BSc. Artificial Intelligence & Data Science

Level 05

CM 2601

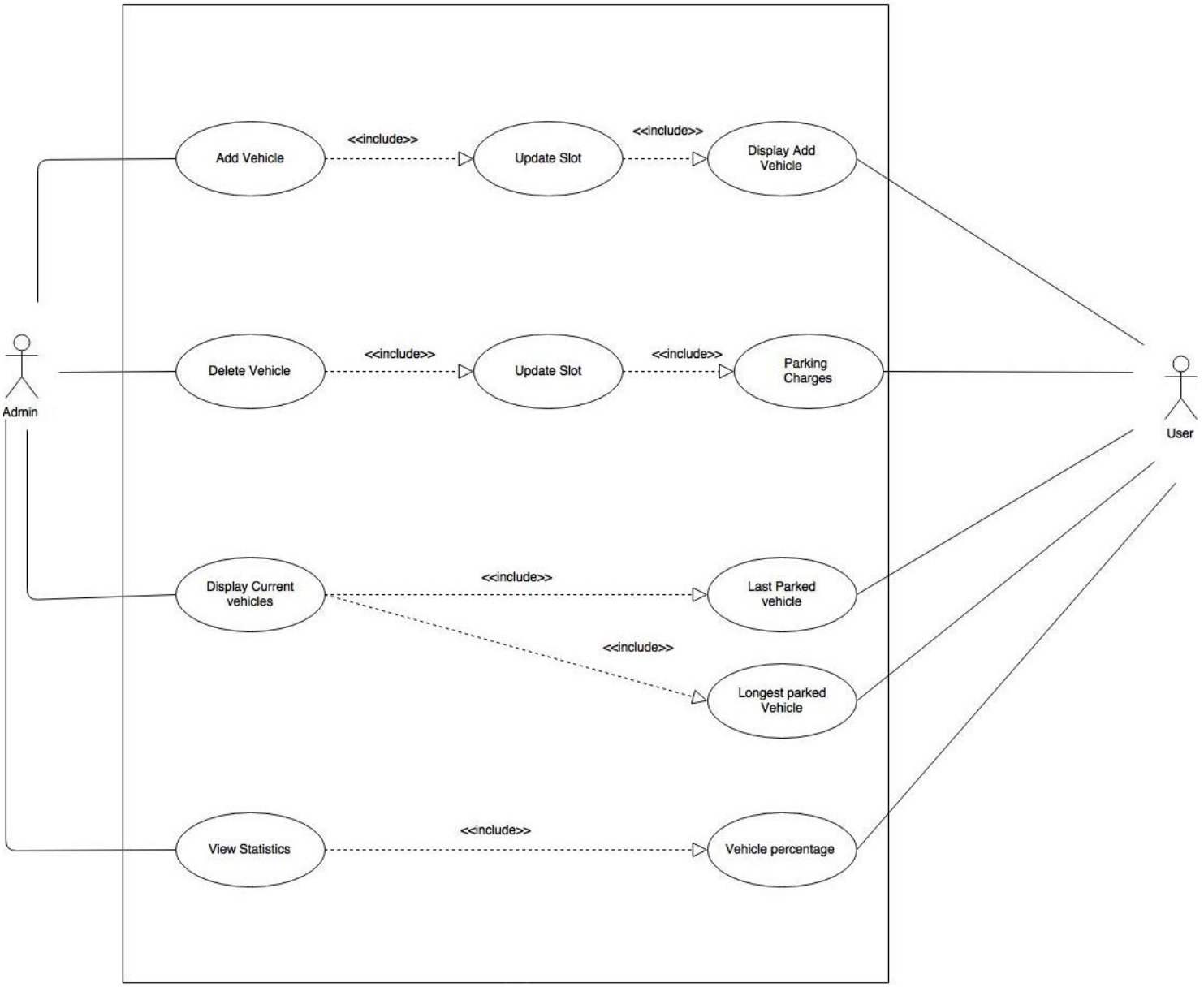
OBJECT ORIENTED DEVELOPMENT

COURSEWORK

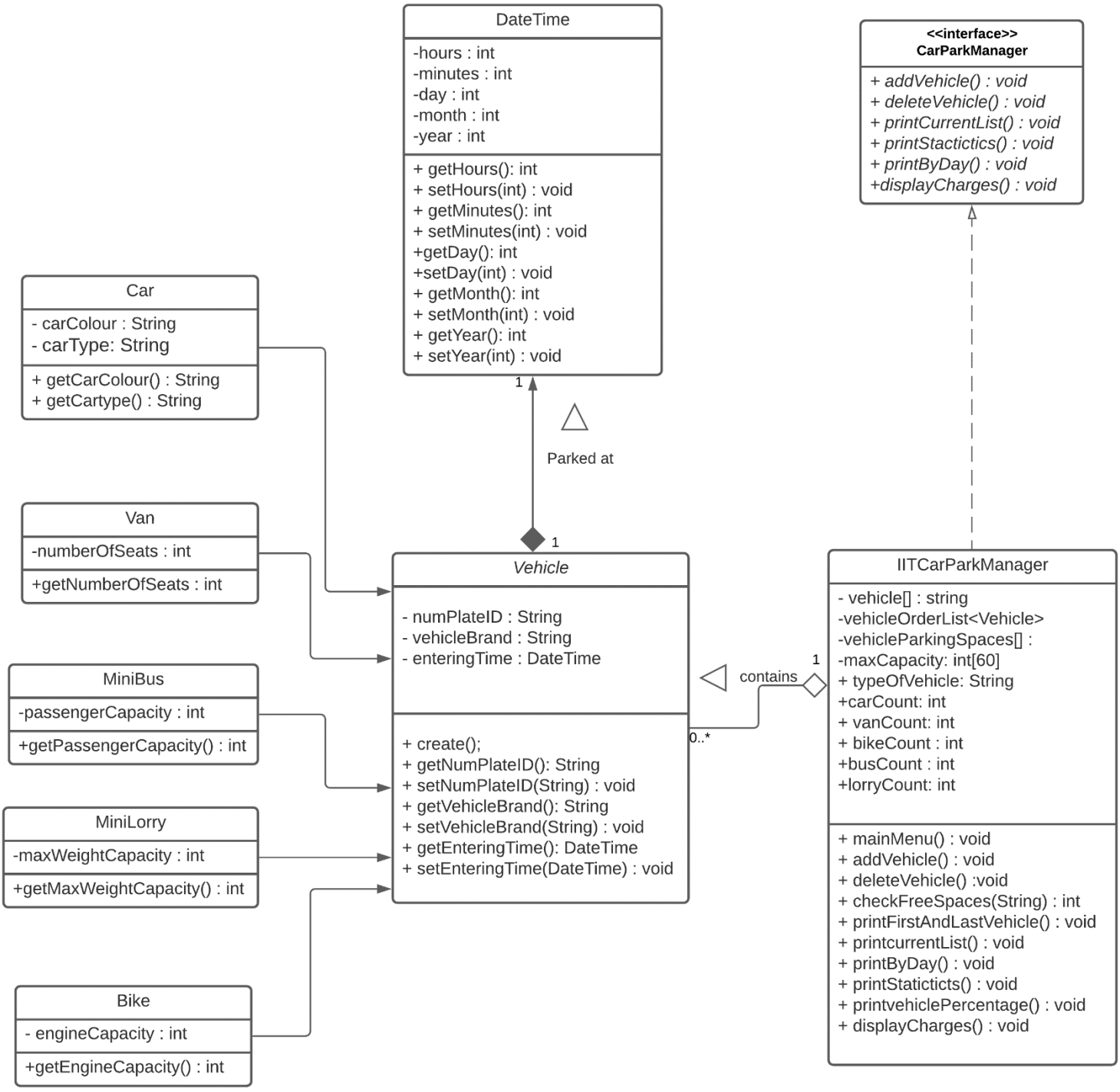
Individual Report

Hemachandrage Hansa Heshan | 20200999 | 2017916

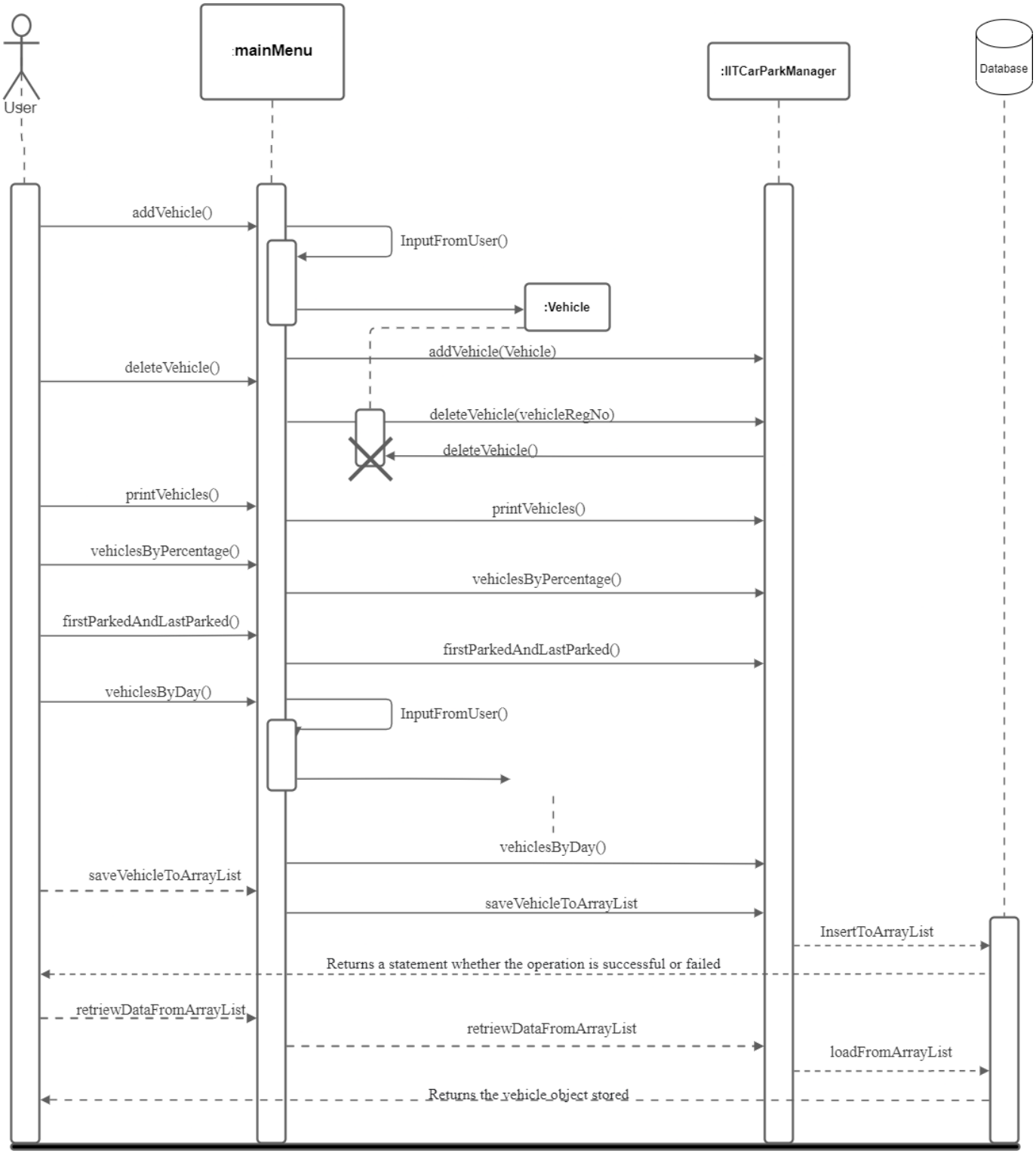
Use Case Diagram



Class Diagram



Sequence Diagram



Source Code

DateTime Class

```
package com.company;


public class DateTime {
   private int hours;
   private int minutes;
   private int day;
   private int month;
   private int year;

   //constructor
   DateTime(int hours, int minutes, int day, int month, int year) {
      super();
      this.hours = hours;
      this.minutes = minutes;
      this.day = day;
      this.month = month;
      this.year = year;
   }

   public int getHours() {
      return hours;
   }

   public void setHours(int hours) {
      this.hours = hours;
   }

   public int getMinutes() {
      return minutes;
   }

   public void setMinutes(int minutes) {
      this.minutes = minutes;
   }

   public int getDay() {
      return day;
   }

   public void setDay(int day) {
      this.day = day;
```

```
    }

    public int getMonth() {
        return month;
    }

    public void setMonth(int month) {
        this.month = month;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }
}
```

Vehicle Class

```
package com.company;


abstract public class Vehicle {
    private String numPlateID;
    private String vehicleBrand;
    private DateTime enteringTime;

    // constructor
    Vehicle(String numPlateID, String vehicleBrand, DateTime enteringTime) {
        this.numPlateID = numPlateID;
        this.vehicleBrand = vehicleBrand;
        this.enteringTime = enteringTime;
    }



    public Vehicle(String numPlateID, String vehicleBrand, DateTime enteringTime, String
carColor) {

    }

    // setter & getters
```

```
    public String getNumPlateID() {
        return numPlateID;
    }

    public void setNumPlateID(String id) {
        this.numPlateID = id;
    }

    public String getVehicleBrand() {
        return vehicleBrand;
    }

    public void setVehicleBrand(String vehicleBrand) {
        this.vehicleBrand = vehicleBrand;
    }

    public String getEnteringTime() {
        return enteringTime.getHours() + " : " + enteringTime.getMinutes() + " " +
enteringTime.getDay() + " / " + enteringTime.getMonth()
                + " / " + enteringTime.getYear();
    }

    public void setEnteringTime(DateTime enteringTime) {

        this.enteringTime = enteringTime;
    }

    public DateTime getEntryTimeObject() {

        return enteringTime;
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "numPlateID='" + numPlateID + '\'' +
                ", vehicleBrand='" + vehicleBrand + '\'' +
                ", enteringTime=" + enteringTime +
                '}';
    }
}
```

Car Class

```
package com.company;


public class Car extends Vehicle{
   private String carColor;
   private String carType;

   public Car(String numPlateID, String vehicleBrand, DateTime enteringTime, String carColor,
String carType) {
      super(numPlateID, vehicleBrand, enteringTime);
      this.carColor = carColor;
      this.carType = carType;
   }

   public Car(String numPlateID, String vehicleBrand, DateTime enteringTime, String carColor) {
      super(numPlateID, vehicleBrand, enteringTime, carColor);
   }

   //setters & getters
   public String getCarType() {
      return carType;
   }

   public void setCarType(String carType) {
      this.carType = carType;
   }

   public String getCarColor() {
      return carColor;
   }
   public void setCarColor(String carColor) {
      this.carColor = carColor;
   }
}
```

Van Class

```
package com.company;


public class Van extends Vehicle{

   private int numberOfSeats;

   Van(String numPlateID, String vehicleBrand, DateTime enteringTime, int numberOfSeats){
      super(numPlateID, vehicleBrand, enteringTime);
      this.numberOfSeats = numberOfSeats;
   }

   //setter & getter for numberOfSeats
   public void setNumberOfSeats(int numberOfSeats){
      this.numberOfSeats = numberOfSeats;
   }

   public int getNumberOfSeats(){
      return numberOfSeats;
   }
}
```

MotorBike Class

```
package com.company;


public class MotorBike extends Vehicle {
   protected int engineCapacity;

   MotorBike(String numPlateID, String vehicleBrand, DateTime entryTime, int engineCapacity) {
      super(numPlateID, vehicleBrand, entryTime);
      this.engineCapacity = engineCapacity;
   }

   public int getEngineCapacity() {

      return engineCapacity;
   }

   public void setEngineCapacity(int engineCapacity) {
      this.engineCapacity = engineCapacity;
```

```
    }
}
```

Minibus Class

```
package com.company;

public class MiniBus extends Vehicle{

    private int passengerCapacity;

    MiniBus(String numPlateID, String vehicleBrand, DateTime enteringTime, int
    passengerCapacity){
        super(numPlateID, vehicleBrand, enteringTime);
        this.passengerCapacity = passengerCapacity;
    }

    //setter & getter for numberOfPassengers
    public void setPassengerCapacity(int passengerCapacity){
        this.passengerCapacity = passengerCapacity;
    }

    public int getPassengerCapacity(){
        return passengerCapacity;
    }
}
```

MiniLorry Class

```
package com.company;

public class MiniLorry extends Vehicle {

        private int maxWeightCapacity;

        MiniLorry(String numPlateID, String vehicleBrand, DateTime enteringTime, int
        maxWeightCapacity){
            super(numPlateID, vehicleBrand, enteringTime);
            this.maxWeightCapacity = maxWeightCapacity;
        }

        //setter & getter for maxWeightCapacity
        public void setMaxWeightCapacity(int maxWeightCapacity){
```

```
        this.maxWeightCapacity = maxWeightCapacity;
    }

    public int getMaxWeightCapacity(){
        return maxWeightCapacity;
    }
}
```

CarParkManager Class - Interface

```
package com.company;


interface CarParkManager {

    public abstract void addVehicle();

    public abstract void deleteVehicle();

    public abstract void printCurrentList();

    public abstract void printStatistics();

    public abstract void printByDay();

    // public abstract void displayParkingCharges();
}
```

IITCarParkManager Class

```
package com.company;

import java.util.ArrayList;
import java.util.Scanner;

public class IITCarParkManager implements CarParkManager {
    private Vehicle[] vehicleParkingSpaces = new Vehicle[60];//parking space array to store
vehicle objects
    static Scanner input = new Scanner(System.in);
    private Vehicle lastEntry = null;//to find the last entered vehicle
```

```
        private ArrayList<Integer> vehicleOrderList = new ArrayList<Integer>();//is used to have the index of the vehicles
        //which are currently parked in the last in First out approach
        private ArrayList<Vehicle> deletedTempVehicleList = new ArrayList<Vehicle>();//stores the vehicle object which had left the parking space

        public static void main(String[] args) {
           System.out.println(" ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
           System.out.println(" |***********  WELCOME TO THE  ***********|");
           System.out.println(" |********- IIT CAR PARK MANAGER -********|");
           System.out.println(" ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
           MainMenu();   //redirected to the MainMenu method
        }


        public static void MainMenu() {
           CarParkManager carParkObject = new IITCarParkManager();
           while (true) {//infinity loop
              System.out.println("");
              System.out.println("1. Add a vehicle to the parking Space");
              System.out.println("2. Delete a vehicle from the parking Space");
              System.out.println("3. Display all vehicles parked currently in the park");
              System.out.println("4. Display statistics of the car park");
              System.out.println("5. Display vehicles of a specific Day");
              System.out.println("Press Q to Quit the Program.");
              System.out.print("Enter Selection: ");

              String userInput = input.next();// prompted for input and stored in
              // the userInput variable
              System.out.println();

              switch (userInput.toLowerCase()) {
                 case "1":
                    carParkObject.addVehicle();
                    break;
                 case "2":
                    carParkObject.deleteVehicle();
                    break;
                 case "3":
                    carParkObject.printCurrentList();
                    break;
                 case "4":
                    carParkObject.printStatistics();
                    break;
                 case "5":
                    carParkObject.printByDay();
                    break;
                 case "q":
```

```
                    System.exit(0);// terminates the program
                    break;
                default:
                    System.err.println("\n**Please, Enter a Valid Input**");
                    System.out.println();
                }
            }
        }

        // Override
        public void addVehicle() {
            boolean typeVal = false;
            String typeOfVehicle = ""; //this string variable holding the type of vehicle
            while (!typeVal) { // loop to make sure only valid type is being entered
                System.out.print("Enter vehicle type(car/bike/van/minibus/minilorry): ");
                typeOfVehicle = input.next();
                if (typeOfVehicle.equalsIgnoreCase("car") || typeOfVehicle.equalsIgnoreCase("van")
                        ||
typeOfVehicle.equalsIgnoreCase("bike")||typeOfVehicle.equalsIgnoreCase("minibus")
                        ||typeOfVehicle.equalsIgnoreCase("minilorry")) {
                    typeVal = true;
                } else {
                    System.err.println("\n**Please, Enter a Valid Input**");
                    System.out.println("");
                }
            }

            int checkFreeSpace = checkForFreeSpaces(typeOfVehicle);
            if (checkFreeSpace == -1) {
                System.out.println("**Parking Slot Full, No free slot available!**");
                return;
            }

            System.out.print("Enter Vehicle's Number Plate ID : ");
            String numPlateID = input.next();
            System.out.print("Enter Vehicle's Brand name (Nissan,Toyota etc.): ");
            String vehicleBrand = input.next();
            int hours;
            int minutes;
            int date;
            int month;
            int year;

            do {
                System.out.print(
                        "Enter the Date & Time (HH)(MM)(DD)(MM)(YYYY): ");
                hours = input.nextInt();
                minutes = input.nextInt();
```

```
                date = input.nextInt();
                month = input.nextInt();
                year = input.nextInt();
            } while (hours < 0 || hours > 24 || minutes < 0 || minutes > 60 || date < 0 || date > 31 ||
month < 0 || month > 12
                || year <= 2020); // Date & Time Validation

            DateTime enteringTime = new DateTime(hours, minutes, date, month, year);
            vehicleOrderList.add(checkFreeSpace);

            switch (typeOfVehicle) {
                case "car":
                    System.out.print("Enter the colour of car: ");
                    while (!input.hasNext()) {
                        System.err.print("Please, Enter a valid colour : ");
                        input.next();// validating user input
                    }
                    String carColour = input.next();
                    System.out.print("Enter car type (sedan/hatchback etc.): ");
                    String carType = input.next();
                    vehicleParkingSpaces[checkFreeSpace] = new Car(numPlateID, vehicleBrand,
enteringTime, carColour);
                    break;

                case "van":
                    System.out.print("Enter Number Of Seats: ");
                    while (!input.hasNextInt()) {
                        System.err.print("Please, Enter an integer value for Number of seats : ");
                        input.next();// validating user input
                    }
                    int numberOfSeats = input.nextInt();
                    vehicleParkingSpaces[checkFreeSpace] = new Van(numPlateID, vehicleBrand,
enteringTime, numberOfSeats);
                    vehicleParkingSpaces[checkFreeSpace + 1] = new Van(numPlateID, vehicleBrand,
enteringTime, numberOfSeats);
                    break;

                case "minibus":
                    System.out.print("Enter Number Of Passengers can be travelled: ");
                    while (!input.hasNextInt()) {
                        System.err.print("Please, Enter an integer value for Number of passengers : ");
                        input.next();// validating user input
                    }
                    int passengerCapacity = input.nextInt();
                    vehicleParkingSpaces[checkFreeSpace] = new MiniBus(numPlateID, vehicleBrand,
enteringTime, passengerCapacity);
                    vehicleParkingSpaces[checkFreeSpace +1] = new MiniBus(numPlateID, vehicleBrand,
enteringTime, passengerCapacity);
```

```
                    vehicleParkingSpaces[checkFreeSpace +2] = new MiniBus(numPlateID, vehicleBrand,
enteringTime, passengerCapacity);
                    break;

                case "minilorry":
                    System.out.print("Enter Maximum Weight can be loaded: ");
                    while (!input.hasNextInt()) {
                        System.err.print("Please, Enter an integer value for Maximum weight : ");
                        input.next();// validating user input
                    }
                    int maxWeightCapacity = input.nextInt();
                    vehicleParkingSpaces[checkFreeSpace] = new MiniLorry(numPlateID, vehicleBrand,
enteringTime, maxWeightCapacity);
                    vehicleParkingSpaces[checkFreeSpace +1] = new MiniLorry(numPlateID, vehicleBrand,
enteringTime, maxWeightCapacity);
                    vehicleParkingSpaces[checkFreeSpace +2] = new MiniLorry(numPlateID, vehicleBrand,
enteringTime, maxWeightCapacity);
                    break;

                case "bike":
                    System.out.print("Enter engine capacity (in cc): ");
                    while (!input.hasNextInt()) {
                        System.err.print("Please, Enter a integer value for engine capacity : ");
                        input.next();// validating user input based on string value
                    }
                    int engineCapacity = input.nextInt();
                    vehicleParkingSpaces[checkFreeSpace] = new MotorBike(numPlateID, vehicleBrand,
enteringTime, engineCapacity);
                    break;
            }
            lastEntry = vehicleParkingSpaces[checkFreeSpace];
            System.out.println("");
            System.out.println(" The Vehicle has parked Successfully! ");
            System.out.println(" No of free spaces remaining is " + totalOfSlots());
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }

        public int checkForFreeSpaces(String VehicleType) {
            for (int i = 0; i < 60; i++) {
                if (vehicleParkingSpaces[i] == null) {
                    if (VehicleType.equalsIgnoreCase("van")) {
                        if (vehicleParkingSpaces[i + 1] == null) {
                            return i;
                        }

                    }
                    else if (VehicleType.equalsIgnoreCase("minibus")) {
                        if ((vehicleParkingSpaces[i + 1] == null) && (vehicleParkingSpaces[i + 2]) == null){
```

```
                return i;
            }
        }
        else if (VehicleType.equalsIgnoreCase("minilorry")) {
            if((vehicleParkingSpaces[i + 1] == null) && (vehicleParkingSpaces[i + 2]) == null){
                return i;
            }
        }
        else { // since one slot is sufficient for cars and bikes
                return i;
            }
        }
      }
      return -1; // if there is no free slots
   }


public int totalOfSlots() {
   int number = 0;
   for (int i = 0; i < 60; i++) {
      if (vehicleParkingSpaces[i] == null) {
         ++number;
      }
   }
   return number;
}

// Override
public void deleteVehicle() {
   boolean foundFlag = false;
   int i;
   System.out.print("Enter Number Plate ID of the vehicle to be Deleted: ");
   String id = input.next();

   for (i = 0; i < 60; i++) { // loop to find the element index
      if (vehicleParkingSpaces[i] != null) {
         if (vehicleParkingSpaces[i].getNumPlateID().equalsIgnoreCase(id)) {
            foundFlag = true;
            break; // end for loop once the value is found
         }
      }
   }
   if (!foundFlag) {
      System.err.println(" Invalid Vehicle Number Plate ID, Please Try Again!!");

      return;
   }
   // to display the vehicle leaving
```

```
    String VehicleType = vehicleParkingSpaces[i].getClass().getSimpleName();
    System.out.println("A " + VehicleType + " left the Vehicle Park.");
    deletedTempVehicleList.add(vehicleParkingSpaces[i]);
    if (VehicleType.equalsIgnoreCase("van")) {
      // to physically remove the element from the vehicle array
      vehicleParkingSpaces[i] = null;
      vehicleParkingSpaces[i + 1] = null;
    }
    else if (VehicleType.equalsIgnoreCase("minibus")) {
      vehicleParkingSpaces[i] = null;
      vehicleParkingSpaces[i + 1] = null;
      vehicleParkingSpaces[i + 2] = null;
    }
    else if (VehicleType.equalsIgnoreCase("minilorry")) {
      vehicleParkingSpaces[i] = null;
      vehicleParkingSpaces[i + 1] = null;
      vehicleParkingSpaces[i + 2] = null;
    }
    else {
      vehicleParkingSpaces[i] = null;
    }
    vehicleOrderList.remove(i);

  }

  // Override
  public void printStatistics() {
    printVehiclePercentage(); // method call of the method which prints
    // vehicle percentage
    printFirstAndLastVehicle();
  }

  private void printVehiclePercentage() {
    int car = 0;
    int van = 0;
    int bike = 0;
    int minibus = 0;
    int minilorry = 0;
    int total = 0;
    String vehicleType;

    for (int i = 0; i < 60; i++) { // loop to find the element index
      if (vehicleParkingSpaces[i] != null) { // if an element is not empty
        vehicleType = vehicleParkingSpaces[i].getClass().getSimpleName();
        ++total;
        switch (vehicleType) { // to increment each vehicle type counter
          case "Car":
            ++car;
```

```
            break;
        case "Van":
            ++van;
            ++i; // to skip the next space as well since a van occupied 2 spaces.
            break;
        case "MiniBus":
            ++minibus;
            ++i;
            ++i; // to skip the next spaces as well since a minibus occupied 3 spaces.
            break;
        case "MiniLorry":
            ++minilorry;
            ++i;
            ++i; // to skip the next spaces as well since a minilorry occupied 3 spaces.
            break;
        case "Motorbike":
            ++bike;
            break;
      }
    }
}
// Percentage calculation
int carPercentage;
int vanPercentage;
int minibusPercentage;
int minilorryPercentage;
int bikePercentage;

if (total == 0) { // if car park is empty(to avoid arithmeticException)
    carPercentage = 0;
    vanPercentage = 0;
    bikePercentage = 0;
    minibusPercentage = 0;
    minilorryPercentage = 0;
}
else {
    carPercentage = (car * 100 / total);
    vanPercentage = (van * 100 / total);
    minibusPercentage = (minibus * 100 / total);
    minilorryPercentage = (minilorry * 100 / total);
    bikePercentage = (bike * 100 / total);
}

System.out.println("Currently Parked Vehicle percentage");
System.out.println("-----------------------------------");
System.out.println("        CAR      : " + carPercentage + "%");
System.out.println("        VAN      : " + vanPercentage + "%");
System.out.println("        MINI BUS  : " + minibusPercentage + "%");
```

```
            System.out.println("        MINI LORRY: " + minilorryPercentage + "%");
            System.out.println("        BIKE     :" + bikePercentage + "%");
            System.out.println("");
        }

        private void printFirstAndLastVehicle() {
            // to find the vehicle that was parked first.
            if (vehicleOrderList.size() != 0) {
                System.out.println("First vehicle Which was parked");
                System.out.println("------------------------------");
                System.out.println("ID : " +
vehicleParkingSpaces[vehicleOrderList.get(0)].getNumPlateID());
                System.out.println("Type : " +
vehicleParkingSpaces[vehicleOrderList.get(0)].getClass().getSimpleName());
                System.out.println("Entry time : " +
vehicleParkingSpaces[vehicleOrderList.get(0)].getEnteringTime());
            } else {
                System.out.println("No vehicle in the parking currently");
            }

            // to find the last vehicle that entered the parking slot.
            if (lastEntry != null) {
                System.out.println("Last vehicle which was parked");
                System.out.println("------------------------------");
                System.out.println("ID : " + lastEntry.getNumPlateID());
                System.out.println("Type : " + lastEntry.getClass().getSimpleName());
                System.out.println("Entry time : " + lastEntry.getEnteringTime());
            } else {
                System.out.println("**The Parking space is Empty no vehicles are parked currently**");
            }

        }

        // Override
        public void printCurrentList() {
            if (vehicleOrderList.size() == 0) {
                System.out.println("No vehicles are currently available in the parking space");
            }
            for (int i = (vehicleOrderList.size() - 1); i >= 0; i--) {
                int index = vehicleOrderList.get(i);
                System.out.println("Slot " + (index + 1) + " is Occupied.");
                System.out.println("ID plate: " + vehicleParkingSpaces[index].getNumPlateID());
                System.out.println("Entry time: " + vehicleParkingSpaces[index].getEnteringTime());
                System.out.println("Type: " + vehicleParkingSpaces[index].getClass().getSimpleName());
                System.out.println("");
            }

        }
```

```
        // Override
        public void printByDay() {
            boolean isValidFlag = true;
            int date = 0, month = 0, year = 0;
            do {
                System.out.print("Enter DayDay(DD) MonthMonth(MM) YearYearYearYear(YYYY) to
search for Vehicles: ");
                try {
                    date = Integer.parseInt(input.next());
                    month = Integer.parseInt(input.next());
                    year = Integer.parseInt(input.next());
                    if (date > 0 || date < 31 || month > 0 || month < 12 || year == 2021) {
                        isValidFlag = true;
                    } else {
                        isValidFlag = false;
                        System.err.println("Invalid Date input, Please Try Again");
                    }
                } catch (Exception ex) {
                    System.err.println("Invalid input, Please Try Again");
                    isValidFlag = false;
                }
            } while (!isValidFlag);
            int count = 0;
            for (int i = 0; i < 60; i++) {
                if (vehicleParkingSpaces[i] != null) {
                    int objectDate = vehicleParkingSpaces[i].getEntryTimeObject().getDay();
                    int objectMonth = vehicleParkingSpaces[i].getEntryTimeObject().getMonth();
                    int objectYear = vehicleParkingSpaces[i].getEntryTimeObject().getYear();
                    if ((objectDate == date) && (objectMonth == month) && (objectYear == year)) {
                        count++;
                        String type = vehicleParkingSpaces[i].getClass().getSimpleName();
                        String id = vehicleParkingSpaces[i].getNumPlateID();
                        System.out.println(count + " : " + type + " Number Plate ID No : " + id);
                        if (type.equalsIgnoreCase("van")) {
                            i++;
                        }
                        if (type.equalsIgnoreCase("minibus")) {
                            i++;
                            i++;
                        }
                        if (type.equalsIgnoreCase("minilorry")) {
                            i++;
                            i++;
                        }

                    }
                }
```

```
            }
            for (int y = 0; y < deletedTempVehicleList.size(); y++) {
               System.out.println("im here 3");
               int objDate = deletedTempVehicleList.get(y).getEntryTimeObject().getDay();
               int objMonth = deletedTempVehicleList.get(y).getEntryTimeObject().getMonth();
               int objYear = deletedTempVehicleList.get(y).getEntryTimeObject().getYear();
               if ((objDate == date) && (objMonth == month) && (objYear == year)) {
                  count++;
                  String type = deletedTempVehicleList.get(y).getClass().getSimpleName();
                  String id = deletedTempVehicleList.get(y).getNumPlateID();
                  System.out.println(count + " : " + type + " ID No : " + id);
                  if (type.equalsIgnoreCase("van")) {
                     y++;
                  }
                  if (type.equalsIgnoreCase("minibus")) {
                     y++;
                     y++;
                  }
                  if (type.equalsIgnoreCase("minilorry")) {
                     y++;
                     y++;
                  }
               }
            }

            if (count == 0) {
               System.out.println("**No vehicles were currently parked on " + date + "-" + month + "-" +
year + "**");
            }
         }

      }
```

Output

Main Menu

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 |***********  WELCOME TO THE  ***********|
 |********- IIT CAR PARK MANAGER -********|
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


1. Add a vehicle to the parking Space
2. Delete a vehicle from the parking Space
3. Display all vehicles parked currently in the park
4. Display statistics of the car park
5. Display vehicles of a specific Day
Press Q to Quit the Program.
Enter Selection: 
```

Adding a vehicle

```
Enter Selection: 1

Enter vehicle type(car/bike/van/minibus/minilorry): car
Enter Vehicle's Number Plate ID : kl2134
Enter Vehicle's Brand name (Nissan,Toyota etc.): toyota
Enter the Date & Time (HH)(MM)(DD)(MM)(YYYY): 08
09
12
12
2021
Enter the colour of car: red
Enter car type (sedan/hatchback etc.): sedan

 The Vehicle has parked Successfully!
 No of free spaces remaining is 59
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. Add a vehicle to the parking Space
2. Delete a vehicle from the parking Space
3. Display all vehicles parked currently in the park
4. Display statistics of the car park
5. Display vehicles of a specific Day
Press Q to Quit the Program.
Enter Selection: 
```

Deleting A vehicle

```
Enter Selection: 1

Enter vehicle type(car/bike/van/minibus/minilorry): bike
Enter Vehicle's Number Plate ID : kl9090
Enter Vehicle's Brand name (Nissan,Toyota etc.): bajaj
Enter the Date & Time (HH)(MM)(DD)(MM)(YYYY): 09
09
09
12
2021
Enter engine capacity (in cc): 100

 The Vehicle has parked Successfully!
 No of free spaces remaining is 59
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. Add a vehicle to the parking Space
2. Delete a vehicle from the parking Space
3. Display all vehicles parked currently in the park
4. Display statistics of the car park
5. Display vehicles of a specific Day
Press Q to Quit the Program.
Enter Selection: 2

Enter Number Plate ID of the vehicle to be Deleted: kl9090
A MotorBike left the Vehicle Park.

1. Add a vehicle to the parking Space
2. Delete a vehicle from the parking Space
3. Display all vehicles parked currently in the park
4. Display statistics of the car park
```