Name	Lekh Sanatan Nayak
UID no.	2023800068
Experiment No.	4

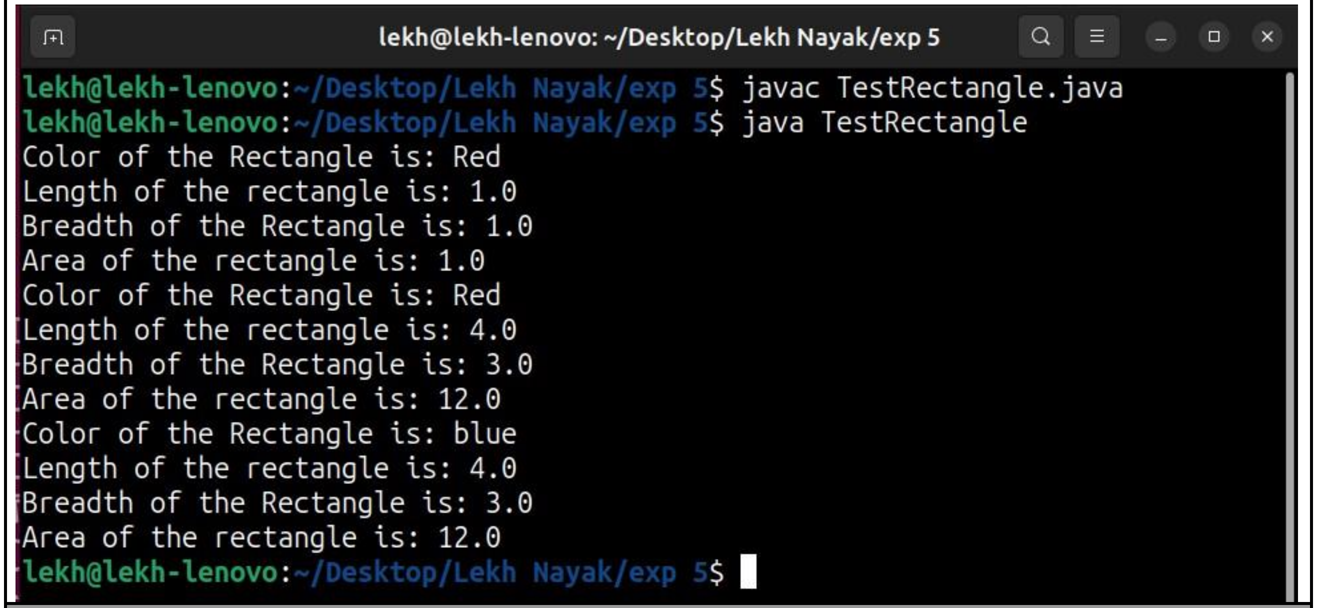
AIM:	**Implement a Program to demonstrate single, multilevel Inheritance**
Program 1	
PROBLEM STATEMENT :	Consider a class called Shape with member color of the shape. Consider a class called Rectangle, derived from the Shape class, which adds features length and breadth, add a method to print the rectangle’s details and find the area of the rectangle. Use appropriate access modifiers. Create a TestRectangle class to create objects of the rectangle class and test the methods.

PROGRAM:

```
import java.util.*;

// Define a Shape class
class Shape {
  private String color; // Private attribute to store color

  // Constructors
  Shape() {
    color = "Red"; // Default color is set to Red
  }

  Shape(String color) {
    this.color = color; // Initialize color with the provided value
  }

  // Getter method for color
  String getColor() {
    return color; // Return the color of the shape
  }
}

// Define a Rectangle class that extends Shape
class Rectangle extends Shape {
  private float length, breadth; // Private attributes to store dimensions

  // Constructors
  Rectangle() {
    length = breadth = 1; // Default dimensions set to 1x1
  }

  Rectangle(float length, float breadth) {
    this.length = length; // Initialize length
    this.breadth = breadth; // Initialize breadth
  }

  Rectangle(float length, float breadth, String color) {
    super(color); // Call superclass constructor to set color
    this.length = length; // Initialize length
    this.breadth = breadth; // Initialize breadth
  }

  // Method to calculate area of the rectangle
  float area() {
    return length * breadth; // Calculate and return area
  }

  // Method to display details of the rectangle
  void display() {
    // Print color, length, and breadth of the rectangle
    System.out.println("Color of the Rectangle is: " + getColor());
    System.out.println("Length of the rectangle is: " + length);
    System.out.println("Breadth of the Rectangle is: " + breadth);
  }
}

// Main class to test Rectangle functionality
public class TestRectangle {
  public static void main(String[] args) {
    // Create rectangle objects and display their details
    Rectangle r1 = new Rectangle();
    r1.display(); // Display details of the rectangle
    System.out.println("Area of the rectangle is: " + r1.area()); // Calculate
and print area

    Rectangle r2 = new Rectangle(4, 3);
    r2.display(); // Display details of the rectangle
    System.out.println("Area of the rectangle is: " + r2.area()); // Calculate
```

```
and print area

    Rectangle r3 = new Rectangle(4, 3, "blue");
    r3.display(); // Display details of the rectangle
    System.out.println("Area of the rectangle is: " + r3.area()); // Calculate
and print area
  }
}
```

RESULT:



lekh@lekh-lenovo: ~/Desktop/Lekh Nayak/exp 5

```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ javac TestRectangle.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ java TestRectangle
Color of the Rectangle is: Red
Length of the rectangle is: 1.0
Breadth of the Rectangle is: 1.0
Area of the rectangle is: 1.0
Color of the Rectangle is: Red
Length of the rectangle is: 4.0
Breadth of the Rectangle is: 3.0
Area of the rectangle is: 12.0
Color of the Rectangle is: blue
Length of the rectangle is: 4.0
Breadth of the Rectangle is: 3.0
Area of the rectangle is: 12.0
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$
```

Program 2

PROBLEM STATEMENT :	Define parent **class "Employee"** that has 3 private attributes String name, String id, int age. Employee has constructor with 3 arguments that set value of name, id, age. It also has getter and setter methods for all 3 private attributes. **Class "SalariedEmployee"** is a sub class of Employee and has 1 private attribute empSalary. "SalariedEmployee" can be permanent or on contract and has constructor SalariedEmployee(String name, String id, int age, double empSalary) to set the values. constructor SalariedEmployee must call the superclass constructor to set name, id, age and call setter method to set the salary. Employee salary is empSalary + 2000(allowance) if he is a permanent employee else Employee salary is empSalary (no allowance).

	Accept the details of 5 salaried employees and print details of the employee with highest salary. Hint: Use array of objects Create **class Tester** with main method
PROGRAM:	See code below

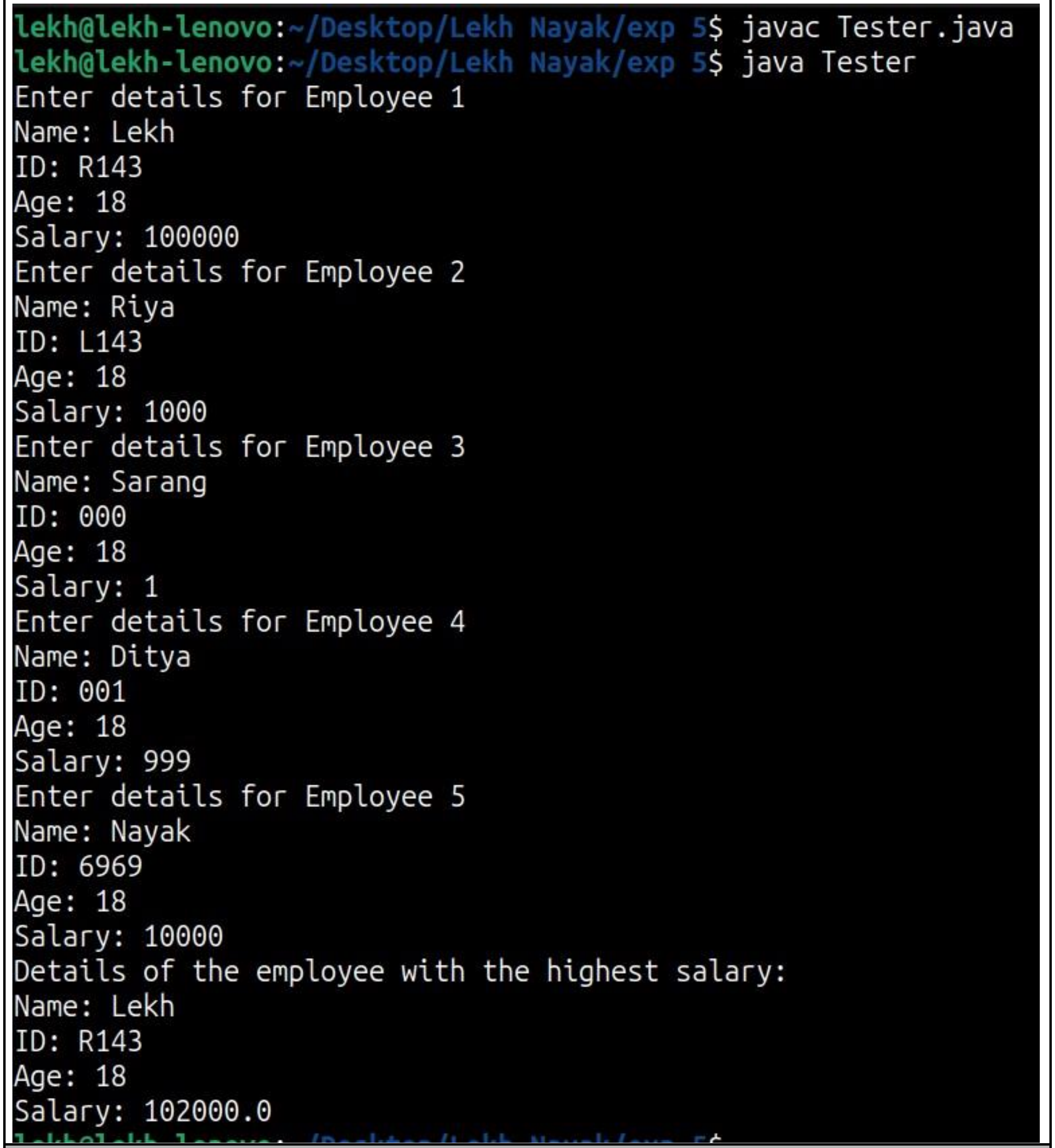
```
import java.util.*;

// Define parent class Employee
class Employee {
  private String name;
  private String id;
  private int age;

  // Constructor for Employee class
  Employee(String name, String id, int age) {
    this.name = name;
    this.id = id;
    this.age = age;
  }

  // Getter methods for private attributes of Employee class
  String getName() {
    return name;
  }

  String getId() {
    return id;
  }

  int getAge() {
    return age;
  }

  // Setter methods for private attributes of Employee class
  void setName(String name) {
    this.name = name;
  }

  void setId(String id) {
    this.id = id;
  }
```

```java
  void setAge(int age) {
    this.age = age;
  }
}

// Define subclass SalariedEmployee
class SalariedEmployee extends Employee {
  private double empSalary;

  // Constructor for SalariedEmployee class
  SalariedEmployee(String name, String id, int age, double empSalary) {
    // Call superclass constructor to set name, id, and age
    super(name, id, age);
    // Set the salary using setter method
    setEmpSalary(empSalary);
  }

  // Getter method for empSalary attribute
  double getEmpSalary() {
    return empSalary;
  }

  // Setter method for empSalary attribute
  void setEmpSalary(double empSalary) {
    this.empSalary = empSalary;
  }

  // Method to calculate total salary based on employment type
  double calculateTotalSalary() {
    // If permanent employee, add allowance, else return salary without allowance
    if (empSalary >= 0) {
      return empSalary + 2000;
    } else {
      return empSalary;
    }
  }
}
```

```
// Main class Tester
public class Tester {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    SalariedEmployee[] employees = new SalariedEmployee[5];

    // Accept details of 5 salaried employees
    for (int i = 0; i < 5; i++) {
      System.out.println("Enter details for Employee " + (i + 1));
      System.out.print("Name: ");
      String name = scanner.nextLine();
      System.out.print("ID: ");
      String id = scanner.nextLine();
      System.out.print("Age: ");
      int age = scanner.nextInt();
      System.out.print("Salary: ");
      double salary = scanner.nextDouble();
      scanner.nextLine(); // Consume newline character

      // Create instances of SalariedEmployee and store them in the array
      employees[i] = new SalariedEmployee(name, id, age, salary);
    }

    // Find employee with highest salary
    SalariedEmployee highestPaidEmployee = employees[0];
    for (int i = 1; i < 5; i++) {
      if (employees[i].calculateTotalSalary() >
highestPaidEmployee.calculateTotalSalary()) {
        highestPaidEmployee = employees[i];
      }
    }

    // Print details of the employee with highest salary
    System.out.println("Details of the employee with the highest salary:");
    System.out.println("Name: " + highestPaidEmployee.getName());
    System.out.println("ID: " + highestPaidEmployee.getId());
    System.out.println("Age: " + highestPaidEmployee.getAge());
    System.out.println("Salary: " +
highestPaidEmployee.calculateTotalSalary());
  }
```

	}

RESULT:



```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ javac Tester.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ java Tester
Enter details for Employee 1
Name: Lekh
ID: R143
Age: 18
Salary: 100000
Enter details for Employee 2
Name: Riya
ID: L143
Age: 18
Salary: 1000
Enter details for Employee 3
Name: Sarang
ID: 000
Age: 18
Salary: 1
Enter details for Employee 4
Name: Ditya
ID: 001
Age: 18
Salary: 999
Enter details for Employee 5
Name: Nayak
ID: 6969
Age: 18
Salary: 10000
Details of the employee with the highest salary:
Name: Lekh
ID: R143
Age: 18
Salary: 102000.0
```

Program 3

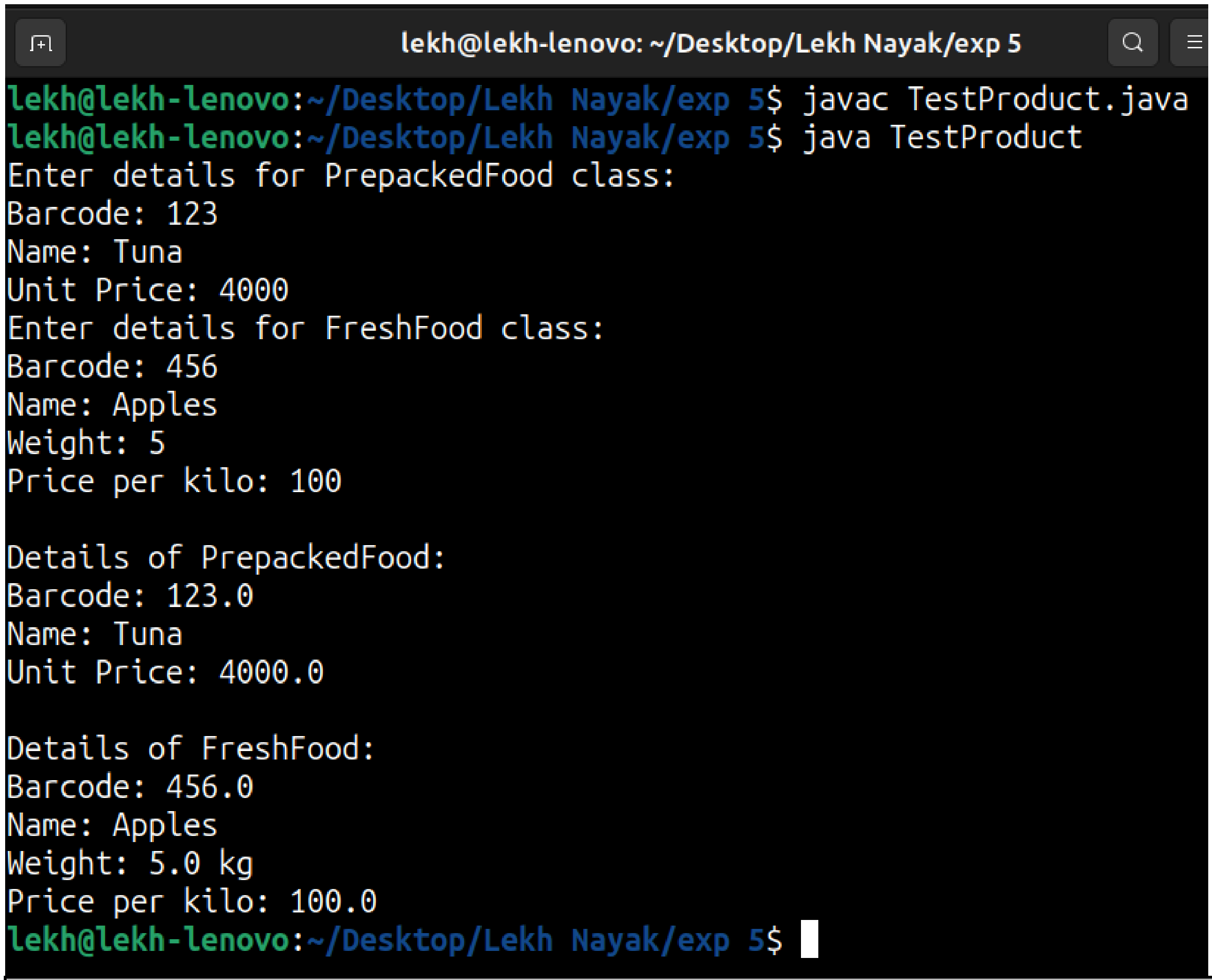
PROBLEM STATEMENT:	Consider a class Product with data members barcode and name of the product. Create the appropriate constructor and write getter methods for the

	individual data members. Derive 2 classes from Product, 1st class is PrepackedFood and 2nd class is FreshFood. the PrepackedFood class should contain the unit price and the FreshFood class should contain a weight and a price per kilo as data members. Create methods to read the details and print the details of a FreshFood and PrepackedFood product in the main class.

PROGRAM:

```
import java.util.*;

class Product {
  private double barcode;
  private String name;

  //parameterised constructor of class product
  Product(double barcode, String name) {
    this.barcode = barcode;
    this.name = name;
  }

  // getter methods for private attributes of Product class
  double getBarcode() {
    return barcode;
  }
  String getName() {
    return name;
  }
}

class PrepackedFood extends Product {
  private double unitPrice;

  // Parameterized constructor of PrepackedFood
  PrepackedFood(double barcode, String name, double unitPrice) {
    // Call superclass constructor to set barcode and name
    super(barcode, name);
    this.unitPrice = unitPrice;
  }
   // getter methods for private attributes of  class PrepackedFood
  double getUnitPrice() {
    return unitPrice;
  }
}

class FreshFood extends Product {
  private double weight;
  private double PricePerKilo;

  // Parameterized constructor of FreshFood
  FreshFood(double barcode, String name, double weight, double
PricePerKilo) {
    // Call superclass constructor to set barcode and name
    super(barcode, name);
    this.weight = weight;
    this.PricePerKilo = PricePerKilo;
  }

  // getter methods for private attributes of  class FreshFood
  double getWeight() {
    return weight;
  }
  double getPricePerKilo() {
    return PricePerKilo;
  }
}

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

      //taking user input for data in PrepackedFood class
      System.out.println("Enter details for PrepackedFood class: ");
      System.out.print("Barcode: ");
      double barcode1 = sc.nextDouble();
      sc.nextLine(); // Consume newline
      System.out.print("Name: ");
      String name1 = sc.nextLine();
      System.out.print("Unit Price: ");
      double unitPrice = sc.nextDouble();
      PrepackedFood prepackedFood = new PrepackedFood(barcode1,
name1, unitPrice);

      //taking user input for data in FreshFood class
      System.out.println("Enter details for FreshFood class: ");
      System.out.print("Barcode: ");
      double barcode2 = sc.nextDouble();
      sc.nextLine(); // Consume newline
      System.out.print("Name: ");
      String name2 = sc.nextLine();
      System.out.print("Weight: ");
      double weight = sc.nextDouble();
      System.out.print("Price per kilo: ");
      double pricePerKilo = sc.nextDouble();

      FreshFood freshFood = new FreshFood(barcode2, name2, weight,
pricePerKilo);

      // Printing details of PrepackedFood
      System.out.println("\nDetails of PrepackedFood:");
      System.out.println("Barcode: " + prepackedFood.getBarcode());
      System.out.println("Name: " + prepackedFood.getName());
      System.out.println("Unit Price: " + prepackedFood.getUnitPrice());

      // Printing details of FreshFood
      System.out.println("\nDetails of FreshFood:");
      System.out.println("Barcode: " + freshFood.getBarcode());
      System.out.println("Name: " + freshFood.getName());
      System.out.println("Weight: " + freshFood.getWeight() + " kg");
      System.out.println("Price per kilo: " + freshFood.getPricePerKilo());

      sc.close();
  }
}
```

RESULT:



```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ javac TestProduct.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ java TestProduct
Enter details for PrepackedFood class:
Barcode: 123
Name: Tuna
Unit Price: 4000
Enter details for FreshFood class:
Barcode: 456
Name: Apples
Weight: 5
Price per kilo: 100

Details of PrepackedFood:
Barcode: 123.0
Name: Tuna
Unit Price: 4000.0

Details of FreshFood:
Barcode: 456.0
Name: Apples
Weight: 5.0 kg
Price per kilo: 100.0
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$
```

Program 4	
PROBLEM STATEMENT:	Define class Production that has attributes String title, String director, String writer. Production class has 3 argument constructor that sets the values. It also has getter and setter methods and Overridden toString() of object class to display details of class. class Play is a sub class of Production with getter and setter methods and has an attribute int performances that is incremented every time a play happens. Add Overridden toString() of object class to display details of the class.

class Musical is a Play with songs. Musical object has all attributes of Play as well as String composer and String lyricist along with getter and setter methods. Override toString display all attributes of Musical object

In main create 3 objects of Play and 2 objects of Musical. Every time an object of Play or Musical is created, performances get incremented. Also add the number of seats booked for each play or musical.

Find the total box office collection, provided cost of 1 seat for Play is Rs 500(can be variable) and cost of 1 seat for Musical is Rs 800(can be variable)

Display total No. of performances as 5 and display the box office collection.

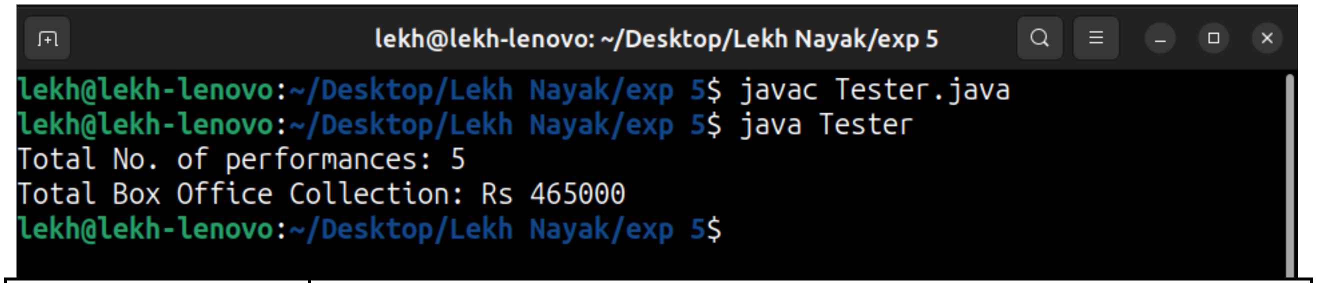
Create class Tester with main method

PROGRAM:

```
import java.util.*;

//Creating a class named Production
class Production {
  private String title;
  private String director;
  private String writer;

  //Creating a paramterized constructor for Production class to initialize the attributes
  Production(String title, String director, String writer) {
    this.title = title;
    this.director = director;
    this.writer = writer;
  }

  //Getter methods for private attributes of Production class
  String getTitle() {
    return title;
```

```java
  }
  String getDirector() {
    return director;
  }
  String getWriter() {
    return writer;
  }

  //Setter methods for private attributes of production class
  void setTitle(String title) {
    this.title = title;
  }
  void setDirector(String director) {
    this.director = director;
  }
  void setWriter(String writer) {
    this.writer = writer;
  }

  //toString method
  public String toString() {
      return "Title: " + title + ", Director: " + director + ", Writer: " + writer;
  }
}

class Play extends Production {
  private int performances;
  private int seatsBooked;

  Play(String title, String director, String writer, int performances, int
seatsBooked) {
        super(title, director, writer);
        this.performances = performances;
        this.seatsBooked = seatsBooked;
  }

  //getter methods for private attributes in Play class
  int getPerformances() {
    return performances;
  }
  int getSeatsBooked() {
    return seatsBooked;
  }

  //setter methods for private attributes in play class
  void setPerformances(int performances) {
      this.performances = performances;
  }

  public void setSeatsBooked(int seatsBooked) {
      this.seatsBooked = seatsBooked;
  }

  // Override toString method
  public String toString() {
      return super.toString() + ", Performances: " + performances + ", Seats
Booked: " + seatsBooked;
  }
}

class Musical extends Play {
  private String composer;
  private String lyricist;

  Musical(String title, String director, String writer, String composer, String
lyricist, int performances, int seatsBooked) {
    super(title, director, writer, performances, seatsBooked);
    this.composer = composer;
    this.lyricist = lyricist;
    }

    //getter methods for private attributes in Musical production
    String getComposer() {
        return composer;
    }
    String getLyricist() {
        return lyricist;
    }

    //setter methods for private attributes in Musical production
    void setComposer(String composer) {
        this.composer = composer;
    }
    void setLyricist(String lyricist) {
        this.lyricist = lyricist;
    }

    // Override toString method
    public String toString() {
        return super.toString() + ", Composer: " + composer + ", Lyricist: " +
lyricist;
    }
}

public class Tester {
    public static void main(String[] args) {
        // Create objects
        Play play1 = new Play("Play 1", "Director 1", "Writer 1", 0, 0);
        Play play2 = new Play("Play 2", "Director 2", "Writer 2", 0, 0);
        Play play3 = new Play("Play 3", "Director 3", "Writer 3", 0, 0);

        Musical musical1 = new Musical("Musical 1", "Director 4", "Writer
4", "Composer 1", "Lyricist 1", 0, 0);
        Musical musical2 = new Musical("Musical 2", "Director 5", "Writer
5", "Composer 2", "Lyricist 2", 0, 0);

        // Increment performances
        play1.setPerformances(play1.getPerformances() + 1);
        play2.setPerformances(play2.getPerformances() + 1);
        play3.setPerformances(play3.getPerformances() + 1);
        musical1.setPerformances(musical1.getPerformances() + 1);
        musical2.setPerformances(musical2.getPerformances() + 1);

        // Book seats
        play1.setSeatsBooked(100);
        play2.setSeatsBooked(150);
        play3.setSeatsBooked(200);
        musical1.setSeatsBooked(120);
        musical2.setSeatsBooked(180);
        // Calculate box office collection
        int seatCostPlay = 500;
        int seatCostMusical = 800;

        int totalCollection = (play1.getSeatsBooked() +
play2.getSeatsBooked() + play3.getSeatsBooked()) * seatCostPlay +
                        (musical1.getSeatsBooked() +
musical2.getSeatsBooked()) * seatCostMusical;

        // Display total performances and box office collection
        int totalPerformances = play1.getPerformances() +
play2.getPerformances() + play3.getPerformances() +
                        musical1.getPerformances() +
musical2.getPerformances();

        System.out.println("Total No. of performances: " +
totalPerformances);
        System.out.println("Total Box Office Collection: Rs " +
totalCollection);
    }
}
```

RESULT:



```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ javac Tester.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$ java Tester
Total No. of performances: 5
Total Box Office Collection: Rs 465000
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 5$
```

CONCLUSION:	In this experiment i learnt how to implement **Programs to demonstrate single, multilevel Inheritance** to solve real life based problems .