Name	Lekh Sanatan Nayak
UID no.	2023800068
Experiment No.	6

AIM: Implement a Program to demonstrate method overriding

Program 1

PROBLEM STATEMENT :

Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks are given below.

Period	SBI Interest Rate (Rates in % per annum)
	<Rs. 2 Cr
7–14 Days	3.00
15 –30 Days	3.00
31-45 Days	3.00
46 -90 Days	4.05
91–120 Days	4.10
121-180 Days	4.10

Period	ICICI Interest Rate (Rates in % per annum)
	<Rs. 2 Cr
7–14 Days	3.10
15 –30 Days	3.20

31-45 Days	3.50
46 -90 Days	4.50
91–120 Days	4.70
121-180 Days	4.90

Period	AXIS Interest Rate (Rates in % per annum)
	<Rs. 2 Cr
7–14 Days	3.15
15 –30 Days	3.15
31-45 Days	3.45
46 -90 Days	4.05
91–120 Days	4.70
121-180 Days	5.00

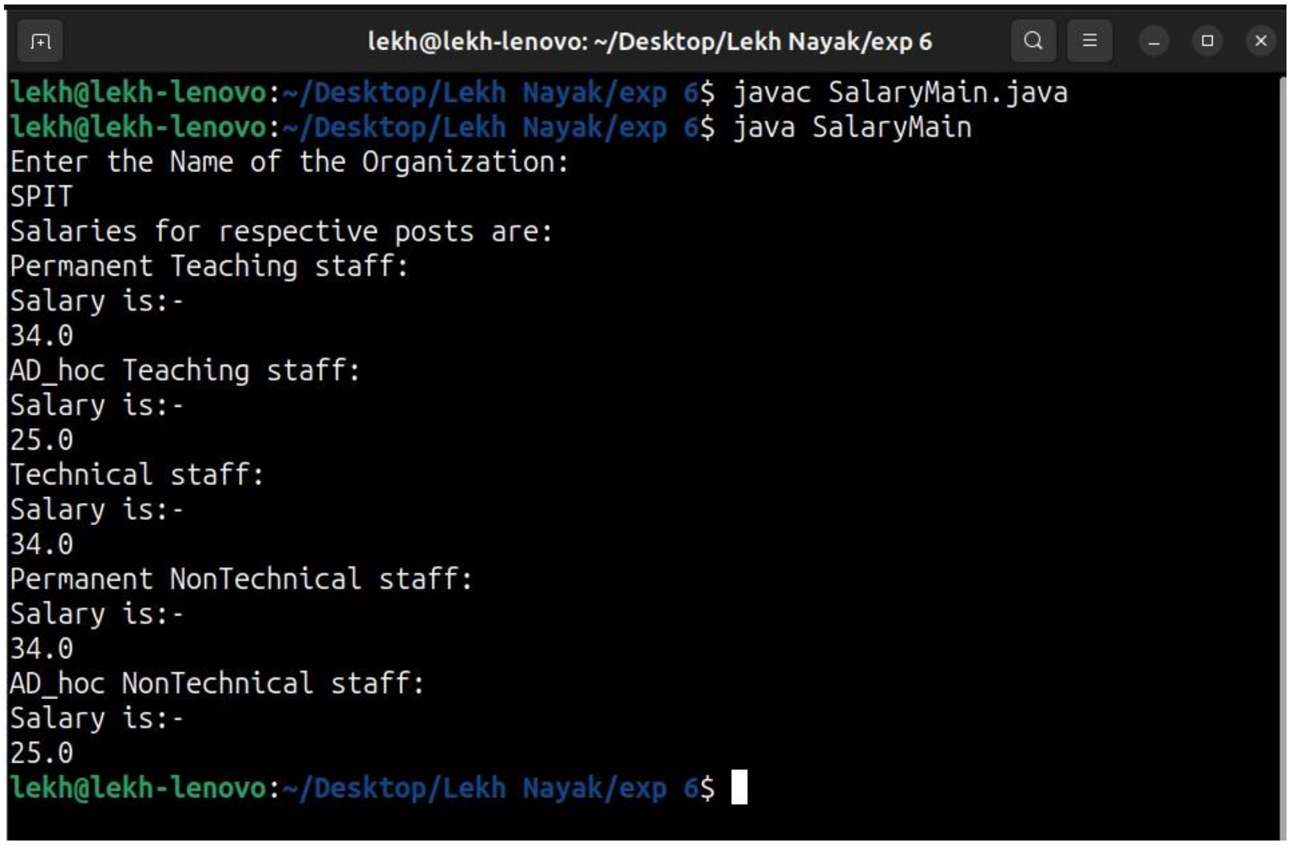
Aayush has deposited Rs. 10000 in SBI Bank, Rs. 12500 in ICICI Bank, and Rs. 20000 in AXIS bank respectively for a particular month.
You need to print the money he will get by applying the rate of interest as per the bank and days.
Create a class 'Bank' with a method 'get_rate_of_interest' which returns 2%.
Make three subclasses named SBI_Bank, 'ICICI_Bank' and 'AXIS_bank' with a method with the same name 'get_rate_of_interest' which returns the rate of interest.
Also, give the final amount Ayush will get from that particular bank by applying the rate of interest and period. Use Calendar Class to count the number of days and amount he will get after maturity with the date of Maturity, if he deposits today.

Note:
1. Use compound interest
2. Get time period from the user
3. Solve using method overriding

PROGRAM:	See code below

```
import java.util.*;

// Parent class representing an Employee
class Employee {
  double basicSalary = 25;  // Default basic salary

  // Method to calculate salary (basic salary only)
  double Salary() {
    return basicSalary;
  }

  // Method to display salary
  void Display() {
    System.out.println("Salary is:-");
    System.out.println(basicSalary);
  }
}

// Subclass representing Teaching staff
class Teaching extends Employee{
  // Method to calculate salary (inherited from superclass)
  double Salary() {
    return super.Salary();
  }

  // Method to display salary (inherited from superclass)
  void Display() {
    super.Display();
  }
}

// Subclass representing Permanent Teaching staff
class PermanentTeaching extends Teaching {
  double totalSalary; // Total salary including DA and HRA

  // Method to calculate salary (including DA and HRA)
  double Salary() {
    double DA = (28.0/100) * basicSalary;
    double HRA = (8.0/100) * basicSalary;
    totalSalary = basicSalary + DA + HRA;
    return totalSalary;
  }

  // Method to display salary
  void Display() {
    System.out.println("Salary is:-");
    System.out.println(totalSalary);
  }
}

// Subclass representing Ad-hoc Teaching staff
class AD_hocTeaching extends Teaching {
  // Method to calculate salary (inherited from superclass)
  double Salary() {
    return super.Salary();
  }

  // Method to display salary (inherited from superclass)
  void Display() {
    super.Display();
  }
}

// Subclass representing Non-Teaching staff
class NonTeaching extends Employee {
  // Method to calculate salary (inherited from superclass)
  double Salary() {
    return super.Salary();
  }

  // Method to display salary (inherited from superclass)
  void Display() {
    super.Display();
  }
}

// Subclass representing Technical staff
class Technical extends NonTeaching {
  double totalSalary; // Total salary including DA and HRA
  // Method to calculate salary (including DA and HRA)
  double Salary() {
    double DA = (28.0/100) * basicSalary;
    double HRA = (8.0/100) * basicSalary;
    totalSalary = basicSalary + DA + HRA;
    return totalSalary;
  }

  // Method to display salary
  void Display() {
    System.out.println("Salary is:-");
    System.out.println(totalSalary);
  }
}

// Subclass representing Non-Technical staff
class NonTechnical extends NonTeaching {
  // Method to calculate salary (inherited from superclass)
  double Salary() {
    return super.Salary();
  }

  // Method to display salary (inherited from superclass)
  void Display() {
    super.Display();
  }
}

// Subclass representing Permanent Non-Technical staff
class PermanetNonTechnical extends NonTechnical {
  double totalSalary; // Total salary including DA and HRA

  // Method to calculate salary (including DA and HRA)
  double Salary() {
    double DA = (28.0/100) * basicSalary;
    double HRA = (8.0/100) * basicSalary;
    totalSalary = basicSalary + DA + HRA;
    return totalSalary;
  }
  // Method to display salary
  void Display() {
    System.out.println("Salary is:-");
    System.out.println(totalSalary);
  }
}

// Subclass representing Ad-hoc Non-Technical staff
class AD_hocNonTechnical extends NonTechnical {
  // Method to calculate salary (inherited from superclass)
  double Salary() {
    return super.Salary();
  }

  // Method to display salary (inherited from superclass)
  void Display() {
    super.Display();
  }
}

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

    // Prompting user to enter the Name of the Organization
    System.out.println("Enter the Name of the Organization:");
    String name = sc.nextLine();

    // Creating instances of different employee classes
    PermanentTeaching sal1 = new PermanentTeaching();
    AD_hocTeaching sal2 = new AD_hocTeaching();
    Technical sal3 = new Technical();
    PermanetNonTechnical sal4 = new PermanetNonTechnical();
    AD_hocNonTechnical sal5 = new AD_hocNonTechnical();

    // Calculating and displaying salaries for each employee category
    sal1.Salary();
    System.out.println("Salaries for respective posts are:");
    System.out.println("Permanent Teaching staff:");
    sal1.Display();

    System.out.println("AD_hoc Teaching staff:");
    sal2.Display();

    sal3.Salary();
    System.out.println("Technical staff:");
    sal3.Display();

    sal4.Salary();
    System.out.println("Permanent NonTechnical staff:");
    sal4.Display();

    System.out.println("AD_hoc NonTechnical staff:");
    sal5.Display();

    sc.close(); // Closing scanner
  }
}
```

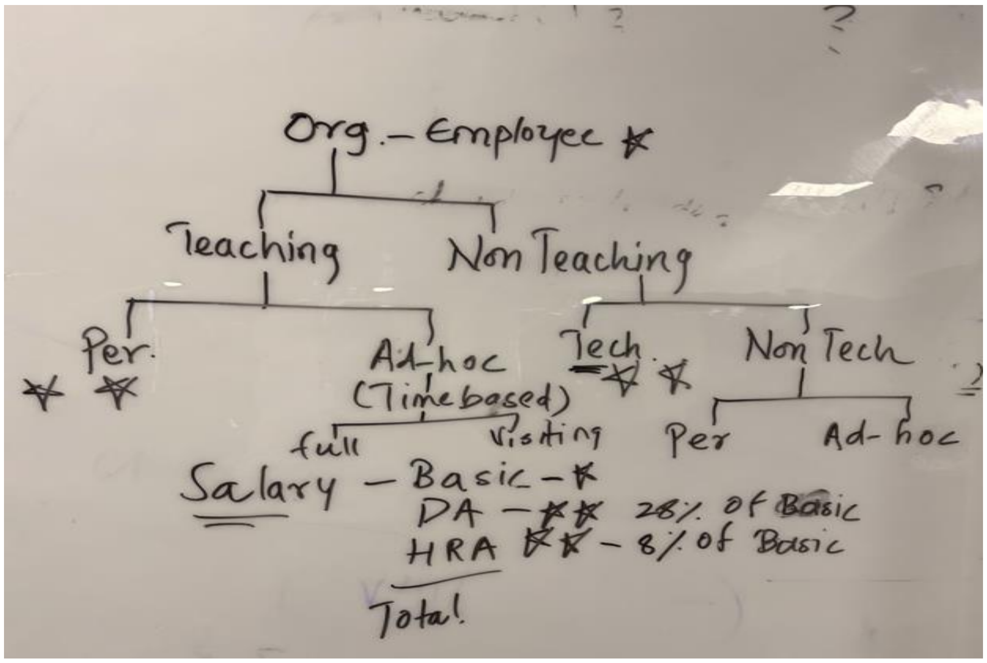
RESULT:



```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ javac SalaryMain.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ java SalaryMain
Enter the Name of the Organization:
SPIT
Salaries for respective posts are:
Permanent Teaching staff:
Salary is:-
34.0
AD_hoc Teaching staff:
Salary is:-
25.0
Technical staff:
Salary is:-
34.0
Permanent NonTechnical staff:
Salary is:-
34.0
AD_hoc NonTechnical staff:
Salary is:-
25.0
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$
```

Program 2

PROBLEM STATEMENT :	1. Write a program to display the salary of an employee working in an organization. Consider the following hierarchy and use the given formulae to calculate the DA and HRA based on the basic salary.

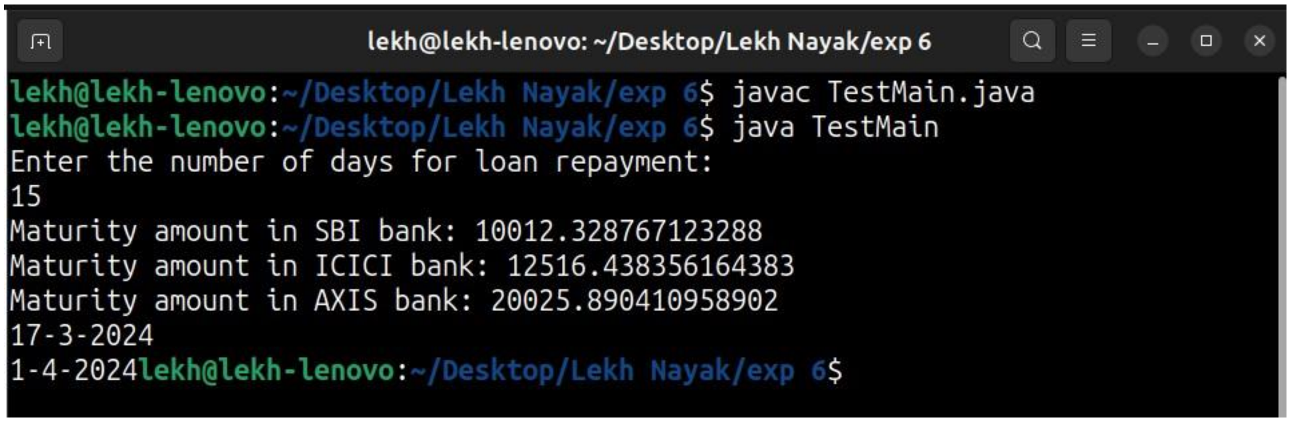


PROGRAM:

```
import java.util.*;

// Parent class representing a Bank
class Bank {
    // Method to get the default rate of interest for all banks
    double get_rate_of_interest () {
      return 2.0;
    }
}

// Subclass representing SBI Bank
class SBI_Bank extends Bank {
    // Method to get the rate of interest for SBI Bank
    double get_rate_of_interest () {
      return super.get_rate_of_interest (); // Using superclass method to get default interest rate
    }
}

// Subclass representing ICICI Bank
class ICICI_Bank extends Bank {
    // Method to get the rate of interest for ICICI Bank
    double get_rate_of_interest () {
      return super.get_rate_of_interest (); // Using superclass method to get
default interest rate
    }
}

// Subclass representing AXIS Bank
class AXIS_Bank extends Bank {
    // Method to get the rate of interest for AXIS Bank
    double get_rate_of_interest () {
      return super.get_rate_of_interest (); // Using superclass method to get
default interest rate
    }
}

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

        // Prompting user to enter the number of days for loan repayment
        System.out.println("Enter the number of days for loan repayment: ");
        int days = sc.nextInt();

        // Initial amounts for different banks
        double amount_SBI = 10000;
        double amount_ICICI = 12500;
        double amount_AXIS = 20000;

        // Creating instances of different bank classes
        SBI_Bank sbi_bank = new SBI_Bank();
        ICICI_Bank icici_bank = new ICICI_Bank();
        AXIS_Bank axis_bank = new AXIS_Bank();

        // Calculating maturity amount for each bank using respective interest
rates
        double maturity_amount_SBI = calculateMaturity(amount_SBI,
getInterestRateSBI(days), days);
        double maturity_amount_ICICI = calculateMaturity(amount_ICICI,
getInterestRateICICI(days), days);
        double maturity_amount_AXIS = calculateMaturity(amount_AXIS,
getInterestRateAXIS(days), days);

        // Printing maturity amounts for each bank
        System.out.println("Maturity amount in SBI bank: " +
maturity_amount_SBI);
        System.out.println("Maturity amount in ICICI bank: " +
maturity_amount_ICICI);
        System.out.println("Maturity amount in AXIS bank: " +
maturity_amount_AXIS);

        sc.close(); // Closing scanner

        // Printing current date and date after 15 days
        Calendar cal;
        cal = Calendar.getInstance();
        System.out.print(cal.get(Calendar.DATE)+ "-");
        System.out.print(cal.get(Calendar.MONTH)+ 1 +"-");
        System.out.print(cal.get(Calendar.YEAR));
        cal.add(Calendar.DAY_OF_YEAR, 15);
        System.out.println();
        System.out.print(cal.get(Calendar.DATE)+ "-");
        System.out.print(cal.get(Calendar.MONTH)+ 1 +"-");
        System.out.print(cal.get(Calendar.YEAR));
    }

    // Method to calculate maturity amount based on principal, interest rate,
and days
    static double calculateMaturity(double principal, double rate, int days) {
        double interest = rate;
        return principal * (1 + (interest / 100) * (days / 365.0));
    }

    // Method to get interest rate for SBI Bank based on number of days
    static double getInterestRateSBI(int days) {
        if (days >= 7 && days <= 14) {
            return 3.0;
        } else if (days >= 15 && days <= 30) {
            return 3.0;
        } else if (days >= 31 && days <= 45) {
            return 3.0;
    } else if (days >= 46 && days <= 90) {
        return 4.05;
    } else if (days >= 91 && days <= 120) {
        return 4.10;
    } else if (days >= 121 && days <= 180) {
        return 4.10;
    } else {
        return 2.0; // Default rate if days don't fall in any category
    }
}

// Method to get interest rate for ICICI Bank based on number of days
static double getInterestRateICICI(int days) {
    if (days >= 7 && days <= 14) {
        return 3.10;
    } else if (days >= 15 && days <= 30) {
        return 3.20;
    } else if (days >= 31 && days <= 45) {
        return 3.50;
    } else if (days >= 46 && days <= 90) {
        return 4.50;
    } else if (days >= 91 && days <= 120) {
        return 4.70;
    } else if (days >= 121 && days <= 180) {
        return 4.90;
    } else {
        return 2.0; // Default rate if days don't fall in any category
    }
}

// Method to get interest rate for AXIS Bank based on number of days
static double getInterestRateAXIS(int days) {
    if (days >= 7 && days <= 14) {
        return 3.15;
    } else if (days >= 15 && days <= 30) {
        return 3.15;
    } else if (days >= 31 && days <= 45) {
        return 3.45;
    } else if (days >= 46 && days <= 90) {
        return 4.05;
```

	```
	    } else if (days >= 91 && days <= 120) {
	      return 4.70;
	    } else if (days >= 121 && days <= 180) {
	      return 5.00;
	    } else {
	      return 2.0; // Default rate if days don't fall in any category
	    }
	  }
	}
	```

RESULT:



```
lekh@lekh-lenovo: ~/Desktop/Lekh Nayak/exp 6
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ javac TestMain.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ java TestMain
Enter the number of days for loan repayment:
15
Maturity amount in SBI bank: 10012.328767123288
Maturity amount in ICICI bank: 12516.438356164383
Maturity amount in AXIS bank: 20025.890410958902
17-3-2024
1-4-2024lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$
```

Program 3

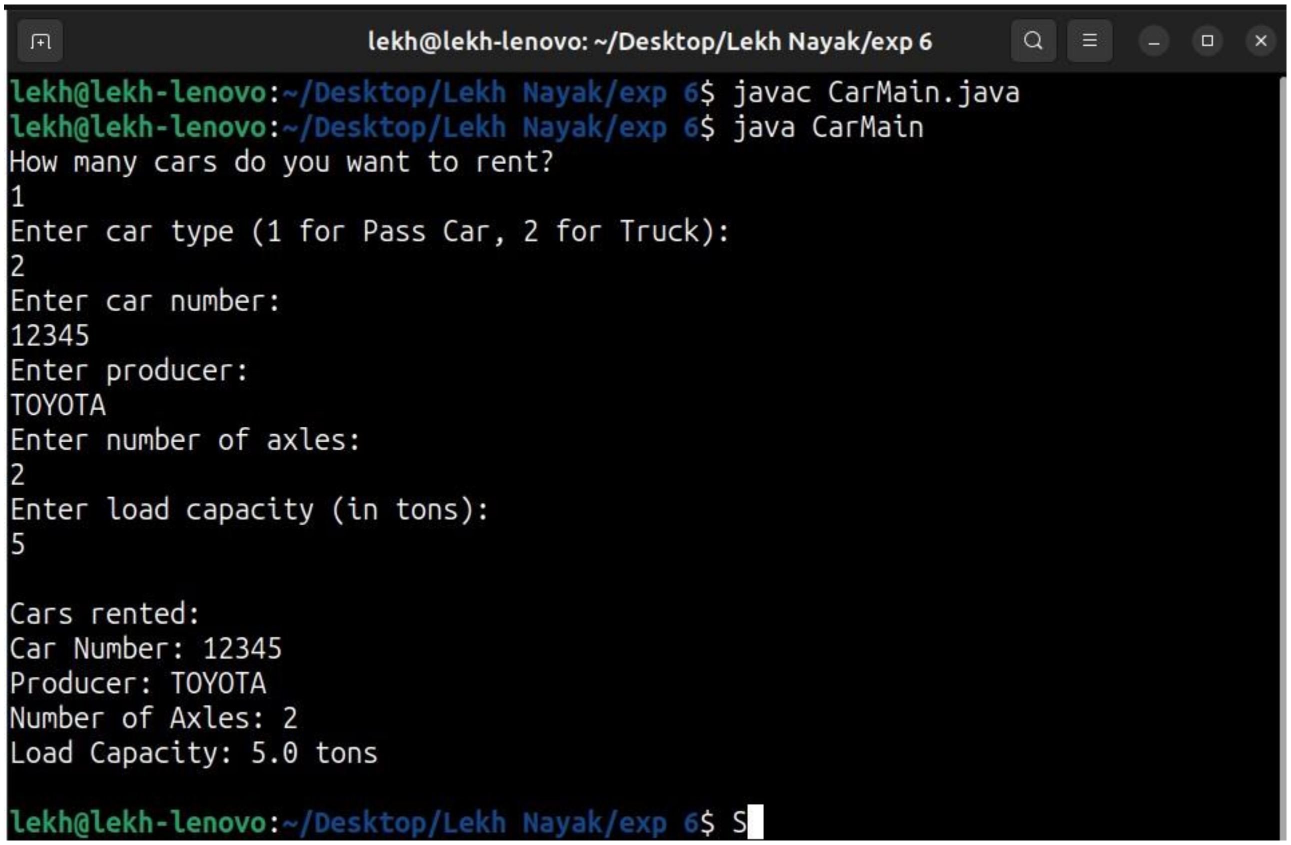
PROBLEM STATEMENT:	Consider a class called Car with data car_no and producer (both private). Write appropriate constructors and setter/getter methods. Add a method called display(). Derive two classes PassCar and Truck from Car. PassCar has private data passCarType and sunRoof(boolean). Add appropriate constructors and setter/getter methods. Override method display(). Class Truck has data members numberAxles, loadCapacity(private). Add constructors and setter/getter methods. Override method display(). Assume a car rental company which has two types of cars. Write a main class in which you would create an array of Cars (Parent type). Ask the user which type of car he wants and appropriately assign each car reference to the appropriate object (truck/passcar). Call the display method once the object is assigned clearly demonstrating runtime polymorphism.

PROGRAM:

```
import java.util.*;

// Parent class Car
class Car {
    // Private member variables to store car number and producer
    private String carNo;
    private String producer;

    // Constructor to initialize Car object with car number and producer
    public Car(String carNo, String producer) {
        this.carNo = carNo;
        this.producer = producer;
    }

    // Getter method for retrieving car number
    public String getCarNo() {
        return carNo;
    }

    // Setter method for setting car number
    public void setCarNo(String carNo) {
        this.carNo = carNo;
    }

    // Getter method for retrieving producer
    public String getProducer() {
        return producer;
    }

    // Setter method for setting producer
    public void setProducer(String producer) {
        this.producer = producer;
    }

    // Method to display car details
    public void display() {
        System.out.println("Car Number: " + carNo);
        System.out.println("Producer: " + producer);
    }
}
```

```
// Child class PassCar inheriting from Car
class PassCar extends Car {
    // Additional member variables for passenger car: passCarType and
sunRoof
    private String passCarType;
    private boolean sunRoof;

    // Constructor to initialize PassCar object with car number, producer,
passCarType, and sunRoof
    public PassCar(String carNo, String producer, String passCarType,
boolean sunRoof) {
        // Call the superclass constructor to initialize car number and producer
        super(carNo, producer);
        // Initialize PassCar specific variables
        this.passCarType = passCarType;
        this.sunRoof = sunRoof;
    }

    // Getter method for retrieving passenger car type
    public String getPassCarType() {
        return passCarType;
    }

    // Setter method for setting passenger car type
    public void setPassCarType(String passCarType) {
        this.passCarType = passCarType;
    }

    // Method to check if the passenger car has sunroof
    public boolean hasSunRoof() {
        return sunRoof;
    }

    // Setter method for setting sunroof status
    public void setSunRoof(boolean sunRoof) {
        this.sunRoof = sunRoof;
    }

    // Override display method to show passenger car details
    @Override
    public void display() {
        super.display(); // Call display method of superclass (Car)
        System.out.println("Pass Car Type: " + passCarType);
        System.out.print("Sun Roof: ");
        if (sunRoof) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }
    }
}

// Child class Truck inheriting from Car
class Truck extends Car {
    // Additional member variables for truck: numberAxles and loadCapacity
    private int numberAxles;
    private double loadCapacity;

    // Constructor to initialize Truck object with car number, producer,
numberAxles, and loadCapacity
    public Truck(String carNo, String producer, int numberAxles, double
loadCapacity) {
        // Call the superclass constructor to initialize car number and producer
        super(carNo, producer);
        // Initialize Truck specific variables
        this.numberAxles = numberAxles;
        this.loadCapacity = loadCapacity;
    }

    // Getter method for retrieving number of axles
    public int getNumberAxles() {
        return numberAxles;
    }

    // Setter method for setting number of axles
    public void setNumberAxles(int numberAxles) {
        this.numberAxles = numberAxles;
    }
    // Getter method for retrieving load capacity
    public double getLoadCapacity() {
        return loadCapacity;
    }

    // Setter method for setting load capacity
    public void setLoadCapacity(double loadCapacity) {
        this.loadCapacity = loadCapacity;
    }

    // Override display method to show truck details
    @Override
    public void display() {
        super.display(); // Call display method of superclass (Car)
        System.out.println("Number of Axles: " + numberAxles);
        System.out.println("Load Capacity: " + loadCapacity + " tons");
    }
}

// Main class
public class CarMain {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("How many cars do you want to rent?");
        int numOfCars = scanner.nextInt(); // Read the number of cars from
user
        scanner.nextLine(); // Consume newline character

        // Create an array to store Car objects
        Car[] cars = new Car[numOfCars];

        // Loop to input details of each car
        for (int i = 0; i < numOfCars; i++) {
            System.out.println("Enter car type (1 for Pass Car, 2 for Truck):");
            int carType = scanner.nextInt(); // Read car type from user
            scanner.nextLine(); // Consume newline character

            System.out.println("Enter car number:");
            String carNo = scanner.nextLine(); // Read car number from user
            System.out.println("Enter producer:");
            String producer = scanner.nextLine(); // Read producer from user

            if (carType == 1) { // If car type is Passenger Car
                System.out.println("Enter pass car type:");
                String passCarType = scanner.nextLine(); // Read passenger car
type from user

                System.out.println("Does it have a sunroof? (true/false):");
                boolean sunRoof = scanner.nextBoolean(); // Read sunroof status
from user

                // Create PassCar object and store it in the array
                cars[i] = new PassCar(carNo, producer, passCarType, sunRoof);
            } else if (carType == 2) { // If car type is Truck
                System.out.println("Enter number of axles:");
                int numberAxles = scanner.nextInt(); // Read number of axles
from user

                System.out.println("Enter load capacity (in tons):");
                double loadCapacity = scanner.nextDouble(); // Read load
capacity from user

                // Create Truck object and store it in the array
                cars[i] = new Truck(carNo, producer, numberAxles,
loadCapacity);
            }
        }

        // Displaying details of all rented cars
        System.out.println("\nCars rented:");
        for (Car car : cars) {
            car.display(); // Call display method of Car (dynamic binding)
            System.out.println();
        }

        scanner.close(); // Close the scanner object
    }
}
```

RESULT:



lekh@lekh-lenovo: ~/Desktop/Lekh Nayak/exp 6

```
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ javac CarMain.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ java CarMain
How many cars do you want to rent?
1
Enter car type (1 for Pass Car, 2 for Truck):
2
Enter car number:
12345
Enter producer:
TOYOTA
Enter number of axles:
2
Enter load capacity (in tons):
5

Cars rented:
Car Number: 12345
Producer: TOYOTA
Number of Axles: 2
Load Capacity: 5.0 tons

lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 6$ S
```

CONCLUSION:	In this experiment, I learnt how to use method overriding to solve real-life situations