Ques1: -Illustrate the usage of abstract class with following Java classes – λ An abstract class ‘student’ with two data members roll no, reg no, a method getinput() and an abstract method course() λ A subclass ‘kiitian’ with course() method implementation Write the driver class to print the all details of a kiitian object

```
import java.util.Scanner;

abstract class Student {

    int rollNo;
    String regNo;

    void getInput() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Roll No: ");
        rollNo = sc.nextInt();
        System.out.print("Enter Reg No: ");
        regNo = sc.next();
        sc.close();

    }

    abstract void course();
}

class KiiTian extends Student {

    void course() {
        System.out.println("Course: Computer Science");
    }
}

public class program1 {
    public static void main(String[] args) {

        KiiTian student1 = new KiiTian();

        student1.getInput();

        System.out.println("Details:");
        System.out.println("Roll No: " + student1.rollNo);
        System.out.println("Reg No: " + student1.regNo);
        student1.course();
    }
}
```

Ques2: Define an interface Motor with a data member –capacity and two methods such as run() and consume(). Define a Java class ‘Washing machine’ which implements this interface and write the code to check the value of the interface data member thru an object

```
interface Motor {
    int capacity = 5;

    void run();

    void consume();
}

class WashingMachine implements Motor {

    public void run() {
        System.out.println("Washing machine is running.");
    }

    public void consume() {
        System.out.println("Washing machine is consuming electricity.");
    }
}

public class program2 {
    public static void main(String[] args) {

        WashingMachine washingMachine = new WashingMachine();

        System.out.println("Motor Capacity: " + Motor.capacity);

        washingMachine.run();
        washingMachine.consume();
    }
}
```

Ques3: Define an interface with three methods – earnings(), deductions() and bonus() and define a Java class ‘Manager’ which uses this interface without implementing bonus() method. Also define another Java class ‘Substaff’ which extends from ‘Manager’ class and implements bonus() method. Write the complete program to find out earnings, deduction and bonus of a sbstaff with basic salary amount entered by the user as per the following guidelines – earnings basic + DA (80% of basic) + HRA (15% of basic) deduction PF 12% of basic bonus 50% of basic

```
import java.util.Scanner;

interface Employee {
    double earnings(double basic);
```

```
    double deductions(double basic);

    default void bonus(){};
}
class Manager implements Employee {

    public double earnings(double basic) {
        double da = 0.8 * basic;
        double hra = 0.15 * basic;
        return basic + da + hra;
    }

    public double deductions(double basic) {
        return 0.12 * basic;
    }

}

class Substaff extends Manager {
    public double bonus(double basic) {
        return 0.5 * basic;
    }
}

public class program3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter Basic Salary: ");
        double basicSalary = scanner.nextDouble();
        scanner.close();

        Substaff substaff = new Substaff();

        System.out.println("Earnings: " + substaff.earnings(basicSalary));
        System.out.println("Deductions: " + substaff.deductions(basicSalary));
        System.out.println("Bonus: " + substaff.bonus(basicSalary));
    }
}
```

Ques 4: Define an interface Emploee with a method getDetails() to get emplyee details as Empid and Ename. Also define a derived interface Manager with a method getDeptDetails() to get department details such as Deptid and Deptname.Then define a class Head which implements Manager interface and also prints all details of the employee. Write the complete program to display all details of one head of the department.

```java
interface Employee {
    void getDetails();
}

interface Manager extends Employee {
    void getDeptDetails();
}

class Head implements Manager {
    private int empId;
    private String empName;
    private int deptId;
    private String deptName;

    public Head(int empId, String empName, int deptId, String deptName) {
        this.empId = empId;
        this.empName = empName;
        this.deptId = deptId;
        this.deptName = deptName;
    }

    public void getDetails() {
        System.out.println("Employee Details:");
        System.out.println("Employee ID: " + empId);
        System.out.println("Employee Name: " + empName);
    }

    public void getDeptDetails() {
        System.out.println("Department Details:");
        System.out.println("Department ID: " + deptId);
        System.out.println("Department Name: " + deptName);
    }

    public void displayHeadDetails() {
        getDetails();
        getDeptDetails();
    }

    public static void main(String[] args) {

        Head head = new Head(101, "John Doe", 201, "Engineering");

        head.displayHeadDetails();
    }
}
```