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

NAME:PAVAN KRISHNA
ROLL NO:24BCS201

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 is declared using the abstract keyword and **cannot be instantiated**. It may contain **abstract methods (without body)** as well as **concrete methods (with body)**. It is used as a **base class** for other classes.

✓ Why are abstract methods used?

Abstract methods are used to **force subclasses to provide their own implementation**. They define a **method structure** without implementation, ensuring **uniform behavior** across different subclasses.

✓ Difference between abstract class and interface.

Abstract class can have both abstract and normal methods and supports constructors.

Interface contains only abstract methods (and constants) and does not support constructors.

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.

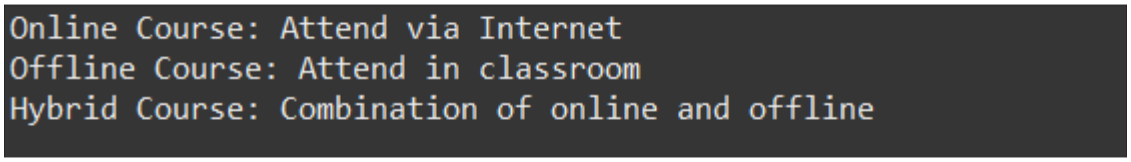
Code:

```
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:

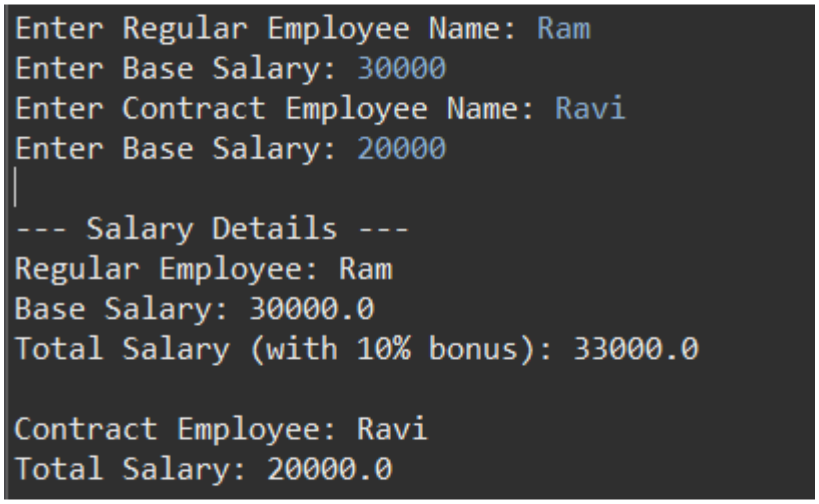
```
import java.util.Scanner;

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



```
Enter Regular Employee Name: Ram
Enter Base Salary: 30000
Enter Contract Employee Name: Ravi
Enter Base Salary: 20000

--- Salary Details ---
Regular Employee: Ram
Base Salary: 30000.0
Total Salary (with 10% bonus): 33000.0

Contract Employee: Ravi
Total Salary: 20000.0
```

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

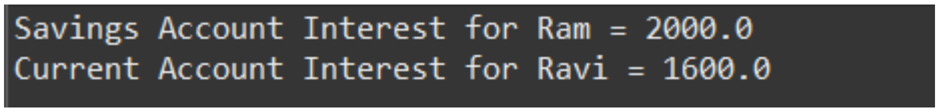
```
abstract class BankAccount {
   String accountHolder;
   double balance;


   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
```



```
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?

An abstract class can contain **abstract methods (methods without body)**, whereas a regular class cannot. An abstract class **cannot be instantiated**, while a regular class **can be instantiated**.

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

  No, you **cannot create an object of an abstract class** because it may contain abstract methods that are **incomplete and must be implemented by subclasses**.

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

  If a subclass does not implement all abstract methods of its parent class, then the subclass **must also be declared abstract**, otherwise a **compile-time error** occurs.

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

  Yes, an abstract class **can exist without any abstract methods**. It is used to **prevent object creation** and to provide a **base class** for other classes.

- ✓ 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 abstract class.

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		