A Project Report

On

PARKING LOT

For

Object Oriented Analysis and Design using Java
(20B12CS334)

Submitted by:

Vaishali Ranjan

Siddhant Singh

Jaypee Institute of Information Technology University, Noida

December, 2022

Chapter 1: Introduction

Our project focusses on how we can solve the parking lot problem that we face everyday. Parking has become a very a tedious job in today's busy world and in order to save our precious time and energy we have created an interface using Java.

We will solve this problem after following certain steps. As the customer enters the parking lot, customer waits for a green signal to check the availability of a vacant parking space in the parking lot. Parking automated system will check for the available. This process sometimes becomes too much time taking and we have devised this project to solve this issue.

When a car enters the parking lot, we want to have a ticket issued to the driver. The ticket issuing process includes us documenting the registration number (number plate) and the colour of the car and allocating an available parking slot to the car before actually handing over a ticket to the driver (we assume that our customers are nice enough to always park in the slots allocated to them).

The customer should be allocated parking slot which is nearest to the entry. At the exit the customer returns the ticket which then marks the slot they were using as being available. They can use both credit card and cash as the method of payment. The parking system have two separate entry and exit points.

Chapter 2: Functional and Non Functional Requirements

Functional Requirements:

Functional Requirements describes the system requirements of features, functions and facility. Functional Requirements essentially specifies something the system should be able to do or provide for users.

1. Customer will be able to check the availability in the parking lot.
2. Customer can reserve a parking slot
3. A ticket will be generated for the customer. The ticket will contain several information like vehicle number, date and time of arrival and available parking slot number.
4. the customer will get to know about his own parking slot number through ticket only.
5. Administrator can perform a standard administrative function such as generating ticket, updating parking lots and calculating final amount.

Non-functional requirements:

Non-functional requirements is a description of a variety of quality factors or attributes, which affect the functionality's effectiveness.

1. Performance- The system shall be to provide continuous updating of parking slot once a car enter/exits the parking lot. The system shall be able to provide the user with the information that contains the most recent updates of the parking spaces.
2. Reliability- Reliability is defined as providing the user up to date, correct information when they need it. Information is considered correct when the parking spaces are accurately reported.
3. Maintainability- This feature indicates the average time and ease and rapidity with which a system can be restored after a failure.
4. Data integrity- Data integrity refers to maintaining and assuring data accuracy and consistency over its entire lifecycle. Maintaining data of the date and time of the vehicle enters the parking slot is much needed to generate the amount the customer has to pay.
5. Environmental- The system will not cause physical harm to users and non-users. It will also not get affected by any external factors.

Chapter 4: Use Case Diagram

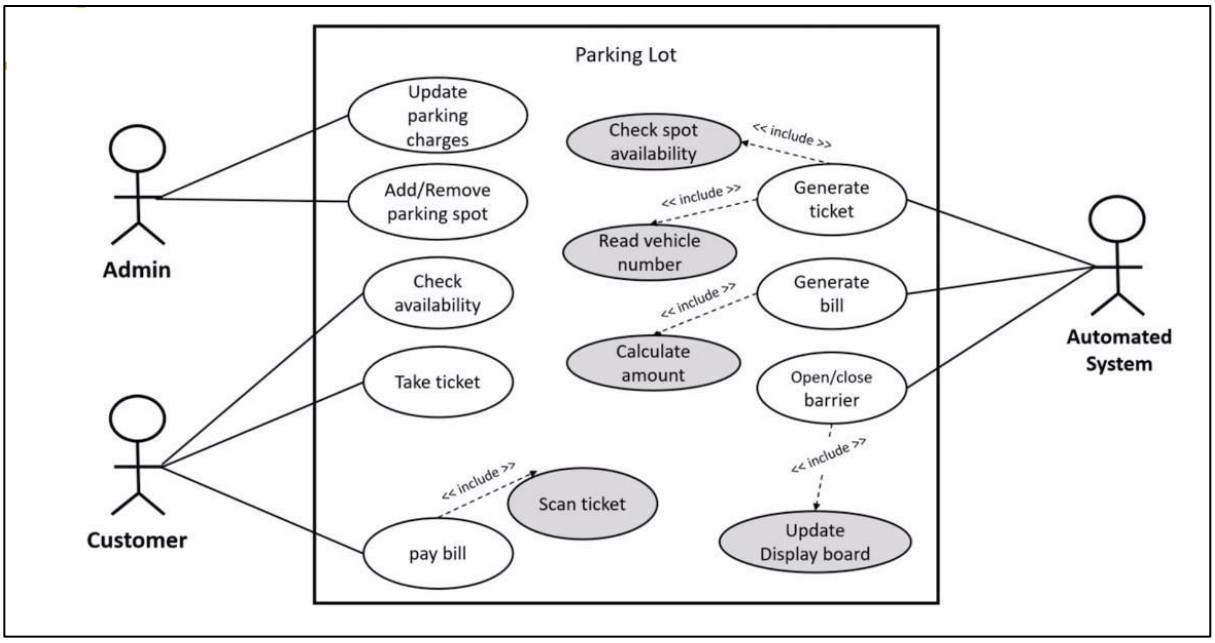

Figure 4.1: Use Case Diagram

4.2 Program Code

Class ParkingLot:

```
import java.util.ArrayList;

public class ParkingLot {
private String name;
private int capacity;
private String Location;

   public double getParkingCharges() {
return parkingCharges;
   }

   private double parkingCharges;



   private Admin admin;

   private AutomatedSystem automatedSystem;
private ArrayList<ParkingSpot> parkingSpots;
private ArrayList<ParkingSpot> availableSpots;
private DisplayBoard displayBoard;



   public ParkingLot(String name, int capacity, String location, double parkingCharges, AutomatedSystem automatedSystem, DisplayBoard
displayBoard) {
       this.name = name;
this.capacity = capacity;
Location = location;
       this.parkingCharges = parkingCharges;
this.automatedSystem = automatedSystem;
this.automatedSystem.setParkingLot(this);
this.displayBoard = displayBoard;
       this.parkingSpots = createParkingSpots(capacity);
       availableSpots = (ArrayList<ParkingSpot>) parkingSpots.clone();
   }
   private ArrayList<ParkingSpot> createParkingSpots(int
capacity){        ArrayList<ParkingSpot> tempList = new
ArrayList<>();        for (int i = 0; i <capacity ; i++) {
```

```
            ParkingSpot temp = new ParkingSpot("MPL "+i);
tempList.add(temp);
        }
        return tempList;
    }

    public DisplayBoard getDisplayBoard() {
return displayBoard;
    }

    public ParkingSpot getAvailableSpot() {
return availableSpots.remove(0);
    }

    public int getAvailability(){
return availableSpots.size();
    }

    public void updateParkingCharges(double newCharges){
this.parkingCharges = newCharges;
    }
    public void setAdmin(Admin admin) {
this.admin = admin;
    }

    public static void main(String[] args) {
        DisplayBoard displayBoard = new DisplayBoard();
        AutomatedSystem automatedSystem = new AutomatedSystem(1);
        ParkingLot myParkingLot = new ParkingLot("MyParkingLot",4, "Sec-135,Noida",50,automatedSystem,displayBoard);

        System.out.println("********************************");
        System.out.println("*                                                    *");
        System.out.println("*                                                    *");
        System.out.println("*                                                    *");
        System.out.println("*                WELCOME TO JIIT PARKING LOT                *");
        System.out.println("*                                                    *");
        System.out.println("*              (Your Vehicle is Safe with Us)              *");
        System.out.println("*                                                    *");
        System.out.println("*                                                    *");
        System.out.println("********************************\n\n\n\n");
```

```
System.out.println("Capacity: ");
// Checking the availability of this parking Lot

int availavility = myParkingLot.getAvailability();
System.out.println(availavility);

System.out.println("\n");

// Creating Customer
Customer vaishali = new Customer("Vaishali","UP85 AX 5454");

// Automated System Generating Ticket
Ticket vaishaliTicket = myParkingLot.automatedSystem.generateTicket(vaishali);

// Printing Ticket Info
System.out.println(vaishaliTicket);

System.out.println("\n");

// Creating Customer
Customer asmi = new Customer("Asmi","UP86 AB 9999");

// Automated System Generating Ticket
Ticket asmiTicket = myParkingLot.automatedSystem.generateTicket(asmi);

// Printing Ticket Info
System.out.println(asmiTicket);

System.out.println("\n");

 // Creating Customer
 Customer tanmay = new Customer("Tanmay","UP86 DK 6810");

 // Automated System Generating Ticket
 Ticket tanmayTicket = myParkingLot.automatedSystem.generateTicket(tanmay);

 // Printing Ticket Info
 System.out.println(tanmayTicket);

 System.out.println("\n");

 // Creating Customer
Customer pulkit = new Customer("Pulkit","UP86 ACC 5473");
```

```
        // Automated System Generating Ticket
        Ticket pulkitTicket = myParkingLot.automatedSystem.generateTicket(pulkit);

        // Printing Ticket Info
        System.out.println(pulkitTicket);

        System.out.println("\n");

        // Creating Customer
        Customer shantanu = new Customer("Shantanu","UP86 AB 4567");

        // Automated System Generating Ticket
        Ticket shantanuTicket = myParkingLot.automatedSystem.generateTicket(shantanu);

        // Printing Ticket Info
        System.out.println(shantanuTicket);
        //Exceed Capacity

    }

}
```

Class AutomatedSystem:

```
import java.time.Duration; import
java.time.LocalDateTime;
import java.util.HashMap;

public class AutomatedSystem {
   private int id;



   private ParkingLot parkingLot;
HashMap<Integer,Ticket> currentTickets;

   public AutomatedSystem(int id){
      this.id = id;
      currentTickets = new HashMap<>();
   }

   public ParkingLot getParkingLot() {
      return parkingLot;
   }

   public void setParkingLot(ParkingLot parkingLot) {
      this.parkingLot = parkingLot;
   }
   private ParkingSpot fetchAvailableSpot(){
return this.parkingLot.getAvailableSpot();    }
```

```
    public Ticket generateTicket(Customer customer){
ParkingSpot availableSpot = fetchAvailableSpot();
        Vehicle vehicle = customer.getVehicle();
Ticket ticket = new Ticket(vehicle,availableSpot);
currentTickets.put(ticket.getId(),ticket);
        return ticket;
    }
    public Ticket scanTicket(){
        // Code for scanning the ticket and return ticketId
// Here we are assuming that scanned ticket id is 1234
return currentTickets.get(1234);
    }

    public double caluclateCharges(){
Ticket ticket= scanTicket();
         long duration = Duration.between(LocalDateTime.now(),ticket.getArrivalTime()).toHours();
double charges = duration * parkingLot.getParkingCharges();          return charges;
    }

    public void openEntryBarrier(){
// Code for opening Entry Barrier
        this.parkingLot.getDisplayBoard().update(Status.FULL);
    }
    public void closeEntryBarrier(){
// Code for closing Entry Barrier
    }
    public void openExitBarrier(){
// Code for opening Entry Barrier
        this.parkingLot.getDisplayBoard().update(Status.AVAILABLE);
    }
    public void closeExitBarrier(){
// Code for closing Entry Barrier
    }

}
```

Class Customer:

```
public class Customer {
private String name;
    private Vehicle vehicle;

    public Customer(String name, String vehicleNumber) {
        this.name = name;
        this.vehicle = new Vehicle(vehicleNumber);
    }

    public Vehicle getVehicle() {
        return vehicle;
    }
}
```

Class Admin:

```
public class Admin {
    String name;
}
```

Enum Status:

```
public enum Status {
```

```
    AVAILABLE, FULL;
}
```

Class DisplayBoard:

```
public class DisplayBoard {
    public Status status;


    public DisplayBoard() {
        this.status = Status.AVAILABLE;

    }

    public void update(Status newStatus){
        this.status = newStatus;

    }
}
```

Class ParkingSpot:

```
public class ParkingSpot { String

spotNumber; public ParkingSpot(String

spotNumber) { this.spotNumber =

spotNumber;

} public String getSpotNumber()

{ return spotNumber;

}

}
```

Class Ticket:

```
import java.sql.Time; import
java.time.LocalDateTime;
import java.util.Date;

public class Ticket {
static int idCounter=0;
    private int id;
    private String vehicleRegistrationNumber;     private LocalDateTime arrivalTime;
    private String parkingSpotNumber;

    public int getId() {
return id;
    }

    public Ticket (Vehicle vehicle, ParkingSpot availableSpot) {
this.id = ++idCounter;         this.vehicleRegistrationNumber =
vehicle.getVehicleNumber();         this.parkingSpotNumber =
```

```
availableSpot.getSpotNumber();        this.arrivalTime =
LocalDateTime.now();
    }

    public LocalDateTime getArrivalTime() {
        return arrivalTime;
    }

    @Override     public
String toString() {
return "Ticket{" +
                "id=" + id +
                ", vehicleRegistrationNumber='" + vehicleRegistrationNumber + '\'' +
                ", arrivalTime=" + arrivalTime +
                ", parkingSpotNumber='" + parkingSpotNumber + '\'' +
                '}';
    }
}
```

Class Vehicle:

```
public class Vehicle {
String vehicleNumber;

    public Vehicle(String vehicleNumber) {
        this.vehicleNumber = vehicleNumber;
    }

    public String getVehicleNumber() {
        return vehicleNumber;
    }
}
```

Chapter 5: Results

Welcome Window and Capacity

```
********************************************************************************
*                                                                              *
*                                                                              *
*                                                                              *
*                          WELCOME TO JIIT PARKING LOT                         *
*                                                                              *
*                         (Your Vehicle is Safe with Us)                       *
*                                                                              *
*                                                                              *
********************************************************************************



Capacity:
4
```

Customer Created

```
// Creating Customer
Customer vaishali = new Customer(name: "Vaishali",vehicleNumber: "UP85 AX 5454");

// Automated System Generating Ticket
Ticket vaishaliTicket = myParkingLot.automatedSystem.generateTicket(vaishali);

// Printing Ticket Info
System.out.println(vaishaliTicket);

System.out.println(x: "\n");

// Creating Customer
Customer asmi = new Customer(name: "Asmi",vehicleNumber: "UP86 AB 9999");

// Automated System Generating Ticket
Ticket asmiTicket = myParkingLot.automatedSystem.generateTicket(asmi);

// Printing Ticket Info
System.out.println(asmiTicket);

System.out.println(x: "\n");

// Creating Customer
Customer tanmay = new Customer(name: "Tanmay",vehicleNumber: "UP86 DK 6810");

// Automated System Generating Ticket
Ticket tanmayTicket = myParkingLot.automatedSystem.generateTicket(tanmay);

// Printing Ticket Info
System.out.println(tanmayTicket);
```

```
// Creating Customer
Customer pulkit = new Customer(name: "Pulkit",vehicleNumber: "UP86 ACC 5473");

// Automated System Generating Ticket
Ticket pulkitTicket = myParkingLot.automatedSystem.generateTicket(pulkit);

// Printing Ticket Info
System.out.println(pulkitTicket);

System.out.println(x: "\n");

// Creating Customer
Customer shantanu = new Customer(name: "Shantanu",vehicleNumber: "UP86 AB 4567");

// Automated System Generating Ticket
Ticket shantanuTicket = myParkingLot.automatedSystem.generateTicket(shantanu);

// Printing Ticket Info
System.out.println(shantanuTicket);
//Exceed Capacity
```

Currently Parked Vehicle Information

```
Ticket{id=1, vehicleRegistrationNumber='UP85 AX 5454', arrivalTime=2022-12-21T09:50:59.381142300, parkingSpotNumber='MPL 0'}

Ticket{id=2, vehicleRegistrationNumber='UP86 AB 9999', arrivalTime=2022-12-21T09:50:59.384142900, parkingSpotNumber='MPL 1'}

Ticket{id=3, vehicleRegistrationNumber='UP86 DK 6810', arrivalTime=2022-12-21T09:50:59.388142300, parkingSpotNumber='MPL 2'}

Ticket{id=4, vehicleRegistrationNumber='UP86 ACC 5473', arrivalTime=2022-12-21T09:50:59.398144300, parkingSpotNumber='MPL 3'}
```

Exceeding Capacity

```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
        at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
        at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
        at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:248)
        at java.base/java.util.Objects.checkIndex(Objects.java:372)
        at java.base/java.util.ArrayList.remove(ArrayList.java:536)
        at ParkingLot.getAvailableSpot(ParkingLot.java:50)
        at AutomatedSystem.fetchAvailableSpot(AutomatedSystem.java:26)
        at AutomatedSystem.generateTicket(AutomatedSystem.java:30)
        at ParkingLot.main(ParkingLot.java:137)
```

Chapter 6: Conclusion

Parking has become a very a tedious job in today's busy world and We solved this problem with following certain steps. As the customer enters the parking lot, customer waits for a green signal to check the availability of a vacant parking space in the parking lot. Parking automated system will check for the available. This process sometimes becomes too much time taking and we have created an interface using Java to solve this issue in order to save our precious time and energy.

Chapter 7: References

[1] https://www.researchgate.net/publication/335807455_Smart_Parking_Management_System

[2] https://www.researchgate.net/figure/Use-Case-Diagram-of-Car-Parking-InformationSystem_fig2_337402229

[3] https://www.w3schools.com/java/java_classes.asp

[4] https://www.w3schools.com/java/java_class_attributes.asp

[5] https://beginnersbook.com/2013/03/oops-in-java-encapsulation-inheritance-polymorphismabstraction/