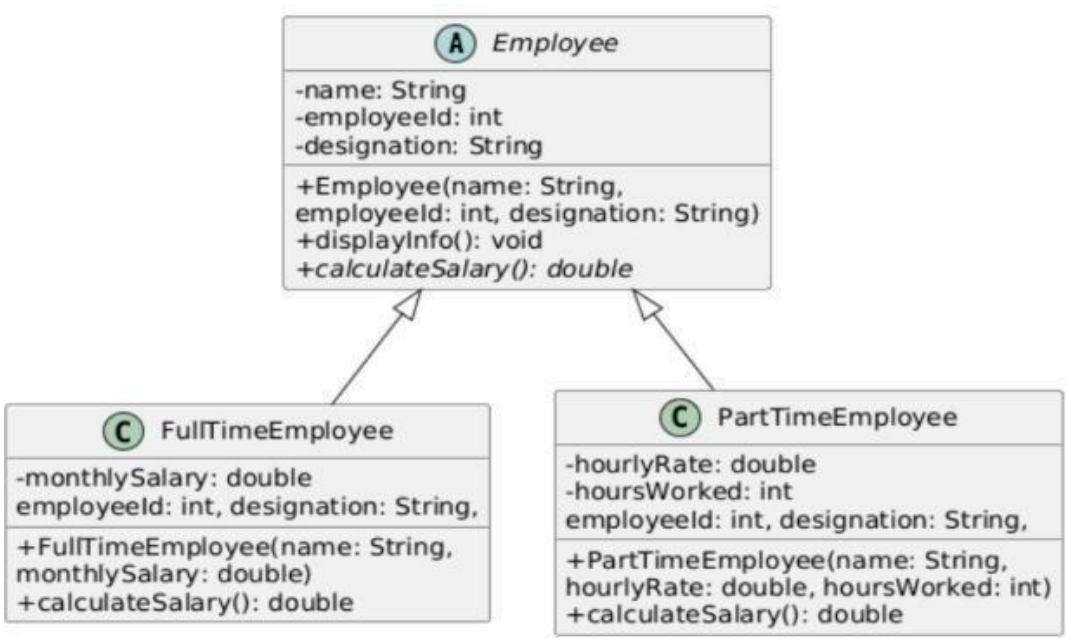
Ex no: 3	Develop programs incorporating packages, abstract classes, and interfaces to structure code modularly and enforce abstraction.
Date:	

Aim:

To Develop programs incorporating packages, abstract classes, and interfaces to structure code modularly and enforce abstraction.

Question 1:

Write a Java program to implement the following relationship and create a Main class to invoke all the methods.



Code:

```
package javaproject;

public class Employee
{
private String name;
private int employeeId;
private String designation;
public Employee(String name, int employeeId, String designation) {
this.name = name;
this.employeeId = employeeId;
this.designation = designation;
   }
public String getName() {
return name;
   }
   public int getEmployeeId()
{

return employeeId;
```

```
    }
      public String getDesignation() {
      return designation;
      }
   public void displayInfo() {
   System.out.println("Name: " + getName());
   System.out.println("Employee ID: " + getEmployeeId());
   System.out.println("Designation: " + getDesignation());
     }
    public double calculateSalary() {
    return0.0;
     }
   }
   class FullTimeEmployee extends Employee {
    private double monthlySalary;
    public FullTimeEmployee(String name, int employeeId, String designation, double monthlySalary)
     {
    super(name, employeeId, designation);
    this.monthlySalary = monthlySalary;
     }
   @Override
public double calculateSalary() {
return monthlySalary;
    }
}
class PartTimeEmployee extends Employee {
private double hourlyRate;
private int hoursWorked;
public PartTimeEmployee(String name, int employeeId, String designation, double hourlyRate,
int hoursWorked)
{
super(name, employeeId, designation);
this.hourlyRate = hourlyRate;
this.hoursWorked = hoursWorked;
    }
    @Override
public double calculateSalary() {
return hourlyRate * hoursWorked;
    }
public void setHourlyRate(double hourlyRate) {
this.hourlyRate = hourlyRate;
    }
    public void setHoursWorked(int hoursWorked)
   {
    this.hoursWorked = hoursWorked;
    }
}
public class Main {
```

```
 public static void main(String[]
args) {
        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee("Geetha", 101, "Software
Engineer",  5000.0);
fullTimeEmployee.displayInfo();
 System.out.println("Salary: " + fullTimeEmployee.calculateSalary());
 System.out.println();
 PartTimeEmployee partTimeEmployee = new PartTimeEmployee("Kumar",105, "Intern", 80.0, 100);
partTimeEmployee.displayInfo();
 System.out.println("Salary: " + partTimeEmployee.calculateSalary());
partTimeEmployee.setHourlyRate(25.0);
partTimeEmployee.setHoursWorked(80);
 System.out.println("\nAfter updating Bob's salary:");
partTimeEmployee.displayInfo();
 System.out.println("Salary: " + partTimeEmployee.calculateSalary());
        }
}
```

Output:

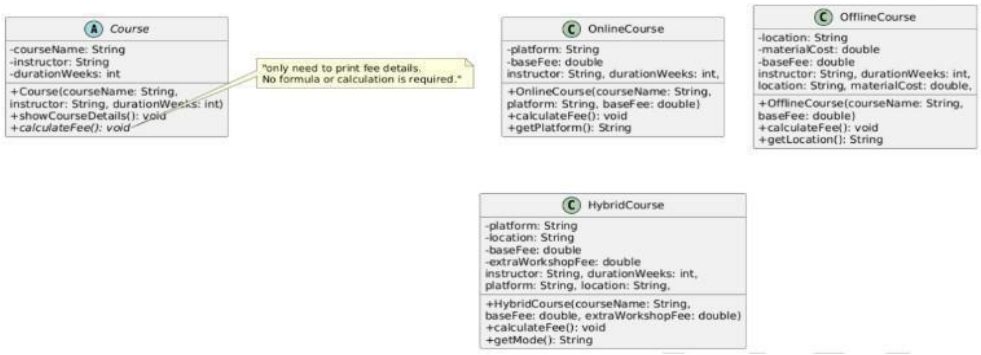
```
<terminated> Main [Java Application] C:\Users\SRINITHI\.p2\pool\plugins\org
Name: Geetha
Employee ID: 101
Designation: Software Engineer
Salary: 5000.0

Name: Kumar
Employee ID: 105
Designation: Intern
Salary: 8000.0

After updating Bob's salary:
Name: Kumar
Employee ID: 105
Designation: Intern
Salary: 2000.0
```

Question 2:

Identify the relationship between the classes and write a Java program to implement the relationship.



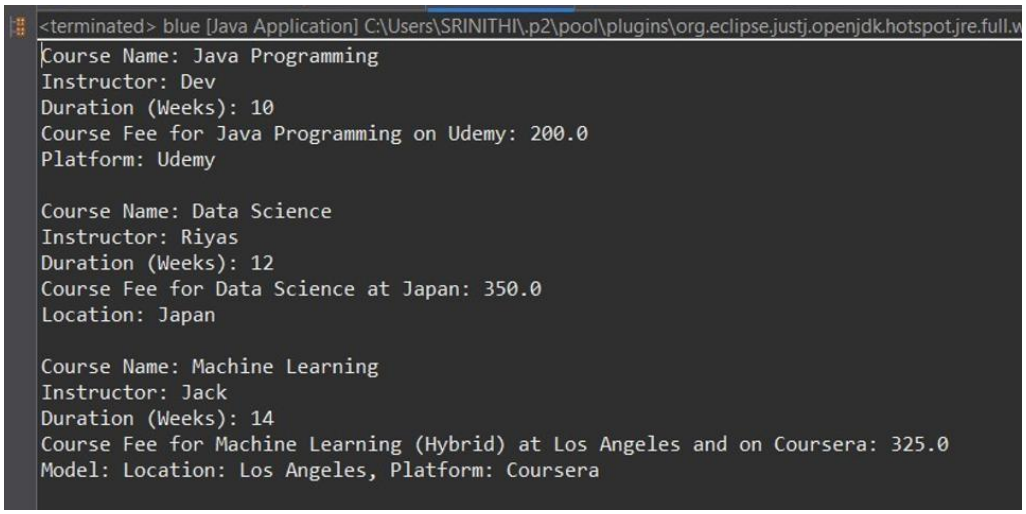
Code:

```
class Course {
    protected String courseName;
    protected String instructor;
    protected int durationWeeks;
    public Course(String courseName, String instructor, int durationWeeks) {
        this.courseName = courseName;
        this.instructor = instructor;
        this.durationWeeks = durationWeeks;
    }
    public void showCourseDetails() {
        System.out.println("Course Name: " + courseName);
        System.out.println("Instructor: " + instructor);
        System.out.println("Duration (weeks): " + durationWeeks);
    }
    public void calculateFee() {
        System.out.println("Fee details are not specified for base Course.");
    }
}
class OnlineCourse extends Course {
    private String platform;
    private double baseFee;
    public OnlineCourse(String courseName, String instructor, int durationWeeks,
                        String platform, double baseFee) {
        super(courseName, instructor, durationWeeks);
        this.platform = platform;
        this.baseFee = baseFee;
    }
    @Override
    public void calculateFee() {
        System.out.println("Online Course Fee: " + baseFee);
    }
    public String getPlatform() {
        return platform;
```

```
    }
}
class OfflineCourse extends Course {
    private String location;
    private double materialCost;
    private double baseFee;
    public OfflineCourse(String courseName, String instructor, int durationWeeks,
                         String location, double materialCost, double baseFee) {
        super(courseName, instructor, durationWeeks);
        this.location = location;
        this.materialCost = materialCost;
        this.baseFee = baseFee;
    }
    @Override
    public void calculateFee() {
        System.out.println("Offline Course Fee: " + (baseFee + materialCost));
    }
    public String getLocation() {
        return location;
    }
}
class HybridCourse extends Course {
    private String platform;
    private String location;
    private double baseFee;
    private double extraWorkshopFee;
    public HybridCourse(String courseName, String instructor, int durationWeeks,
                        String platform, String location,
                        double baseFee, double extraWorkshopFee) {
        super(courseName, instructor, durationWeeks);
        this.platform = platform;
        this.location = location;
        this.baseFee = baseFee;
        this.extraWorkshopFee = extraWorkshopFee;
    }
    @Override
    public void calculateFee() {
        System.out.println("Hybrid Course Fee: " + (baseFee + extraWorkshopFee));
    }
    public String getMode() {
        return "Platform: " + platform + ", Location: " + location;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Course online = new OnlineCourse("Java Programming", "Dr. Smith", 8, "Udemy", 5000);
        Course offline = new OfflineCourse("C++ Basics", "Prof. John", 6, "New York Center", 1000, 8000);
        Course hybrid = new HybridCourse("Python Advanced", "Dr. Alice", 10, "Coursera", "London
Center", 7000, 2000);
        System.out.println("=== Online Course ===");
        online.showCourseDetails();
        online.calculateFee();
        System.out.println("\n=== Offline Course ===");
        offline.showCourseDetails();
        offline.calculateFee();
        System.out.println("\n=== Hybrid Course ===");
        hybrid.showCourseDetails();
        hybrid.calculateFee();
    }
}
```

Output:



Question 3:

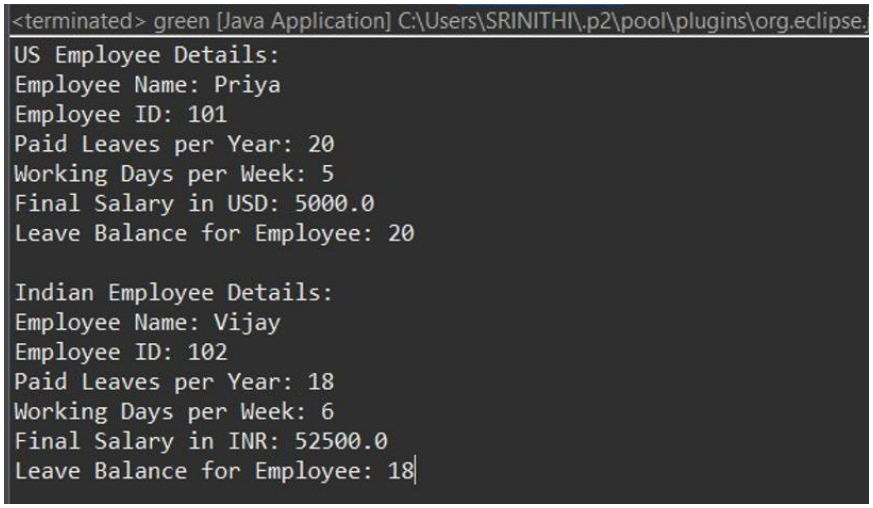
Write a Java program to implement the following relationship and create a Main class to invoke all the methods.

Code:

```
abstract class Employee {
    public static final String COMPANY_NAME = "TechCorp Pvt Ltd";
    public abstract void getDetails();
    public abstract void calculateSalary(double basicPay, double bonus, double taxRate);
    public abstract void showLeaveBalance();
}
class USEmployee extends Employee {
    private String name;
    private int employeeId;
    private int paidLeavesPerYear;
    private int workingDaysPerWeek;
    public USEmployee(String name, int employeeId, int paidLeavesPerYear, int workingDaysPerWeek)
{
        this.name = name;
        this.employeeId = employeeId;
        this.paidLeavesPerYear = paidLeavesPerYear;
        this.workingDaysPerWeek = workingDaysPerWeek;
    }
    @Override
    public void getDetails() {
        System.out.println("=== US Employee Details ===");
        System.out.println("Company: " + COMPANY_NAME);
        System.out.println("Name: " + name);
        System.out.println("Employee ID: " + employeeId);
        System.out.println("Paid Leaves per Year: " + paidLeavesPerYear);
        System.out.println("Working Days per Week: " + workingDaysPerWeek);
    }
    @Override
    public void calculateSalary(double basicPay, double bonus, double taxRate) {
        double finalSalary = (basicPay + bonus) - (basicPay * taxRate / 100);
        System.out.println("Final Salary (USD): $" + finalSalary);
    }
    @Override
    public void showLeaveBalance() {
```

```
        System.out.println("Leave Balance (US Employee): " + paidLeavesPerYear + " days/year");
    }
}
class IndianEmployee extends Employee {
    private String name;
    private int employeeId;
    private int paidLeavesPerYear;
    private int workingDaysPerWeek;
    public IndianEmployee(String name, int employeeId, int paidLeavesPerYear, int
workingDaysPerWeek) {
        this.name = name;
        this.employeeId = employeeId;
        this.paidLeavesPerYear = paidLeavesPerYear;
        this.workingDaysPerWeek = workingDaysPerWeek;
    }
    @Override
    public void getDetails() {
        System.out.println("=== Indian Employee Details ===");
        System.out.println("Company: " + COMPANY_NAME);
        System.out.println("Name: " + name);
        System.out.println("Employee ID: " + employeeId);
        System.out.println("Paid Leaves per Year: " + paidLeavesPerYear);
        System.out.println("Working Days per Week: " + workingDaysPerWeek);
    }
    @Override
    public void calculateSalary(double basicPay, double bonus, double taxRate) {
        double finalSalary = (basicPay + bonus) - (basicPay * taxRate / 100);
        System.out.println("Final Salary (INR): ₹" + finalSalary);
    }
    @Override
    public void showLeaveBalance() {
        System.out.println("Leave Balance (Indian Employee): " + paidLeavesPerYear + " days/year");
    }
}
public class Main {
    public static void main(String[] args) {
        Employee usEmp = new USEmployee("John Doe", 101, 20, 5);
        Employee inEmp = new IndianEmployee("Raj Kumar", 202, 25, 6);
        System.out.println("\n--- US Employee ---");
        usEmp.getDetails();
        usEmp.calculateSalary(5000, 500, 10);
        usEmp.showLeaveBalance();
        System.out.println("\n--- Indian Employee ---");
        inEmp.getDetails();
        inEmp.calculateSalary(40000, 5000, 15);
        inEmp.showLeaveBalance();
    }
}
```

Output:



Result:

Thus, the program was implemented using incorporating packages, abstract classes, and interfaces to structure code modularly and enforce abstraction and the output was verified.