BSc. Artificial Intelligence & Data Science

Level 05

CM 2601

OBJECT ORIENTED DEVELOPMENT

Coursework

BHANUKA WADUGE

IIT ID : 20201006

RGU ID : 2017920

Question 01

1. Characteristics of the OO design paradigm
    a. Classes

    A class is like a blueprint of data member and functions and object is an instance of class.

    1. Instance variables

    An instance variable is a variable which is declared in a class but outside of constructors, methods, or blocks. Access modifiers can be given to the instance variable

```
1
2  public class Account {
3
4      //instance variables
5      private String name;
6      private int accountNum;
7      private double balance;
8
9      //Account constructor
10     public Account(String initName, int initId, double initBalance){
11
12         name = initName;
13         accountNum = initId;
14         balance = initBalance;
15
16     }
17
18     public double getBalance(){
19         return balance;
20     }
21
22
23 }
24
```

    2. Instance methods

    Instance method are methods which require an object of its class to be created before it can be called. To invoke an instance method, we have to create an Object of the class in within which it defined

```
10    class Test
11    {
12        static int add()     // Static Method
13        {
14            int x = 10, y = 20;      // Local varaible
15            int z = x+y;
16            return z;
17        }
18        void disp()      // Instance Method
19        {
20            int a = add();      // calling Static Method
21            System.out.println("Addition = "+a);
22        }
23    }
```

3. static variables

Static variables are created when the program starts and destroyed when the program stops. Visibility is similar to instance variables. They are stored in static memory.

```
1  public class Static_Block {
2
3      static String str = "";
4      static
5      {
6          System.out.println("I am first executing...");
7          str = "Hello";
8      }
9      public static void main(String args[])
10     {
11         System.out.println("I am in Main now...");
12         System.out.println("The value of str is : " + str + " World");
13     }
14 }
```

4. static methods

Methods in Java that can be called without creating an object of class. They are referenced by the class name itself or reference to the Object of that class.

```
10  class Test
11  {
12      static int add()     // Static Method
13      {
14          int x = 10, y = 20;      // Local varaible
15          int z = x+y;
16          return z;
17      }
18      void disp()      // Instance Method
19      {
20          int a = add();      // calling Static Method
21          System.out.println("Addition = "+a);
22      }
23  }
```

5. Constructors

A Java constructor is special method that is called when an object is instantiated. In other words, when you use the new keyword. The purpose of a Java constructor is to initializes the newly created object before it is used.

```java

public class Account {

    //instance variables
    private String name;
    private int accountNum;
    private double balance;

    //Account constructor
    public Account(String initName, int initId, double initBalance){

        name = initName;
        accountNum = initId;
        balance = initBalance;

    }

    public double getBalance(){
        return balance;
    }


}

```

6. Getters and setters

Getters and setters are used to protect your data, particularly when creating classes. For each instance variable, a getter method returns its value while a setter method sets or updates its value.

```java
public class Tutorial2 {

        private int number;


        public Tutorial2(){

        }

        public Tutorial2(int num){
            number = num;
        }

        public void message(){
            System.out.printf("The number you entered was %d", number);
        }

        public void setNumber(int num){
            number = num;
        }

        public void getNumber(){

        }
```

b. Objects

Object is a member (also called an instance) of a Java class. Each object has an identity, a behavior and a state. The state of an object is stored in fields (variables), while methods (functions) display the object's behavior. Objects are created at runtime from templates, which are also known as classes.

c. OOP Concepts

1. Abstraction

In Java, Data Abstraction is defined as the process of reducing the object to its essence so that only the necessary characteristics are exposed to the users.

2. Encapsulation

It describes the idea of bundling data and methods that work on that data within one unit,.This concept is also often used to hide the internal representation, or state, of an object from the outside.

3. Inheritance

Inheritance is a concept that acquires the properties from one class to other classes; for example, the relationship between father and son. A class can inherit attributes and methods from another class. The class that inherits the properties is known as the sub-class or the child class

4. Polymorphism

Polymorphism in Java is the ability of an object to take many forms. To simply put, polymorphism in java allows us to perform the same action in many different ways

5. Abstract classes and abstract method

Abstract class: is a restricted class that cannot be used to create objects .To access it, it must be inherited from another class
Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass
.

6. Interface

It is an abstract type that is used to specify a behavior that classes must implement. They are similar to protocols.

Question 02

1. Different types of relationships between classes in the UML class diagram

- **Association** - Association is a broad term that encompasses just about any logical connection or relationship between classes
- **Directed Association** - Directed Association refers to a directional relationship represented by a line with an arrowhead
- **Reflexive Association** - This occurs when a class may have multiple functions or responsibilities.
- **Multiplicity** - Multiplicity is the active logical association when the cardinality of a class in relation to another is being depicted.
- **Aggregation** - It refers to the formation of a particular class as a result of one class being aggregated or built as a collection.
- **Composition** - The composition relationship is very similar to the aggregation relationship. With only difference being its key purpose of emphasizing the dependence of the contained class to the life cycle of the container class.
- **Inheritance/Generalization** - It refers to a type relationship wherein one associated class is a child of another by virtue of assuming the same functionalities of the parent class
- **Realization** - denotes the implementation of the functionality defined in one class by another class. To show the relationship in UML, a broken line with an unfilled solid arrowhead is drawn from the class that defines the functionality of the class that implements the function

Question 03

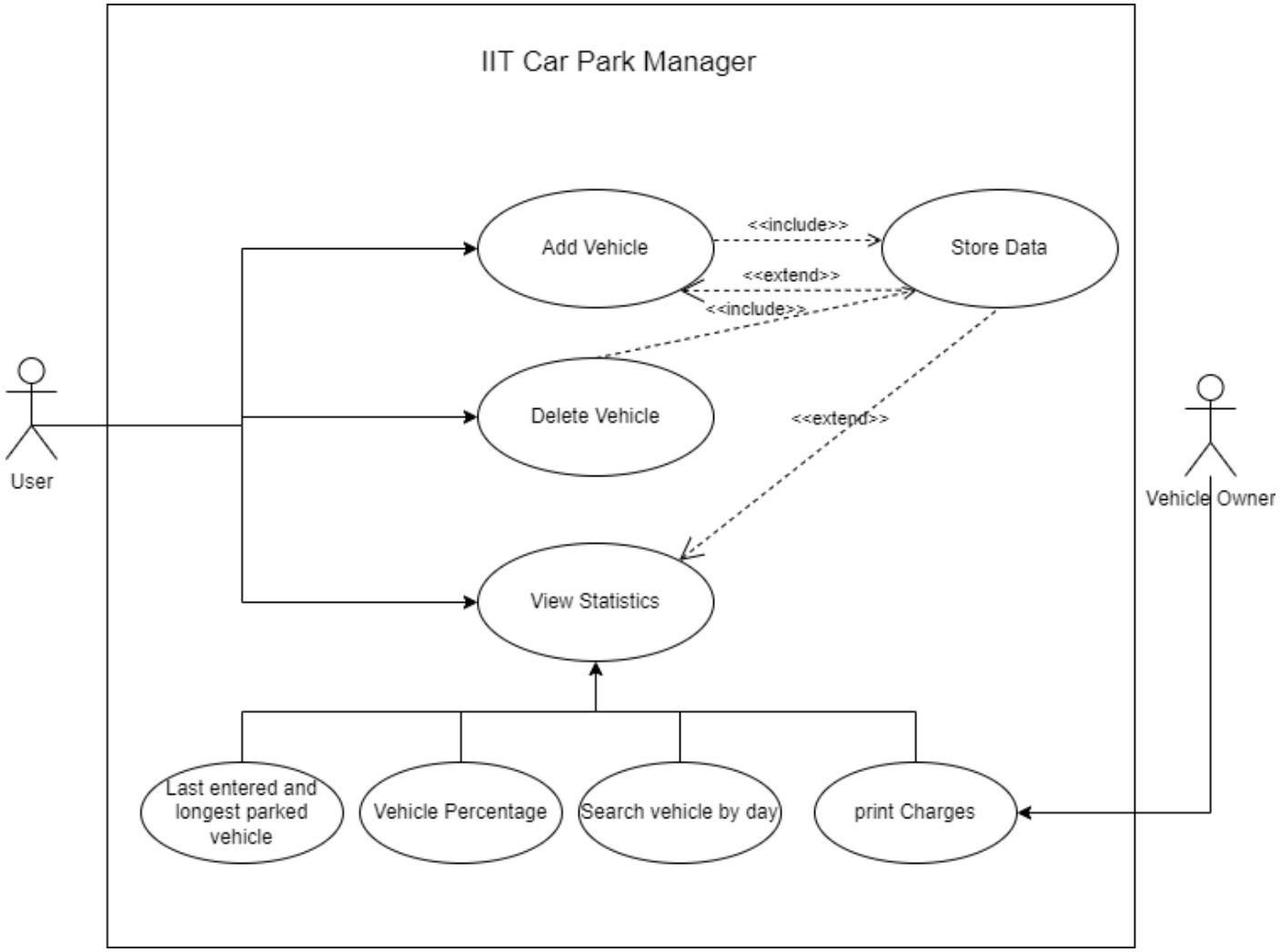
1. Different types of relationships between classes in the UML class diagram

1.
Creational design pattern
creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. The basic form of object creation could result in design problems or in added complexity to the design

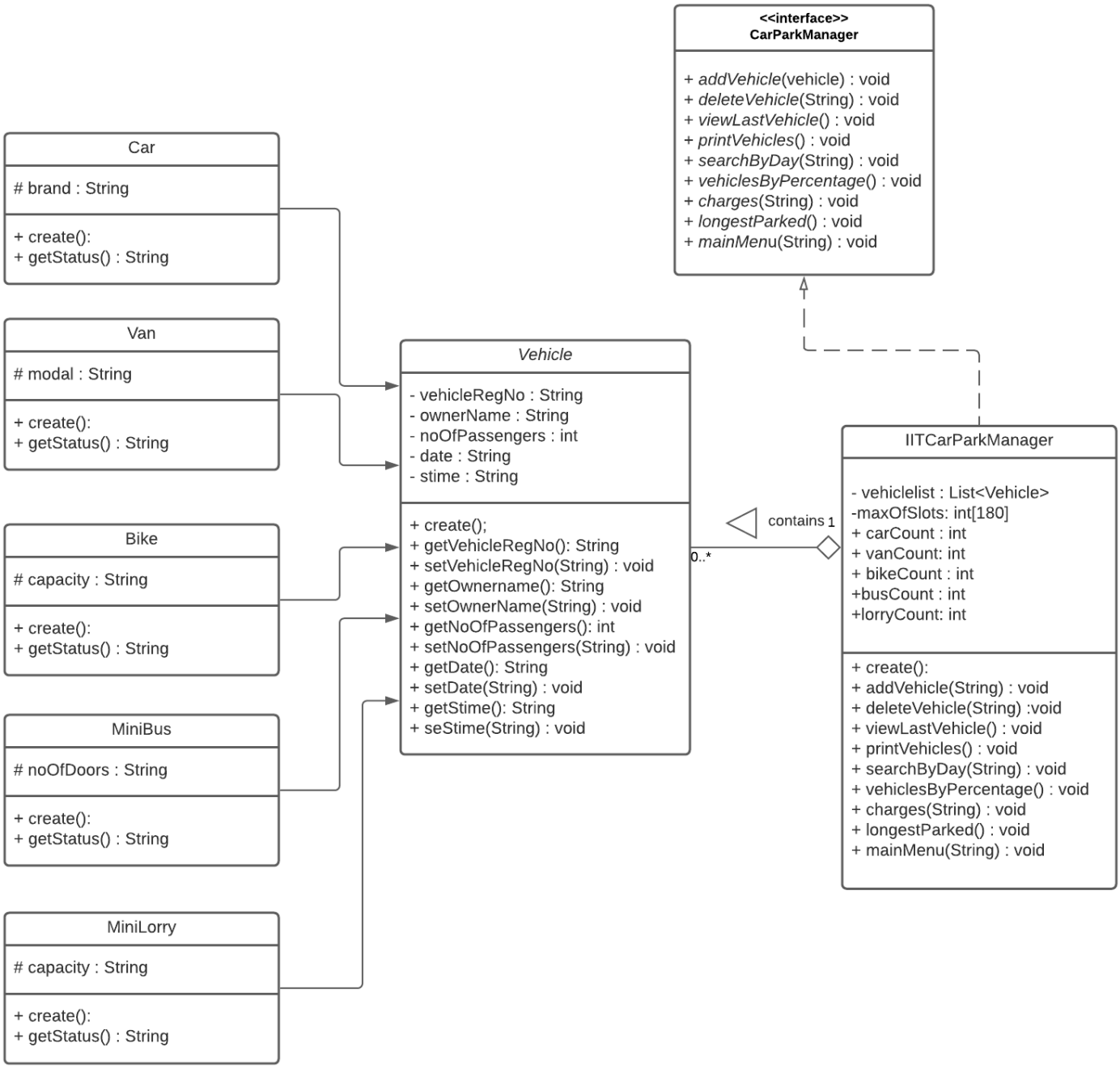
2.
Structural design pattern
Structural design patterns are those that simplify the design of large object structures by identifying relationships between them. They describe common ways of composing classes and objects so that they become repeatable as solutions.

3.
Behavior design pattern
Behavioral design patterns are design patterns that identify common communication patterns among objects. By doing so, these patterns increase flexibility in carrying out communication

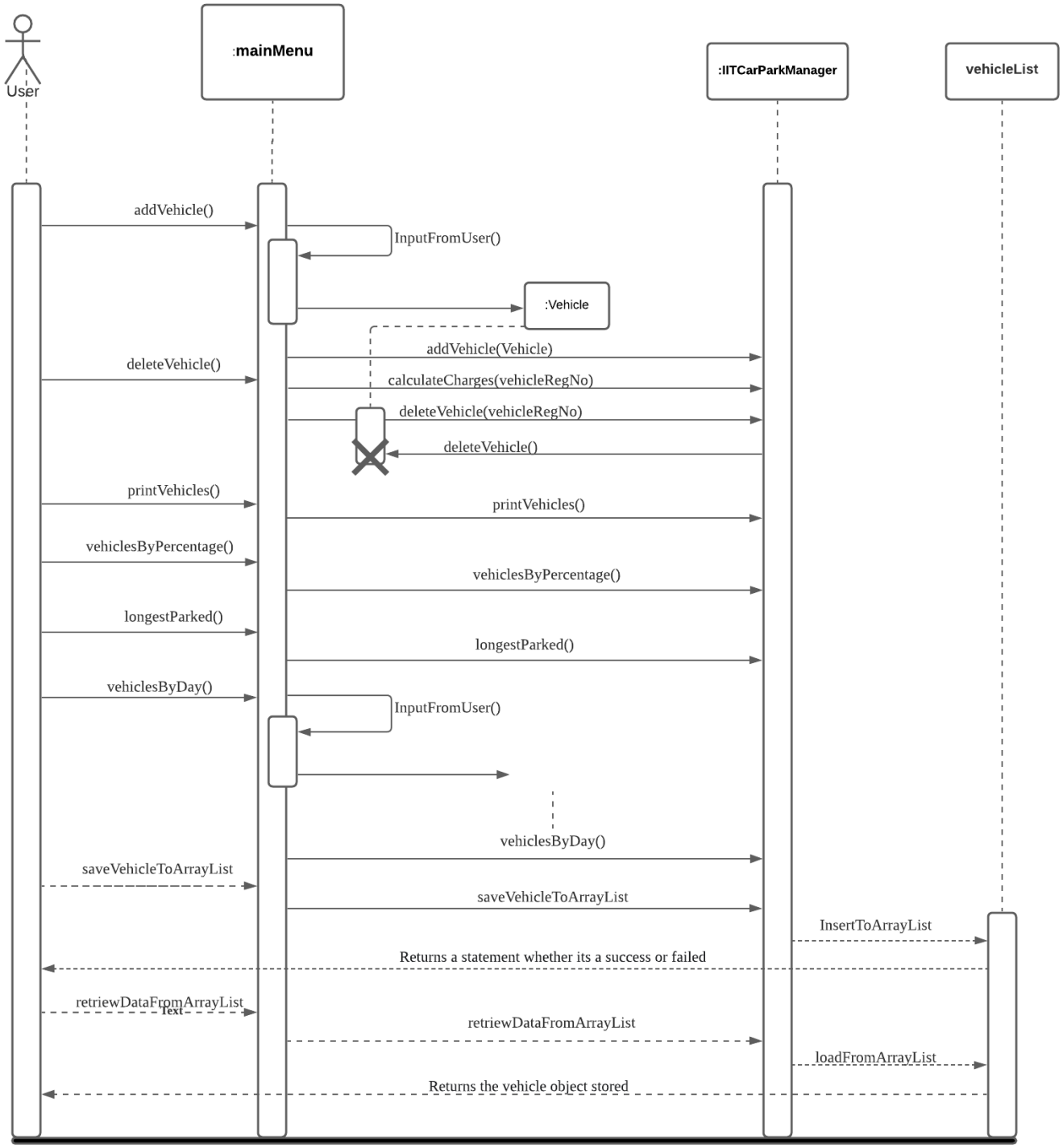
Use Case Diagram



Analysis Class Diagram



Sequence Diagram



Individual Implementation of Car park manager

Vehicle.java

```
import java.util.Comparator;

public abstract class Vehicle {
    private String vehicleRegNo;
    private String ownerName;
    private int noOfPassengers;
    private String date;
    private String stime;

    public Vehicle() {
        super();
        vehicleRegNo = null;
        ownerName = null;
        noOfPassengers = 0;
        date = null;
    }

    public Vehicle(String vehicleRegNo, int noOfPassengers, String ownerName,String
date,String stime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.noOfPassengers = noOfPassengers;
        this.ownerName = ownerName;
        this.date = date;
        this.stime = stime;
    }

    public String getVehicleRegNo() {
        return vehicleRegNo;
    }

    public void setVehicleRegNo(String vehicleRegNo) {
        this.vehicleRegNo = vehicleRegNo;
    }

    public int getNoOfPassengers() {
        return noOfPassengers;
    }

    public void setNoOfPassengers(int noOfPassengers) {
        this.noOfPassengers = noOfPassengers;
    }

    public String getOwnerName(){
        return ownerName;
    }

    public void setOwnerName(String ownerName){
        this.ownerName = ownerName;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getStime() {
        return stime;
    }
```

```
    public void setStime(String stime) {
        this.stime = stime;
    }

    public abstract String getStatus();

    @Override
    public String toString() {
        return "Vehicle{" +
                "vehicleRegNo='" + vehicleRegNo + '\'' +
                ", ownerName='" + ownerName + '\'' +
                ", noOfPassengers=" + noOfPassengers +
                ", date='" + date + '\'' +
                ", stime='" + stime + '\'' +
                '}';
    }
}

class sortItems implements Comparator<Vehicle> {

    public int compare(Vehicle a, Vehicle b)
    {

        return a.getStime().compareTo(b.getStime());
    }
}
```

Car.java

```
public class Car extends Vehicle{

    protected String brand;

    public Car() {
        super();
    }

    public Car(String vehicleRegNo, int noOfPassengers, String brand,String ownerName,
String date,String stime) {
        super();
        this.setVehicleRegNo(vehicleRegNo);
        this.setNoOfPassengers(noOfPassengers);
        this.brand = brand;
        this.setOwnerName(ownerName);
        this.setDate(date);
        this.setStime(stime);
    }


    @Override
    public String getStatus(){

        return brand;
    }

    @Override
    public String toString() {
        return "Car{" +
                "brand='" + brand + '\'' +
                ", vehicleRegNo='" + getVehicleRegNo() + '\'' +
                ", ownerName='" + getOwnerName() + '\'' +
                ", noOfPassengers=" + getNoOfPassengers()+
                ", date='" + getDate() + '\'' +
                ", stime='" + getStime() + '\'' +
                '}';
```

```
    }
}
```

Van.java

```
public class Van extends Vehicle{

    protected String modal;

    public Van() {
        super();
    }

    public Van(String vehicleRegNo, int noOfPassengers, String modal,String
ownerName,String date,String stime) {
        super();
        this.modal = modal;
        this.setVehicleRegNo(vehicleRegNo);
        this.setNoOfPassengers(noOfPassengers);
        this.setOwnerName(ownerName);
        this.setDate(date);
        this.setStime(stime);

    }

    public String getStatus(){
        return modal;
    }

    @Override
    public String toString() {
        return "Van{" +
                "modal='" + modal + '\'' +
                ", vehicleRegNo='" + getVehicleRegNo() + '\'' +
                ", ownerName='" + getOwnerName() + '\'' +
                ", noOfPassengers=" + getNoOfPassengers() +
                ", date='" + getDate() + '\'' +
                ", stime='" + getStime() + '\'' +
                '}';
    }
}
```

MiniBus.java

```
public class MiniBus extends Vehicle{

    protected String noOfDoors;

    public MiniBus() {
        super();
    }

    public MiniBus(String vehicleRegNo, int noOfPassengers, String ownerName, String
date, String stime, String noOfDoors) {
        super();
        this.setVehicleRegNo(vehicleRegNo);
        this.setNoOfPassengers(noOfPassengers);
        this.setOwnerName(ownerName);
        this.setDate(date);
        this.setStime(stime);
        this.noOfDoors = noOfDoors;
    }

    @Override
```

```
    public String getStatus(){

        return noOfDoors;
    }

    @Override
    public String toString() {
        return "MiniBus{" +
                "noOfDoors='" + noOfDoors + '\'' +
                ", vehicleRegNo='" + getVehicleRegNo() + '\'' +
                ", ownerName='" + getOwnerName() + '\'' +
                ", noOfPassengers=" + getNoOfPassengers() +
                ", date='" + getDate() + '\'' +
                ", stime='" + getStime() + '\'' +
                '}';
    }
}
```

MiniLorry.java

```
public class MiniLorry extends Vehicle{
    protected String capacity;

    public MiniLorry() {

        super();
    }

    public MiniLorry(String vehicleRegNo, int noOfPassengers, String ownerName, String
date, String stime, String capacity) {
        super();
        this.capacity = capacity;
        this.setVehicleRegNo(vehicleRegNo);
        this.setNoOfPassengers(noOfPassengers);
        this.setOwnerName(ownerName);
        this.setDate(date);
        this.setStime(stime);
    }

    @Override
    public String getStatus(){

        return capacity;
    }

    @Override
    public String toString() {
        return "MiniLorry{" +
                "capacity='" + capacity + '\'' +
                ", vehicleRegNo='" + getVehicleRegNo() + '\'' +
                ", ownerName='" + getOwnerName() + '\'' +
                ", noOfPassengers=" + getNoOfPassengers() +
                ", date='" + getDate() + '\'' +
                ", stime='" + getStime() + '\'' +
                '}';
    }
}
```

CarParkManager.java (interface)

```
public interface CarParkManager {
    void addVehicle(Vehicle vehicle);
    void deleteVehicle(String regNO);
    //void searchVehicle(String regNo);
    void viewLastVehicle();
    void printStatus();
    void searchByDay(String date);
    void vehiclesByPercentage();
    void charges(String regNo, String etime);
    void longestParked();
    boolean mainMenu();
}
```

IITCarParkManager.java

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.concurrent.TimeUnit;

public class IITCarParkManager implements CarParkManager{

    private List<Vehicle> vehicleList;
    private int maxOfSlots;

    int Vcar =0, Vvan = 0, Vbike = 0, VminiLorry = 0, VminiBus =0;
    float Scar = 0, Svan = 0, Sbike = 0,Slorry = 0, Sbus = 0;



    public IITCarParkManager() {
        super();
        this.vehicleList = new ArrayList<Vehicle>();
        this.maxOfSlots = 180;
    }

    @Override
    public void addVehicle(Vehicle vehicle){
        if (vehicleList.size() < this.maxOfSlots){
            this.vehicleList.add(vehicle);
        } else {
            System.out.println("Car Park is full");
        }
    }

    @Override
    public void deleteVehicle(String regNo){
        List<Vehicle> removeList = new ArrayList<Vehicle>();
        for (Vehicle obj:vehicleList){
            System.out.println("                ");
            System.out.println("================================================");
            System.out.println("<<<<<<<   Vehicle Deleting in processs  >>>>>>>");
            System.out.println("================================================");
            if (obj instanceof Car){
                if (obj.getVehicleRegNo().equals(regNo)){
                    //vehicleList.remove(obj);
                    removeList.add(obj);
                }
            }else if (obj instanceof Van){
                if (obj.getVehicleRegNo().equals(regNo)){
                    //vehicleList.remove(obj);
                    removeList.add(obj);
                }
```

```
            }else if (obj instanceof Bike) {
                if (obj.getVehicleRegNo().equals(regNo)) {
                    //vehicleList.remove(obj);
                    removeList.add(obj);
                }
            }else if (obj instanceof MiniBus) {
                if (obj.getVehicleRegNo().equals(regNo)) {
                    //vehicleList.remove(obj);
                    removeList.add(obj);
                }
            }else if (obj instanceof MiniLorry) {
                if (obj.getVehicleRegNo().equals(regNo)) {
                    //vehicleList.remove(obj);
                    removeList.add(obj);
                }
            }


        }
        if (removeList.size() > 0){
            vehicleList.removeAll(removeList);
        }
        System.out.println("Your vehicle was Deleted Successfully!");

    }

    @Override
    public void viewLastVehicle() {
        Vehicle v = vehicleList.get(vehicleList.size()-1);

        System.out.println(v.getVehicleRegNo());
    }


    @Override
    public void charges(String regNo, String etime){
        int fee = 0;
        for (Vehicle obj:vehicleList){
            System.out.println("<<<<<<<<<<   Wait for Your Bill  >>>>>>>>>>");
            if (obj.getVehicleRegNo().equals(regNo)){
                SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");
                try {
                    Date sd = df.parse(obj.getStime()) ;
                    Date ed = df.parse(etime);

                    long difference_In_Time
                            = ed.getTime() - sd.getTime();

                    long difference_In_Seconds
                            = TimeUnit.MILLISECONDS
                            .toSeconds(difference_In_Time)
                            % 60;

                    long difference_In_Minutes
                            = TimeUnit
                            .MILLISECONDS
                            .toMinutes(difference_In_Time)
                            % 60;

                    long difference_In_Hours
                            = TimeUnit
                            .MILLISECONDS
                            .toHours(difference_In_Time)
                            % 24;

                    //System.out.print("Time Vehicle parked in Car park: ");
```

```
//System.out.println(difference_In_Hours+"hr,"+difference_In_Minutes+"min,"+difference_
In_Seconds+"sec");

                    if (difference_In_Hours <= 3){
                        fee = 300;
                    }else if (difference_In_Hours >3){
                        long diff = difference_In_Hours - 3;
                        int d = (int)diff;
                        fee = 300 + (100 * d);
                    }else if (difference_In_Hours == 00){
                        fee = 3000;
                    }


System.out.println("=============================================");
                    System.out.println("                IIT CAR PARK
");

System.out.println("=============================================");
                    System.out.println("                    INVOICE
");
                    System.out.println("---------------------------------------------
");
                    System.out.println("      Date              : "+obj.getDate());
                    System.out.println("      Starting Time     : "+obj.getStime());
                    System.out.println("      Ending Time       : "+etime);
                    System.out.println("---------------------------------------------
");
                    System.out.println("                  USER DETAILS
");
                    System.out.println("---------------------------------------------
");
                    System.out.println("      User Name         :
"+obj.getOwnerName());
                    System.out.println("      Vehicle Reg.No    :
"+obj.getVehicleRegNo());
                    System.out.println("---------------------------------------------
");
                    System.out.println("               AMOUNT & DENOMINATIONS
");
                    System.out.println("---------------------------------------------
");
                    System.out.println("      Total time parked :
"+difference_In_Hours+":"+difference_In_Minutes+":"+difference_In_Seconds);
                    System.out.println("      Total fee         : "+fee+"LKR");

System.out.println("=============================================");
                    System.out.println("                    Thank you!
");

System.out.println("=============================================");


                }catch (ParseException e){
                    e.printStackTrace();
                }
            }

        }
        deleteVehicle(regNo);
    }

    @Override
    public void longestParked() {
        if (!vehicleList.isEmpty()){
            Collections.sort(vehicleList,new sortItems());
            System.out.println(vehicleList.get(0));
```

```
        }
    }

    /*@Override
    public void searchVehicle(String regNo){
        for (Vehicle obj:vehicleList){
            if (obj instanceof Car){
                if (obj.vehicleRegNo.equals(regNo)){
                    System.out.println(" ");
                    System.out.println("*******************************");
                    System.out.println("Request found!");
                    System.out.println("Owner Name: "+obj.ownerName+"\n"+"Vehicle
Number Plate: "+obj.vehicleRegNo+"\n"+"Number of Passengers: "+obj.noOfPassengers);
                }
            }
        }
    }*/
    @Override
    public void printStatus(){
        //System.out.println(vehicleList.size());
        for (int i = vehicleList.size()-1; i >= 0  ; i--){
            //System.out.println(i);
            Vehicle obj = vehicleList.get(i);

            if (obj instanceof Car){
                System.out.println("*******************************");
                System.out.println("Owner Name: "+obj.getOwnerName()+"\n"+"Vehicle
Number Plate: "+obj.getVehicleRegNo()+"\n"+"Number of Passengers:
"+obj.getNoOfPassengers());
            } else if (obj instanceof Van){
                System.out.println("*******************************");
                System.out.println("Owner Name: "+obj.getOwnerName()+"\n"+"Vehicle
Number Plate: "+obj.getVehicleRegNo()+"\n"+"Number of Passengers:
"+obj.getNoOfPassengers());
            }else if (obj instanceof Bike) {
                System.out.println("*******************************");
                System.out.println("Owner Name: " + obj.getOwnerName() + "\n" +
"Vehicle Number Plate: " + obj.getVehicleRegNo() + "\n" + "Number of Passengers: " +
obj.getNoOfPassengers());
            }else if (obj instanceof MiniBus) {
                System.out.println("*******************************");
                System.out.println("Owner Name: " + obj.getOwnerName() + "\n" +
"Vehicle Number Plate: " + obj.getVehicleRegNo() + "\n" + "Number of Passengers: " +
obj.getNoOfPassengers());
            }else if (obj instanceof MiniLorry) {
                System.out.println("*******************************");
                System.out.println("Owner Name: " + obj.getOwnerName() + "\n" +
"Vehicle Number Plate: " + obj.getVehicleRegNo() + "\n" + "Number of Passengers: " +
obj.getNoOfPassengers());
            }else{
                System.out.println("Unsupported Vehicle!");
            }
        }
    }

    @Override
    public void searchByDay(String day){
        for (Vehicle obj:vehicleList){
            if (obj.getDate().equals(day)){
                System.out.println("Owner Name: "+obj.getOwnerName()+"\n"+"Vehicle
Number Plate: "+obj.getVehicleRegNo()+"\n"+"Number of Passengers:
"+obj.getNoOfPassengers());
            }
        }
    }

    @Override
```

```
    public void vehiclesByPercentage(){
        int total = Vcar + Vvan + Vbike + VminiBus + VminiLorry;
        double Pcar = Vcar*100/total;
        double Pvan = Vvan*100/total;
        double Pbike = Vbike*100/total;
        double Pbus = VminiBus*100/total;
        double Plorry= VminiLorry*100/total;

        System.out.println("Vehicle Percentages: ");
        System.out.println("Cars = "+Pcar+"%");
        System.out.println("Vans = "+Pvan+"%");
        System.out.println("Bike = "+Pbike+"%");
        System.out.println("Mini Bus = "+Pbus+"%");
        System.out.println("Mini Lorry = "+Plorry+"%");
    }

    @Override
    public boolean mainMenu(){
        boolean exit = false;
        System.out.println("=============================================");
        System.out.println("Welcome to the IIT CarPark Management System");
        System.out.println("=============================================");
        System.out.println("Here you can Park your Car,Van, Bike,MiniBus");
        System.out.println("Cars can park in 3rd floor and above");
        System.out.println("Van and bikes are allowed to park in 1stfloor");
        System.out.println("=============================================");
        System.out.println("                                             ");


        System.out.println("=============================================");
        System.out.println("[1] Add new Vehicle");
        System.out.println("[2] Print the list of vehicles parked");
        System.out.println("[3] Delete Vehicle");
        System.out.println("[4] Longest Parked Vehicle");
        System.out.println("[5] Search Vehicle By Day");
        System.out.println("[6] Vehicle percentage");
        System.out.println("[7] View last vehicle");
        System.out.println("[8] Exit");
        System.out.print("Enter your choice: ");
        Scanner sc = new Scanner(System.in);
        int choice = sc.nextInt();

        switch (choice){
            case 1:
                System.out.println("=============================================");
                System.out.println("[1] Add a Car");
                System.out.println("[2] Add a Van");
                System.out.println("[3] Add a Bike");
                System.out.println("[4] Add a Mini Bus");
                System.out.println("[5] Add a Mini Lorry");
                System.out.print("Enter your choice: ");
                int choice2 = sc.nextInt();

                System.out.print("Enter Your Name: ");
                Scanner nm = new Scanner(System.in);
                String name = nm.next();
                System.out.println("Enter Date: [dd-mm-yyyy]");
                Scanner d = new Scanner(System.in);
                String date = d.next();
                System.out.println("Enter Time: [HH:mm:ss]");
                Scanner st = new Scanner(System.in);
                String stime = st.next();

                System.out.print("Enter the Vehicle Registration Number: ");
                String vehicleRegNo = sc.next();
                System.out.print("Enter the number of Passengers on board: ");
                int noOfPassengers = sc.nextInt();
```

```
                Vehicle obj = null;
                switch (choice2){
                    case 1:
                        String brand = null;
                        System.out.print("Enter the car brand: ");
                        brand = sc.next();

                        obj = new
Car(vehicleRegNo,noOfPassengers,brand,name,date,stime);
                        //System.out.println("No.of remaining available slots:
"+(maxOfSlots - 1));
                        Scar += 3;
                        Vcar += 1;
                        break;
                    case 2:
                        String modal = null;
                        System.out.print("Enter modal of van[long/ short]: ");
                        modal = sc.next();

                        obj = new
Van(vehicleRegNo,noOfPassengers,modal,name,date,stime);
                        //System.out.println("No.of remaining available slots:
"+(maxOfSlots - 2));
                        Svan += 6;
                        Vvan +=1;
                        break;
                    case 3:
                        String capacity = null;
                        System.out.print("Enter the engine capacity: ");
                        capacity = sc.next();

                        obj = new
Bike(vehicleRegNo,noOfPassengers,name,date,stime,capacity);
                        //System.out.println("No.of remaining available slots:
"+(maxOfSlots - 1));
                        Sbike +=1;
                        Vbike += 1;
                        break;
                    case 4:
                        String noOfDoors = null;
                        System.out.print("Enter the number of doors[1/2]: ");
                        noOfDoors = sc.next();

                        obj = new
MiniBus(vehicleRegNo,noOfPassengers,name,date,stime,noOfDoors);
                        //System.out.println("No.of remaining available slots:
"+(maxOfSlots - 1));
                        Sbus += 9;
                        VminiBus += 1;
                        break;
                    case 5:
                        String tcapacity = null;
                        System.out.print("Enter the Tank capacity: ");
                        tcapacity = sc.next();

                        obj = new
Bike(vehicleRegNo,noOfPassengers,name,date,stime,tcapacity);
                        //System.out.println("No.of remaining available slots:
"+(maxOfSlots - 1));
                        Slorry += 9;
                        VminiLorry += 1;
                        break;
                    default:
                        System.out.println("<<Invalid vehicle input. Please enter shape
again!>>");
                }
```

```
                float total_spaces = Scar+Svan+Sbike+Sbus+Slorry;
                float final_spaces = (maxOfSlots - total_spaces)/3;
                if (obj != null){
                    addVehicle(obj);
                    System.out.println("Vehicle added Successfully!");
                    System.out.println("No.of remaining available slots:
"+final_spaces);
                } else {
                    System.out.println("ERROR: Object not created Please trt again!");
                }

                break;

            case 2:
                printStatus();
                break;
            case 3:
                Scanner dl = new Scanner(System.in);
                System.out.println("                                                ");
                System.out.println("================================================");
                System.out.println("Enter your car Reg.no: ");
                String regNo = dl.next();
                System.out.print("Enter Time: [HH:mm:ss]");
                Scanner et = new Scanner(System.in);
                String etime = et.next();


                //deleteVehicle(regNo);
                charges(regNo,etime);
                break;
            /*case 4:
                Scanner se = new Scanner(System.in);
                System.out.println("                                                ");
                System.out.println("================================================");
                System.out.println("Enter your car Reg.no: ");
                String vNo = se.next();
                //searchVehicle(vNo);
                break;*/
            case 4:
                longestParked();
                break;
            case 5:
                Scanner dy = new Scanner(System.in);
                System.out.println("                                                ");
                System.out.println("================================================");
                System.out.println("Enter Date: [dd-mm-yyyy]");
                String day = dy.next();
                searchByDay(day);
                break;

            case 6:


                vehiclesByPercentage();
                break;
            case 7:
                viewLastVehicle();
                break;
            case 8:
                exit = true;
                System.out.println("Moving out from the application >>>>>>>");
                break;
            default:
                System.out.println("Invalid Choice please enter 1,2 or 3");

        }
```

```
            return exit;
        }

}
```

OutPut

Menu

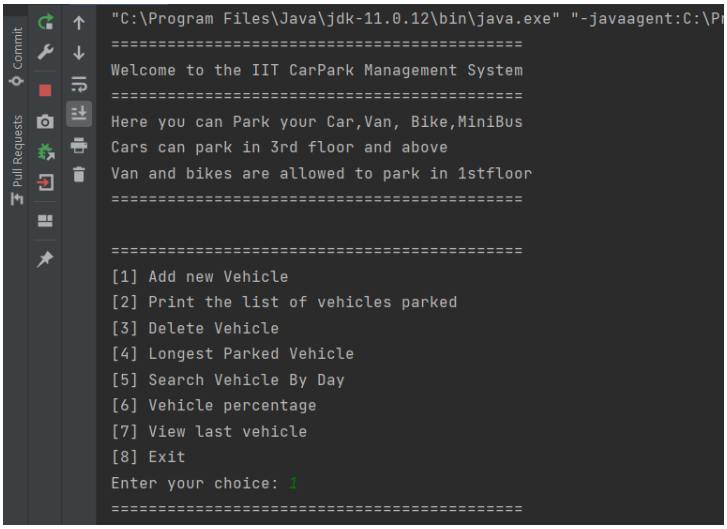


Figure 1

Adding Vehicles to the carpark

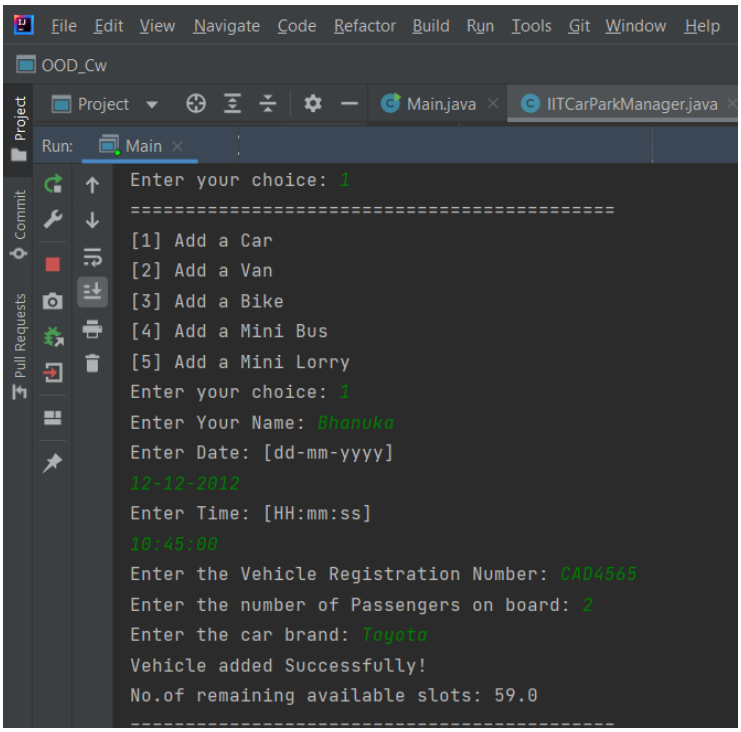


Figure 2

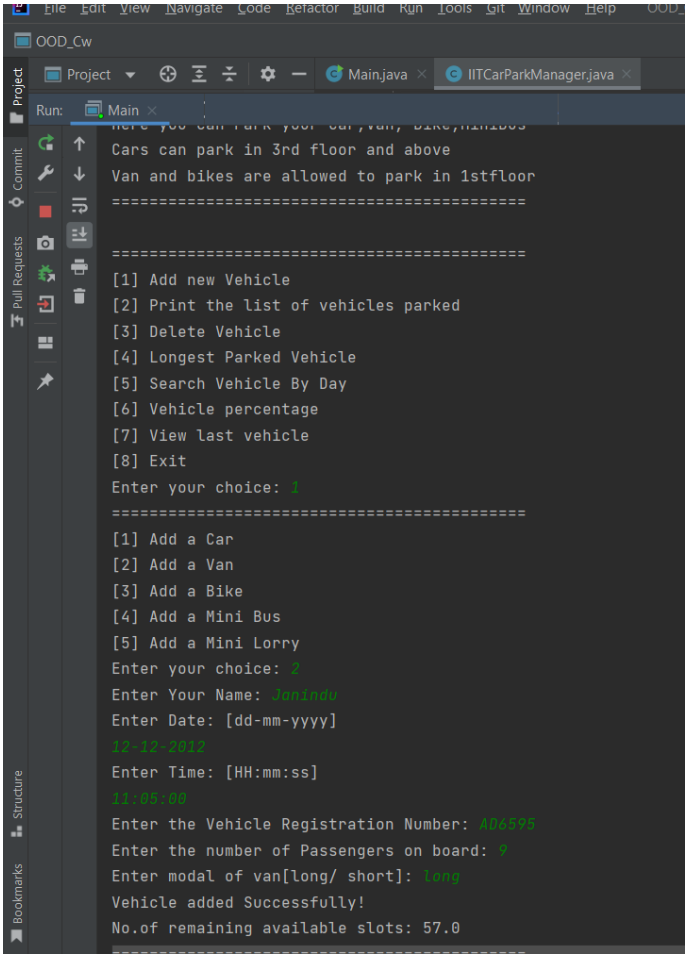


Figure 3

Printing List of vehicles – list of vehicles in carpark in descending order

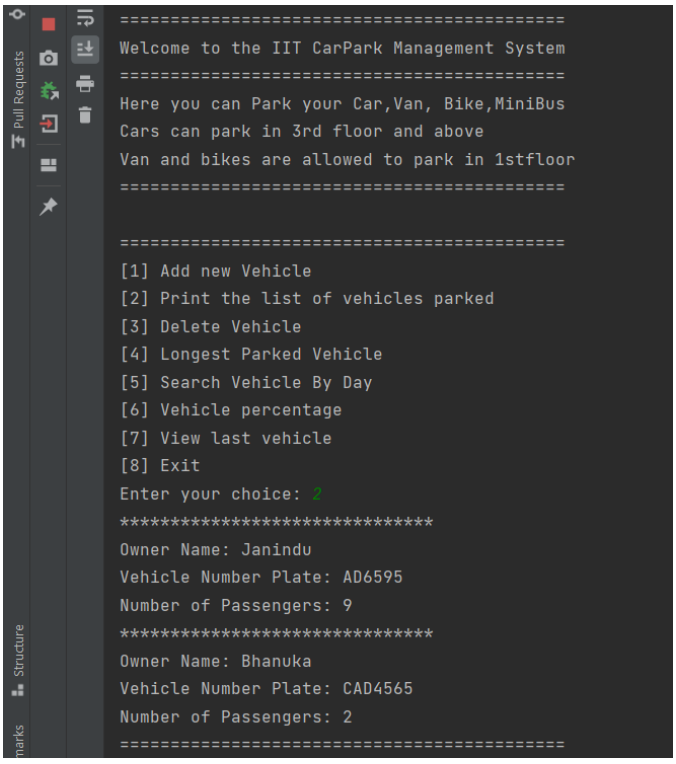


Figure 4

Longest time parked vehicle

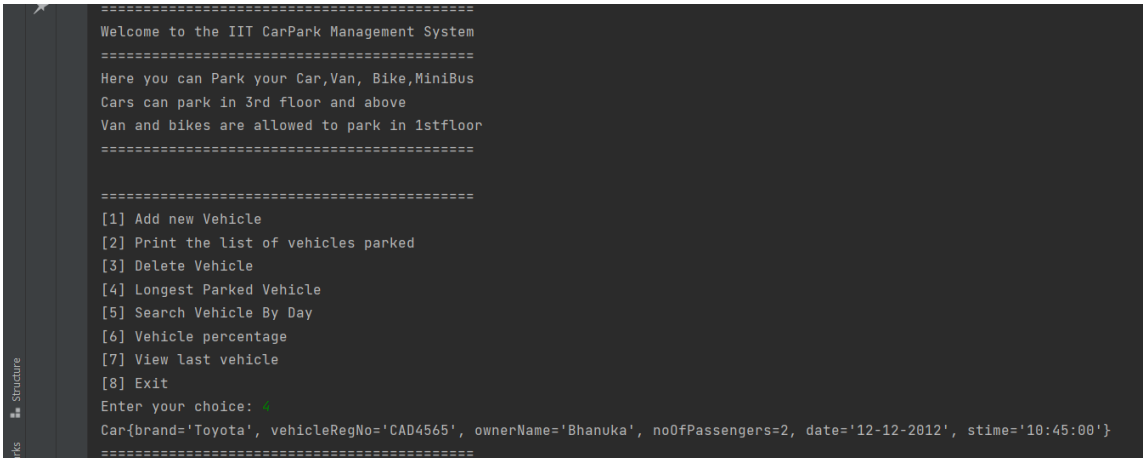


Figure 5

Search Vehicle By day

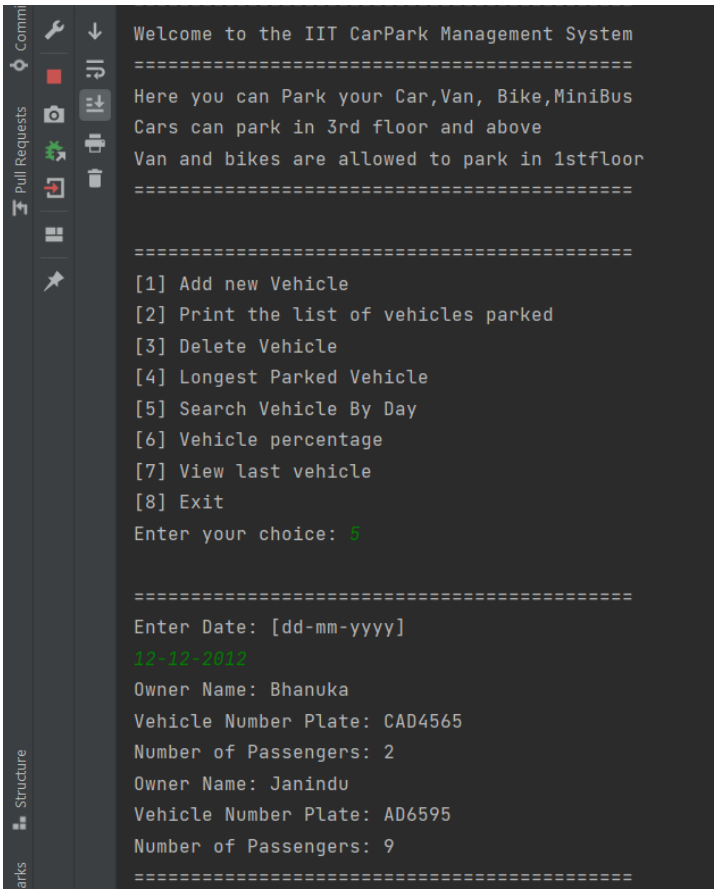


Figure 6

View Last vehicle entered the carpark

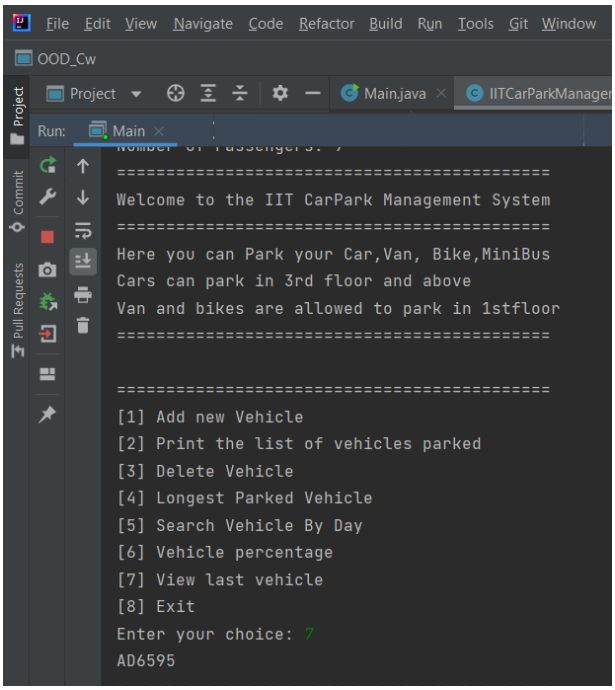


Figure 7

Charges

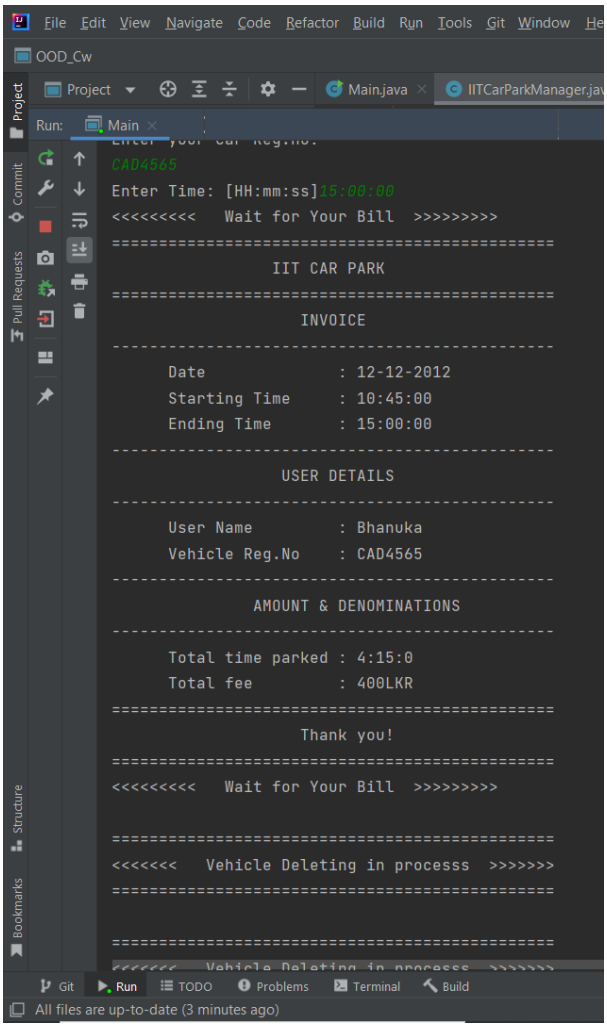


Figure 8

Delete Vehicle

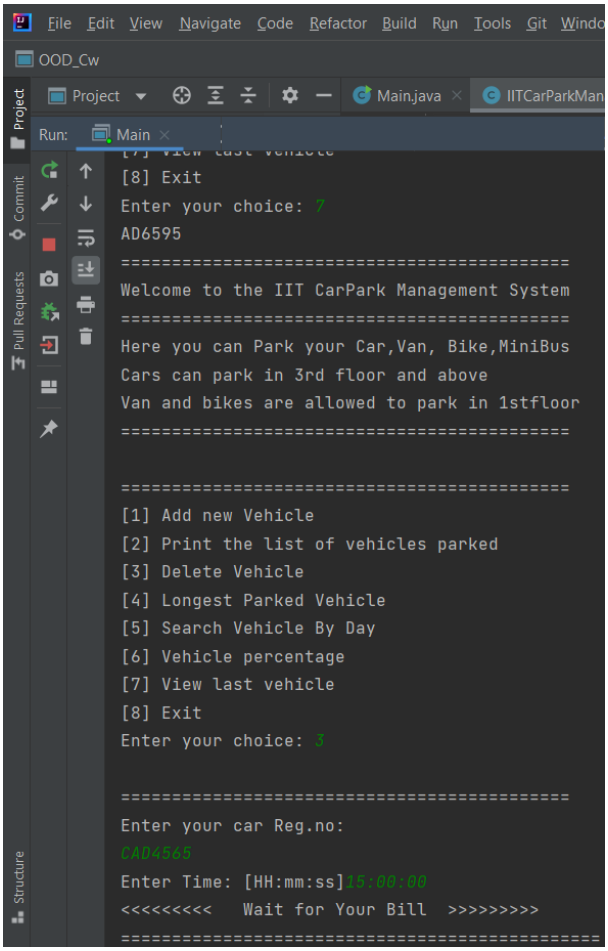


Figure 9