BSc. Artificial Intelligence & Data Science

Level 05

CM 2601

OBJECT ORIENTED DEVELOPMENT

COURSEWORK

Oshadi Irugalbandara

IIT ID : 20200985

RGU USERNAME : 2017898

Question 01

Clases

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type

```
//Student class is created
public class Student{
    public static void main(String[] args) {

    }
}
```

Instance variables

These are non static variables

Instance variables are used by objects to store their states.

```
class Student {
    public String studentName;
    //instance variable with public access
    private int studentID;
    //instance variable with private access
}
```

Instance methods

Instance methods are non static methods

There are two types of instance methods. Those are getters and setters

They are used to access and show the class fields.

```
class Student {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String N) {
        this.name = N;
    }

    public static void main(String[] args) {
        Student obj = new Student();
        obj.setName("My name is Oshadi");
        System.out.println(obj.getName());
    }
}
```

Static variables

Those are variables belong to the class and initialized only once at the start of the execution

```
class Student {
    //static variable
    private static String constant;
    public static final String name = "Oshadi";

    public static void main(String[] args) {
        constant = "My name is ";
        System.out.println(constant + name);
    }
}
```

static methods

Those are the methods that can be called without creating an object of a class

```
class Student {
    public static void main(String[] args) {
        display();
    }
    static void display() {
        System.out.println("I'm Oshadi.");
    }
}
```

Constructors

Constructors used to initialize objects. The constructor is called when an object of a class is created.

```
class Number{
    // Create a class attribute
    int n;
    // Create a class constructor for the Main class
    public Number() {
        n = 7;
    }
    public static void main(String[] args) {
        // Create an object of class Main (This will call the constructor)
        Number myObj = new Number();
        // Print the value of x
        System.out.println(myObj.n);
    }
}
```

Getters and setters

They are used to access and show the class fields.

```
class Student {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String N) {
        this.name = N;
    }

    public static void main(String[] args) {
        Student obj = new Student();
        obj.setName("My name is Oshadi");
        System.out.println(obj.getName());
    }
}
```

Objects

A Java 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.

```
//Student class is created
public class Student{
    int x = 5;

    public static void main(String[] args) {
        //object is created
        Student myObj = new Student();
        System.out.println(myObj.x);
    }
}
```

OOP concepts

OOps, concepts in java is to improve code readability and reusability by defining a Java program efficiently.

Abstraction

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. Abstraction defines an object in terms of its properties (attributes), behavior (methods), and interfaces (means of communicating with other objects).

```
// Abstract class
abstract class Animal {
    // Abstract method (does not have a body)
    public abstract void animalSound();
    // Regular method
    public void sleep() {
        System.out.println("Zzz");
    }
}
// Subclass (inherit from Animal)
class Pig extends Animal {
    public void animalSound() {
        // The body of animalSound() is provided here
        System.out.println("The pig says: wee wee");
    }
}
class AnimalSound{
    public static void main(String[] args) {
        Pig myPig = new Pig(); // Create a Pig object
        myPig.animalSound();
        myPig.sleep();
    }
}
```

Encapsulation

Encapsulation is one of the fundamental concepts in object-oriented programming.It describes the idea of bundling data and methods that work on that data within one unit.

```
public class Student {
    private String name;
    private String idNum;
    private int age;

    public int getAge(){return age;}
    public String getName(){return name;}
    public String getIdNum(){return idNum;}
    public void setAge(int newAge){ age = newAge;}
    public void setName(String newName){name = newName;}
    public void setIdNum(String newId){idNum = newId;}
}
class RunEncap{
    public static void main(String args[]) {
        Student encap = new Student();
        encap.setName("James");
        encap.setAge(20);
        encap.setIdNum("12343ms");

        System.out.print("Name : " + encap.getName() + " Age : " + encap.getAge());
    }
}
```

Inheritance

Inheritance in Java is a concept that acquires the properties from one class to other classes.

```
class Employee{
    float salary=40000;
}
//inheritance used
class Programmer extends Employee{
    int bonus=10000;
    public static void main(String args[]){
        Programmer p=new Programmer();
        System.out.println("Programmer salary is:"+p.salary);
        System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
```

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.

```
class Animal {
    public void animalSound() {
        System.out.println("The animal makes a sound");
    }
}

class Pig extends Animal {
    public void animalSound() {
        System.out.println("The pig says: wee wee");
    }
}

class Dog extends Animal {
    public void animalSound() {
        System.out.println("The dog says: bow wow");
    }
}
```

Abstract classes and abstract method

An abstract class is a class that is declared abstract; it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

```
abstract class Student{
    abstract void run();
}
class Honda4 extends Bike{
    void run(){System.out.println("running safely");}
    public static void main(String args[]){
        Bike obj = new Honda4();
        ((Honda4) obj).run();
    }
}
```

A method without a body (no implementation) is known as an abstract method. A method must always be declared in an abstract class.

```
interface SquareCube {
    public abstract int squareNum(int n);
    public class AbstractMethodEx2 implements SquareCube {

        public int squareNum(int num) {
            return num * num;
        }
        public int cubeNum(int num) {
            return num * num * num;
        }
        public static void main(String args[]) {
            SquareCube obj = new AbstractMethodEx2();
            System.out.println("Square of number is: " + obj.squareNum( n: 7));
        }
    }
}
```

Interface

An interface is a completely "abstract class" that is used to group related methods with empty bodies

```
// interface
interface Animal {
    public void animalSound(); // interface method (does not have a body)
    public void run(); // interface method (does not have a body)
}
```

Question 02

- Association.
- Directed Association.
- Reflexive Association.
- Multiplicity.
- Aggregation.
- Composition.

- Inheritance/Generalization.
- Realization.

Question 03

Creational Design Patterns

Creational design patterns are concerned with the way of creating objects. These design patterns are used when a decision must be made at the time of instantiation of a class

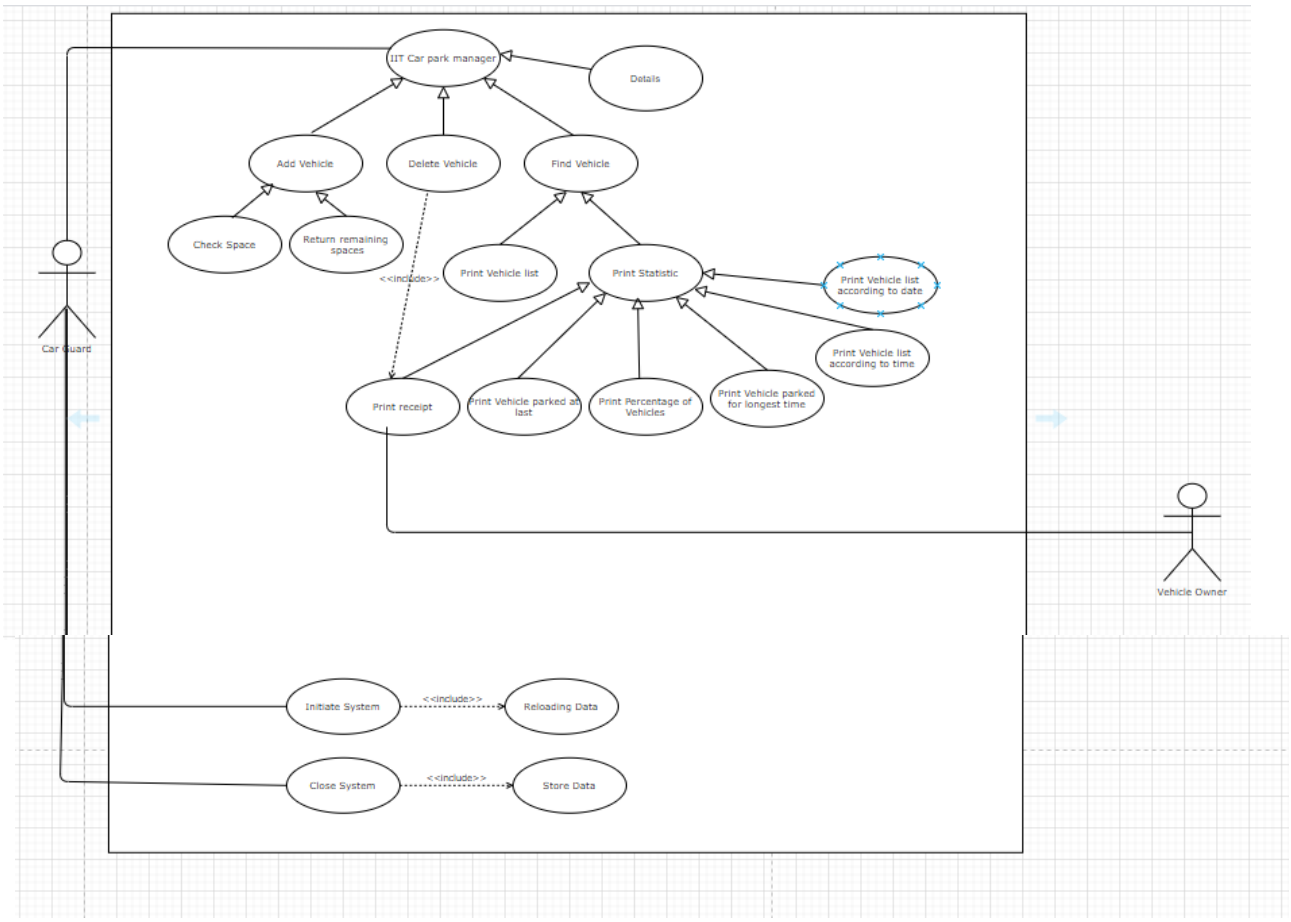
Structural Design pattern

Structural design patterns are concerned with how classes and objects can be composed, to form larger structures.

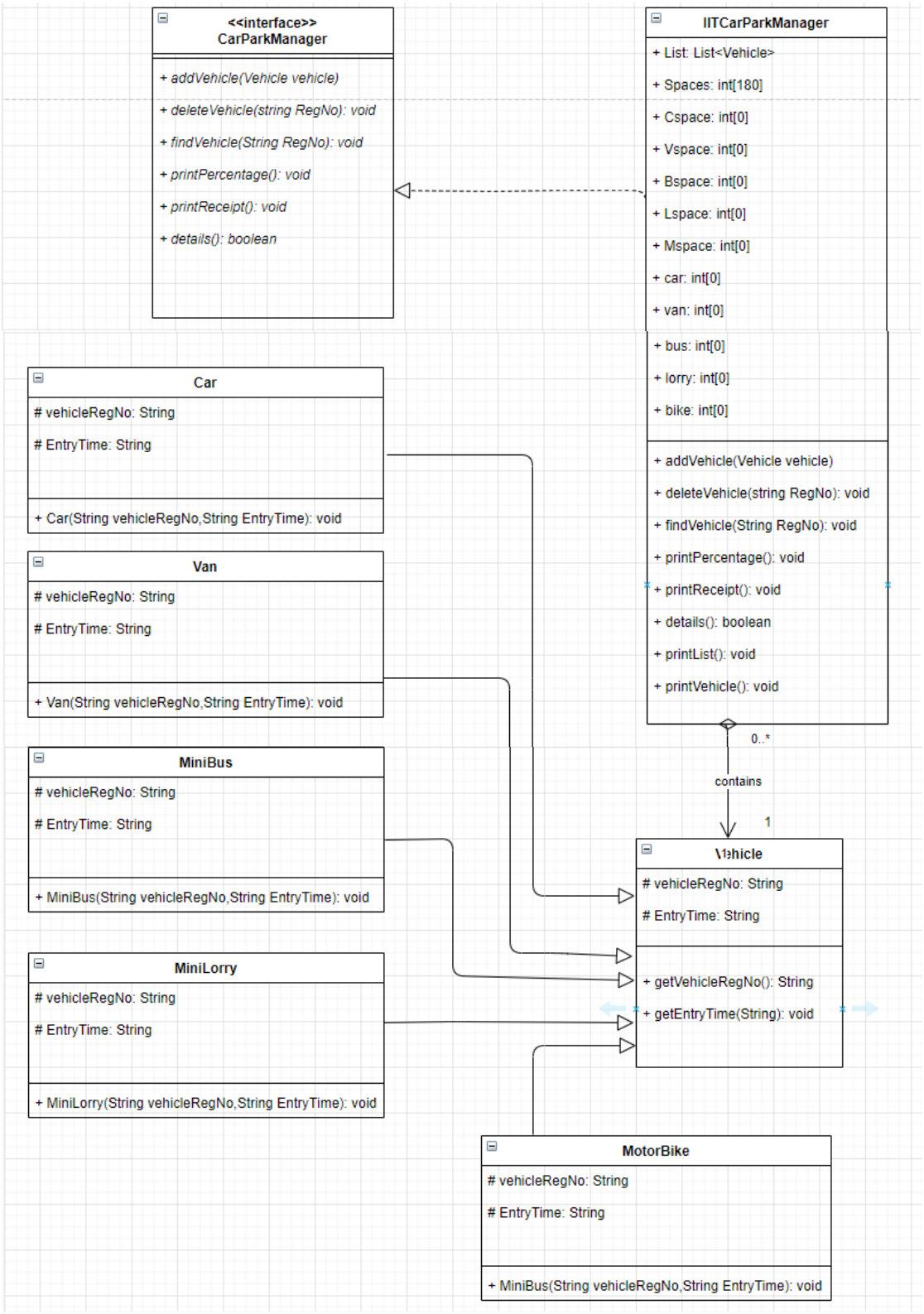
Behavior Design Pattern

Behavioral design patterns are concerned with the interaction and responsibility of objects.

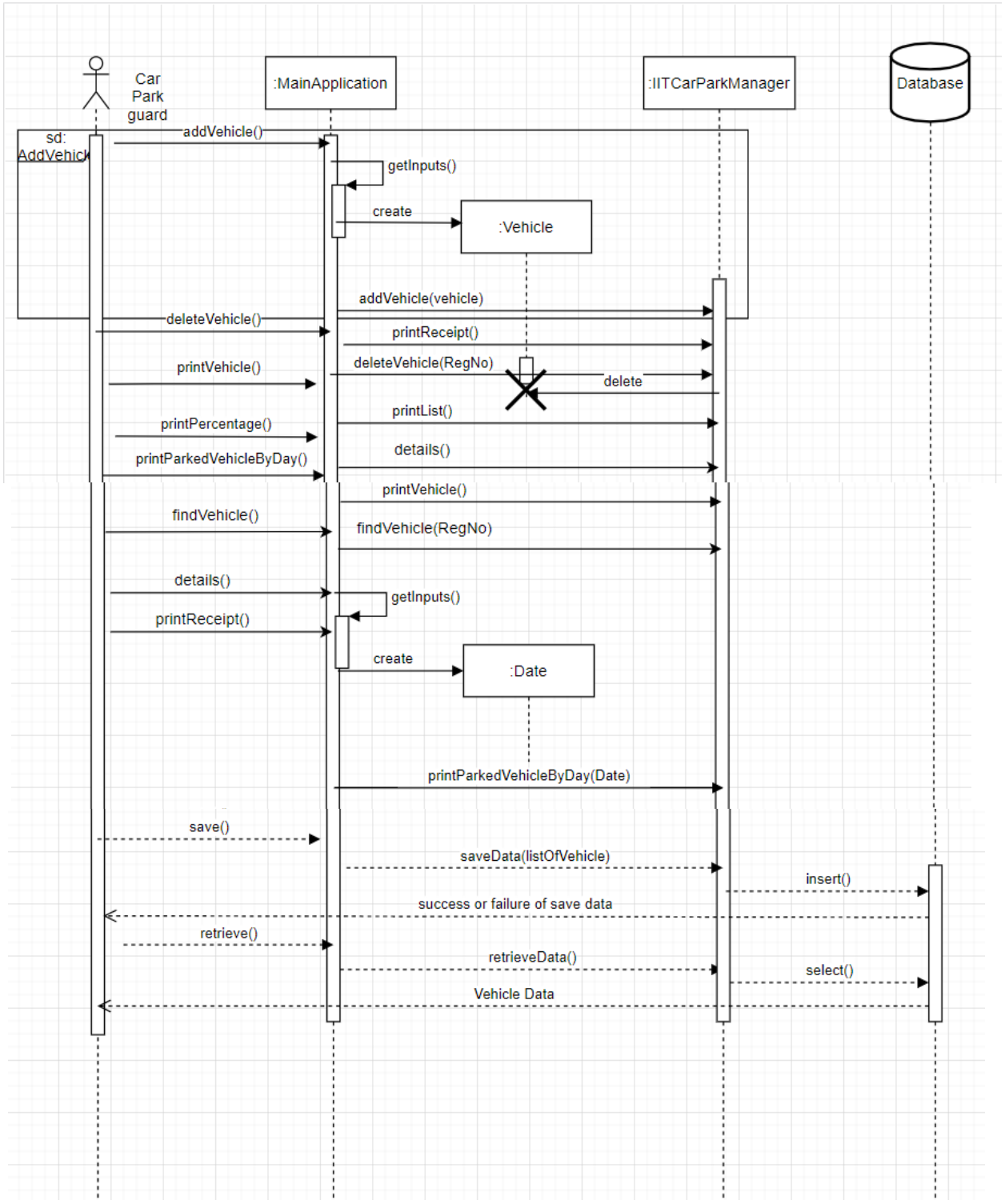
Use Case Diagram



Class Diagram



Sequence Diagram



Individual Code Outputs

Outputs when a car and a mini lorry is added

Add vehicle

```
================================================================
                    WELCOME TO THE CAR PARK
================================================================


                     Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last element that was parked
Please enter number 8 to exit from the system
 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Please enter number 1 to add a car
Please enter number 2 to add a van
Please enter number 3 to add a motor bike
Please enter number 4 to add a mini bus
Please enter number 5 to add a mini lorry
 1
Enter the vehicle register Number: CAA-5663
VEHICLE ADDED!
REMAINING NUMBER OF SPACE: 59
```

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last element that was parked
Please enter number 8 to exit from the system
 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a car
Please enter number 2 to add a van
Please enter number 3 to add a motor bike
Please enter number 4 to add a mini bus
Please enter number 5 to add a mini lorry
 5
Enter the vehicle register Number: GZ-4512
VEHICLE ADDED!
REMAINING NUMBER OF SPACE: 56
```

Print the list of vehicles

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last element that was parked
Please enter number 8 to exit from the system
 3
VEHICLE IS A MINI LORRY AND REGISTER NUMBER IS: GZ-4512
VEHICLE IS A CAR AND REGISTER NUMBER IS: CAA-5663
```

Print the percentage of vehicles

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last element that was parked
Please enter number 8 to exit from the system
 4
Percentage of cars is 50%
Percentage of vans is 0%
Percentage of motor bikes is 0%
Percentage of mini buses is 0%
Percentage of mini lorries is 50%
```

Check the vehicle

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last element that was parked
Please enter number 8 to exit from the system
 6
Enter the vehicle register Number: CAA-5663
VEHICLE IS A CAR, OF REGISTER NUMBER: CAA-5663
```

Print the vehicle parked for the longest time and the last parked vehicle

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last vehicle that was parked
Please enter number 8 to exit from the system
 7
REGISTER NUMBER OF THE VEHICLE PARKED FOR LONGEST TIME IS CAA-5663
REGISTER NUMBER OF THE LAST PARKED VEHICLE IS GZ-4512
```

Delete Vehicle

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last vehicle that was parked
Please enter number 8 to exit from the system
 2

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter the vehicle register Number: CAA-5663
A CAR WITH REGISTER NUMBER CAA-5663 LEFT THE CAR PARK
```

Print the receipt

```
                    Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last vehicle that was parked
Please enter number 8 to exit from the system
 5
Enter the register number: CAA-5663
Enter the entry time in HH:mm:ss order: 08:30:45
Enter the exit time in HH:mm:ss order: 12:40:13
Enter the entry date in DD/MM/YYYY order: 21/12/2021

Enter the vehicle type
[1] Car
[2] Van
[3] Motor Bike
[4] Mini Bus
[5] Mini Lorry
1
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                            CAR PARK
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                            INVOICE
------------------------------------------------------------
            Date                : 21/12/2021
            Starting Time       : 08:30:45
            Ending Time         : 12:40:13
            Vehicle Reg.No      : CAA-5663
           Vehicle parked for : 4 hours 9 minutes 28 Seconds.
------------------------------------------------------------
                            CHARGE
------------------------------------------------------------
         Total charge           : LKR 1000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                            THANK YOU!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Exit from the system

```
                      Select your preference
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter number 1 to add a vehicle to the car park
Please enter number 2 to remove a vehicle from the car park
Please enter number 3 to print the list of vehicles parked
Please enter number 4 to print the percentage of vehicles parked
Please enter number 5 to print the receipt
Please enter number 6 to check the status of the vehicle
Please enter number 7 to print the first and last vehicle that was parked
Please enter number 8 to exit from the system
 8
CLOSE!
```

Individual code

CarParkManager.java

```
package ParkingSystem;
public interface CarParkManager {
   void addVehicle(Vehicle vehicle);
   void deleteVehicle(String RegNO);
   void findVehicle(String RegNo);
   void printPercentage();
   void printReceipt();
   boolean details();
}
class Main {
   public static void main(String[] args){

System.out.println("============================================================
=======");
       System.out.println("                      WELCOME TO THE CAR PARK");

System.out.println("============================================================
=======");
       System.out.println("");
       CarParkManager carPark = new IITCarParkManager();
       boolean exit = false;
       while (!exit){
           exit = carPark.details();
       }
   }
}
```

IITCarParkManager.java

```
package ParkingSystem;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

public class IITCarParkManager implements CarParkManager {

   public List<Vehicle> List;
   public int Spaces;
   public List<String> entryTime;
   int Cspace, Vspace, Bspace, Lspace, Mspace = 0;
   int car, van, bus, lorry, bike = 0;
   int CP, VP, BP, MP, LP;
   int TotalSpace;
   int FinalSpace;
   double CarPercentage, VanPercentage, BusPercentage, LorryPercentage,
BikePercentage;
   double TotalVehicles;
   String EntryTime;
   String ltv;
   String stv;

   public IITCarParkManager() {
       super();
       this.List = new ArrayList<Vehicle>();
       this.Spaces = 180; //60*3
   }

   @Override
   public void addVehicle(Vehicle vehicle) {
       if (List.size() < this.Spaces) {
           this.List.add(vehicle);
           TotalSpace++;
       }
   }

   @Override
   public void deleteVehicle(String RegNo) {
       for (Vehicle obj : List) {
           if (obj instanceof Car) {
```

```
                if (obj.vehicleRegNo.equals(RegNo)) {
                    List.remove(obj);
                    car -= 1;
                    Cspace-=3;
                    System.out.println("A CAR WITH REGISTER NUMBER " +
obj.vehicleRegNo + " LEFT THE CAR PARK");
                }
            }
            else if (obj instanceof Van) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    List.remove(obj);
                    van -= 1;
                    Vspace-=6;
                    System.out.println("A VAN WITH REGISTER NUMBER " +
obj.vehicleRegNo + " LEFT THE CAR PARK");
                }
            }
            else if (obj instanceof MiniBus) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    List.remove(obj);
                    bus -= 1;
                    Bspace-=9;
                    System.out.println("A MINI BUS WITH REGISTER NUMBER " +
obj.vehicleRegNo + " LEFT THE CAR PARK");
                }
            }
            else if (obj instanceof MiniLorry) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    List.remove(obj);
                    lorry -= 1;
                    Lspace-=9;
                    System.out.println("A MINI LORRY WITH REGISTER NUMBER " +
obj.vehicleRegNo + " LEFT THE CAR PARK");
                }
            }
            else if (obj instanceof MotorBike) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    List.remove(obj);
                    bike -= 1;
                    Mspace-=1;
                    System.out.println("A MOTOR BIKE WITH REGISTER NUMBER " +
obj.vehicleRegNo + " LEFT THE CAR PARK");
                }
            }
        }
    }

    @Override
```

```
    public void findVehicle(String RegNo) {
        for (Vehicle obj : List) {
            if (obj instanceof Car) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    System.out.println("VEHICLE IS A CAR, OF REGISTER NUMBER: "
+ obj.vehicleRegNo);
                }
            }
            else if (obj instanceof Van) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    System.out.println("VEHICLE IS A VAN, OF REGISTER NUMBER: "
+ obj.vehicleRegNo);
                }
            }
            else if (obj instanceof MiniBus) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    System.out.println("VEHICLE IS A MINI BUS, OF REGISTER
NUMBER: " + obj.vehicleRegNo);
                }
            }
            else if (obj instanceof MiniLorry) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    System.out.println("VEHICLE IS A MINI LORRY, OF REGISTER
NUMBER: " + obj.vehicleRegNo);
                }
            }
            else if (obj instanceof MotorBike) {
                if (obj.vehicleRegNo.equals(RegNo)) {
                    System.out.println("VEHICLE IS A MOTOR BIKE, OF REGISTER
NUMBER: " + obj.vehicleRegNo);
                }
            }
        }
    }
    @Override
    public boolean details() {
        boolean exit = false;
        System.out.println(" ");
        System.out.println("                        Select your preference");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~");
        System.out.println("Please enter number 1 to add a vehicle to the car
park");
        System.out.println("Please enter number 2 to remove a vehicle from the
car park");
        System.out.println("Please enter number 3 to print the list of vehicles
parked");
```

```
        System.out.println("Please enter number 4 to print the percentage of
vehicles parked");
        System.out.println("Please enter number 5 to print the receipt");
        System.out.println("Please enter number 6 to check the status of the
vehicle");
        System.out.println("Please enter number 7 to print the first and last
vehicle that was parked");
        System.out.println("Please enter number 8 to exit from the system");
        System.out.print(" ");
        Scanner sc = new Scanner(System.in);
        int preference = sc.nextInt();

        if (preference == 1) {

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~");
            System.out.println("Please enter number 1 to add a car");
            System.out.println("Please enter number 2 to add a van");
            System.out.println("Please enter number 3 to add a motor bike");
            System.out.println("Please enter number 4 to add a mini bus");
            System.out.println("Please enter number 5 to add a mini lorry");
            System.out.print(" ");
            int choice2 = sc.nextInt();

            System.out.print("Enter the vehicle register Number: ");
            String vehicleRegNo = sc.next();

            Vehicle veh = null;
            switch (choice2) {
                case 1:
                    veh = new Car(vehicleRegNo, EntryTime);
                    Cspace += 3;
                    car += 1;
                    break;
                case 2:
                    veh = new Van(vehicleRegNo, EntryTime);
                    Vspace += 6;
                    van += 1;
                    break;
                case 3:
                    veh = new MotorBike(vehicleRegNo, EntryTime);
                    Mspace += 1;
                    bike += 1;
                    break;
                case 4:
                    veh = new MiniBus(vehicleRegNo, EntryTime);
                    Bspace += 9;
                    bus += 1;
```

```
                    break;
                case 5:
                    veh = new MiniLorry(vehicleRegNo, EntryTime);
                    Lspace += 9;
                    lorry += 1;
                    break;
                default:
                    System.out.println("INVALID INPUT");
            }
            TotalSpace = Cspace + Vspace + Bspace + Lspace + Mspace;
            FinalSpace = (Spaces - TotalSpace) / 3;

            if (veh != null) {
                addVehicle(veh);
                System.out.println("VEHICLE ADDED!");
                if (FinalSpace != 0)
                    System.out.println("REMAINING NUMBER OF SPACE: " +
FinalSpace);
                else
                     System.out.println("NO SPACE AVAILABLE");
            } else {
                System.out.println("ERROR!");
            }
        }

        else if (preference == 2) {
            Scanner dl = new Scanner(System.in);
            System.out.println(" ");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~");
            System.out.print("Enter the vehicle register Number: ");
            String regNo = dl.nextLine();
            deleteVehicle(regNo);
        }
        else if (preference == 3) {
            printList();
        }
        else if (preference == 4) {
            printPercentage();
        }
        else if (preference == 5) {
            printReceipt();
        }
        else if (preference == 6) {
            Scanner reg = new Scanner(System.in);
            System.out.print("Enter the vehicle register Number: ");
            String regNo2 = reg.nextLine();
```

```
            findVehicle(regNo2);
        }
        else if (preference == 7) {
            printVehicle();
        }
        else if (preference == 8) {
            exit = true;
            System.out.println("CLOSE!");
        }
        else{
            System.out.println("INVALID INPUT");
        }
        return exit;
    }

    @Override
    public void printPercentage() {
        double car1 = car;
        double van1 = van;
        double bike1 = bike;
        double lorry1 = lorry;
        double bus1 = bus;

        TotalVehicles = (car1 + van1 + bike1 + lorry1 + bus1);

        CarPercentage = (car1 / TotalVehicles) * 100;
        VanPercentage = (van1 / TotalVehicles) * 100;
        BikePercentage = (bike1 / TotalVehicles) * 100;
        BusPercentage = (bus1 / TotalVehicles) * 100;
        LorryPercentage = (lorry1 / TotalVehicles) * 100;

        CP = (int) CarPercentage;
        VP = (int) VanPercentage;
        MP = (int) BikePercentage;
        BP = (int) BusPercentage;
        LP = (int) LorryPercentage;

        System.out.println("Percentage of cars is " + CP + "%");
        System.out.println("Percentage of vans is " + VP + "%");
        System.out.println("Percentage of motor bikes is " + MP + "%");
        System.out.println("Percentage of mini buses is " + BP + "%");
        System.out.println("Percentage of mini lorries is " + LP + "%");
    }
    @Override
    public void printReceipt() {
        Scanner sc1 = new Scanner(System.in);
        System.out.print("Enter the register number: ");
        String regno = sc1.nextLine();
```

```
        Scanner sc2 = new Scanner(System.in);
        System.out.print("Enter the entry time in HH:mm:ss order: ");
        String EntryTime = sc2.nextLine();

        Scanner sc3 = new Scanner(System.in);
        System.out.print("Enter the exit time in HH:mm:ss order: ");
        String ExitTime = sc3.nextLine();

        Scanner sc4 = new Scanner(System.in);
        System.out.print("Enter the entry date in DD/MM/YYYY order: ");
        String Date = sc4.nextLine();

        String time1 = EntryTime;
        String time2 = ExitTime;

        try {
            int charge;
            int newCharge=0;
            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("HH:mm:ss");

            Date date1 = simpleDateFormat.parse(time1);
            Date date2 = simpleDateFormat.parse(time2);

            long differenceInMilliSeconds = Math.abs(date2.getTime() -
date1.getTime());
            long differenceInHours = (differenceInMilliSeconds / (60 * 60 *
1000)) % 24;
            long differenceInMinutes = (differenceInMilliSeconds / (60 * 1000))
% 60;
            long differenceInSeconds = (differenceInMilliSeconds / 1000) % 60;

            int l1 = (int) differenceInHours;

            if(l1==00){
                charge = 3000;
            }
            else if((l1-3)>0){
                charge = (3*300) + (l1-3)*100;
            }
            else {
                charge = (l1 * 100);
            }

            Scanner ch = new Scanner(System.in);
            System.out.println(" ");
            System.out.println("Enter the vehicle type");
```

```
            System.out.println("[1] Car");
            System.out.println("[2] Van");
            System.out.println("[3] Motor Bike");
            System.out.println("[4] Mini Bus");
            System.out.println("[5] Mini Lorry");
            int ch2 = ch.nextInt();

            if(ch2 == 1)
                newCharge = charge*1;
            else if(ch2 == 2)
                newCharge = charge*2;
            else if(ch2 == 3)
                newCharge = charge/3;
            if(ch2 == 4)
                newCharge = charge*3;
            if(ch2 == 5)
                newCharge = charge*3;

            System.out.println(" ");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~");
            System.out.println("                                CAR PARK
");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~");
            System.out.println("                                INVOICE
");

System.out.println("-----------------------------------------------------------
--");
            System.out.println("            Date               : " + Date);
            System.out.println("            Starting Time      : " + EntryTime);
            System.out.println("            Ending Time        : " + ExitTime);
            System.out.println("            Vehicle Reg.No     : " + regno);
            System.out.println("           Vehicle parked for : " +
differenceInHours + " hours " + differenceInMinutes + " minutes " +
differenceInSeconds + " Seconds. ");

System.out.println("-----------------------------------------------------------
--");
            System.out.println("                                CHARGE
");

System.out.println("-----------------------------------------------------------
--");
```

```
            System.out.println("          Total charge          : " + "LKR " +
newCharge);

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~");
            System.out.println("                                    THANK YOU!
");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void printList() {
        for (int i = List.size()-1 ; i>=0 ; i--) {
            Vehicle obj = List.get(i);
            if (obj instanceof Car) {
                System.out.println("VEHICLE IS A CAR AND REGISTER NUMBER IS: "
+ obj.vehicleRegNo);
            } else if (obj instanceof Van) {
                System.out.println("VEHICLE IS A VAN AND REGISTER NUMBER IS: "
+ obj.vehicleRegNo);
            } else if (obj instanceof MiniBus) {
                System.out.println("VEHICLE IS A MINI BUS AND REGISTER NUMBER
IS: " + obj.vehicleRegNo);
            } else if (obj instanceof MiniLorry) {
                System.out.println("VEHICLE IS A MINI LORRY AND REGISTER NUMBER
IS: " + obj.vehicleRegNo);
            } else if (obj instanceof MotorBike) {
                System.out.println("VEHICLE IS A MOTOR BIKE AND REGISTER NUMBER
IS: " + obj.vehicleRegNo);
            } else {
                System.out.println("INVALID VEHICLE TYPE!");
            }
        }

    }
    public void printVehicle(){

        for (int i = List.size()-1 ; i>=0 ; i--) {
            Vehicle obj = List.get(0);
            ltv = (obj.vehicleRegNo);
        }
        System.out.println("REGISTER NUMBER OF THE VEHICLE PARKED FOR LONGEST
TIME IS " + ltv);
```

```
        for (int i = 0 ; i<=List.size()-1 ; i++) {
            Vehicle obj = List.get(i);
            stv = (obj.vehicleRegNo);
        }
        System.out.println("REGISTER NUMBER OF THE LAST PARKED VEHICLE IS " +
stv);
    }
}
```

Vehicle.java

```
package ParkingSystem;

public class Vehicle {
    protected String vehicleRegNo;
    protected String EntryTime;
    public Vehicle() {
        super();
        vehicleRegNo = null;
        EntryTime = null;
    }
}
class Car extends Vehicle{
    public Car(String vehicleRegNo,String EntryTime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.EntryTime = EntryTime;
    }
}
class Van extends Vehicle{
    public Van(String vehicleRegNo,String EntryTime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.EntryTime = EntryTime;
    }
}
class MiniBus extends Vehicle{
    public MiniBus(String vehicleRegNo,String EntryTime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.EntryTime = EntryTime;
    }
}
class MiniLorry extends Vehicle{
```

```
    public MiniLorry(String vehicleRegNo,String EntryTime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.EntryTime = EntryTime;
    }
}
class MotorBike extends Vehicle{
    public MotorBike(String vehicleRegNo,String EntryTime) {
        super();
        this.vehicleRegNo = vehicleRegNo;
        this.EntryTime = EntryTime;
    }
}
```