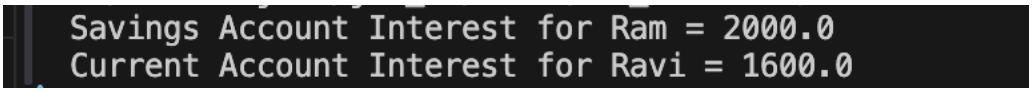
```
    }

    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




**POST LAB EXERCISE**

- ✓ **How is an abstract class different from a regular class?**

  Abstract class vs regular class: An abstract class can have abstract methods, while a regular class cannot.

- ✓ **Can you create an object of an abstract class? Why or why not?**

  Object of abstract class: No, because an abstract class is incomplete and meant to be inherited.

- ✓ **What happens if a subclass does not implement an abstract method?**

  Subclass not implementing abstract method: The subclass must be declared abstract.

- ✓ **Can an abstract class exist without any abstract methods?**

  Abstract class without abstract methods: Yes, it can exist for design purposes.

- ✓ **Can an abstract class extend another abstract class?**

  Abstract class extending another abstract class: Yes, it is allowed in Java.

**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** | | |