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

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 declared using the abstract keyword and may contain abstract and non-abstract methods. It cannot be instantiated directly. Abstract classes are used as a base for other classes. They provide common functionality while forcing subclasses to implement specific methods. This supports abstraction in Java.

- ✓ Why are abstract methods used?

  Abstract methods are used to define a method without providing its implementation. They ensure that subclasses provide their own implementation of the method. This enforces a common structure across related classes. Abstract methods help achieve abstraction and flexibility. They allow different behaviors for the same method.

- ✓ Difference between abstract class and interface.

  An abstract class can contain both abstract and concrete methods, whereas an interface contains only abstract methods (and default/static methods). A class can extend only one abstract class, but it can implement multiple interfaces. Abstract classes can have instance variables, while interfaces have only constants. Interfaces provide full abstraction, while abstract classes provide partial abstraction.

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:

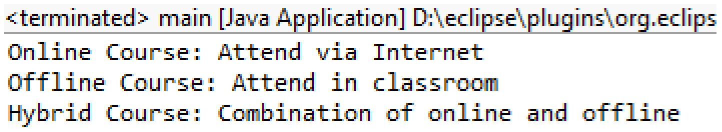
```
abstract class Course {
    abstract void getDetails();
}


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



```
<terminated> main [Java Application] D:\eclipse\plugins\org.eclips
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.

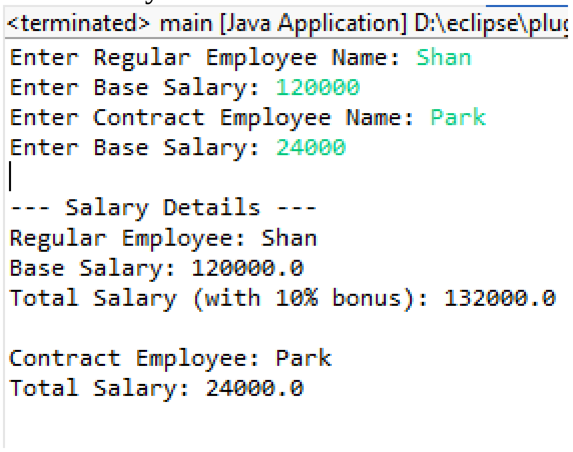
Question:
Design an abstract class Employee with an abstract method calculateSalary(). Implement subclasses RegularEmployee and ContractEmployee to calculate salary differently.

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: Shan
Enter Base Salary: 120000
Enter Contract Employee Name: Park
Enter Base Salary: 24000

--- Salary Details ---
Regular Employee: Shan
Base Salary: 120000.0
Total Salary (with 10% bonus): 132000.0

Contract Employee: Park
Total Salary: 24000.0
```



```
<terminated> main [Java Application] D:\eclipse\plu
Enter Regular Employee Name: Shan
Enter Base Salary: 120000
Enter Contract Employee Name: Park
Enter Base Salary: 24000

--- Salary Details ---
Regular Employee: Shan
Base Salary: 120000.0
Total Salary (with 10% bonus): 132000.0

Contract Employee: Park
Total Salary: 24000.0
```

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:

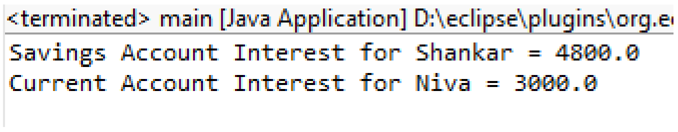
Use an abstract class BankAccount with an abstract method calculateInterest() and implement it in SavingsAccount and CurrentAccount classes.

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("Shankar", 120000);
        BankAccount acc2 = new CurrentAccount("Niva", 150000);

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

Output

```
Savings Account Interest for Shankar = 4800.0
Current Account Interest for Ravi = 3000.0
```



```
<terminated> main [Java Application] D:\eclipse\plugins\org.e
Savings Account Interest for Shankar = 4800.0
Current Account Interest for Niva = 3000.0
```

POST LAB EXERCISE

✓ How is an abstract class different from a regular class?
An abstract class can have abstract methods as well as concrete methods, while a regular class contains only concrete methods. Abstract classes cannot be instantiated directly. They are meant to be inherited by other classes. Regular classes can be used to create objects directly. Abstract classes are mainly used to provide a common base for subclasses.

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

No, an object of an abstract class cannot be created. This is because abstract classes may contain abstract methods without implementation. These methods must be implemented by subclasses. Creating an object without complete method definitions is not allowed. Hence, abstract classes are only used through inheritance.

- ✓ What happens if a subclass does not implement an abstract method?
  If a subclass does not implement all abstract methods, it must be declared as abstract. Otherwise, the compiler will throw an error. This ensures that all abstract methods are implemented before object creation. Java enforces this rule for program consistency. It maintains proper abstraction.

- ✓ Can an abstract class exist without any abstract methods?
  Yes, an abstract class can exist without any abstract methods. It is declared abstract to prevent object creation. Such classes are often used as base classes. They can provide common functionality for subclasses. This supports code reuse and design control.

- ✓ Can an abstract class extend another abstract class?
  Yes, an abstract class can extend another abstract class. It may implement some or none of the abstract methods of the parent class. The remaining abstract methods can be implemented by its subclasses. This allows building layered abstractions. It improves flexibility in class design.

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		