Sir Syed University of Engineering and Technology

Software Engineering Department

Software Design and Architecture (SWE-208L)

Project Report

Topic: Rent A Car

Group Members:

Aleena Waqar (2020-SE-204)

Sidra Khan (2020-SE-198)

Karishma Zaheer (2020-SE-219)

Syeda Maira Hussain (2020-SE-210)

1. Introduction

1.1 Purpose:

The purpose of this product is to make renting a car easier. For anyone living within the country or out of the country. The product will tell if the car you entered is available or not. In this we use design pattern, mediator pattern which asks the end user for input and communicate with the main system.

1.2 Document Convention:

For this product, the coding we have done using JAVA language on Apache NetBeans. The architecture involves the customer entering the input to the server, the server searching for the required car, and the server responding back to the user.

1.3. Intended Audience:

This is for user to help solve renting a car. For the users to search the system for the car that they want. If the car is available then it will be booked otherwise it will print the message that “car out of stock.”

1.4. Product Scope:

If you are an online entrepreneur to launch an online car rental business to connect the local hosts an international customer. This business prototype helps your customer to search for their accommodation all over the world. The Car rental business is a best online booking system**.** Most of the car rental booking websites using the online car rental software which offers the old and low-cost cars which will ultimately save your time and money.

2. Overall Description

2.1 Product Perspective:

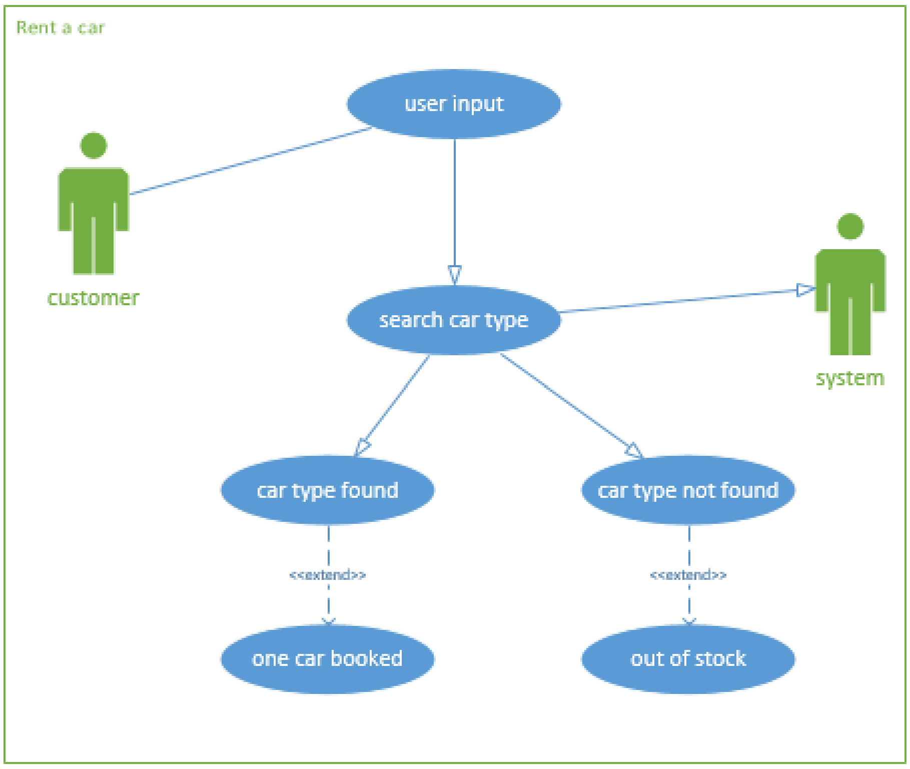
The product is rent a car which will allow user to rent a car easily. As said it above, we use design pattern which will interact with the end user and communicate with the main system. In this we have 3 cars types: luxury, sedan and sports. All three have their own separate car plants. Each car plant can book the car if they have the cars in the stock.

2.2. Product Functions:

We have used mediator design pattern which is part of behavioral pattern. The design pattern is needed when we don't want to expose the functionality to end user. So mediator will interact with the user and then the main system. In this ShowRoomMediator acts as a mediator that creates one customer object that is unique to each showroom and then take the request through all the car plants and match the car type that was requested.

2.3. User Classes and Characteristics:

The user classes include the user entered the input into the system and the system will check if the system has the required car.



2.4. Operating Environment:

We have the done the coding on Apache NetBeans version 12.6 and JDK version 8.

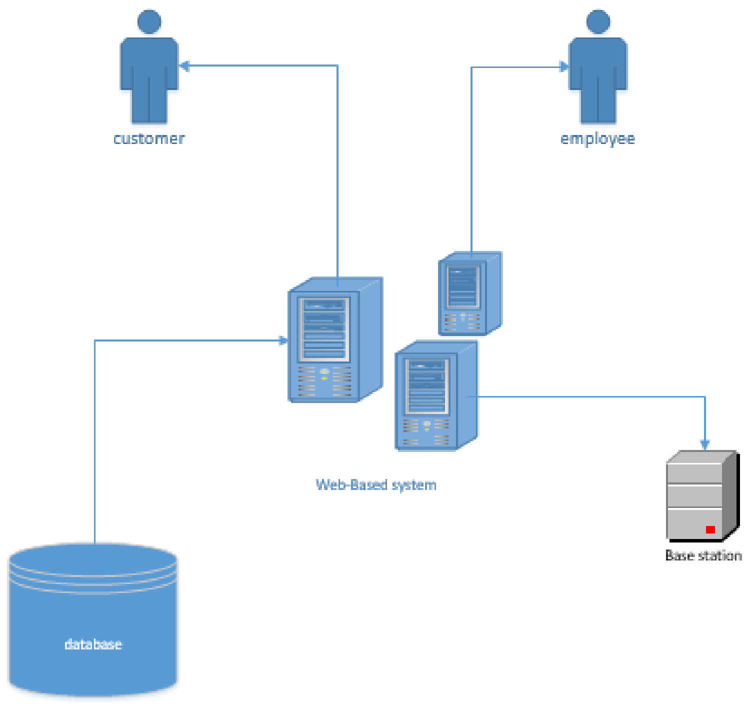
3. External Interface

3.1. User Interfaces:

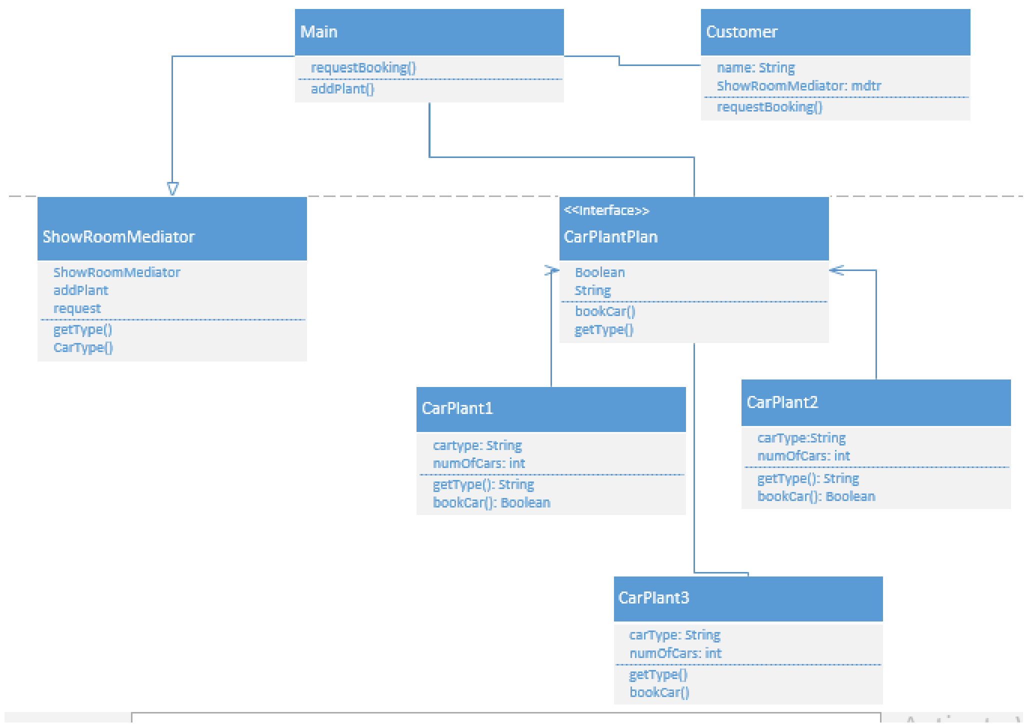
We create one mediator that interacts with the end user and finally communicates with the main system. In our example, we have 3 car types - luxury, sedan, sports. All 3 car tyypes have their separate car plants. Each car plant can book the car if they have the cars in the stock. ShowRoomMediator acts as a mediator, that creates one customer object that is unique to each showroom and then take the request and internally go through all the car plants and match the car type that was requested and book it.

3.2. User Documentation:

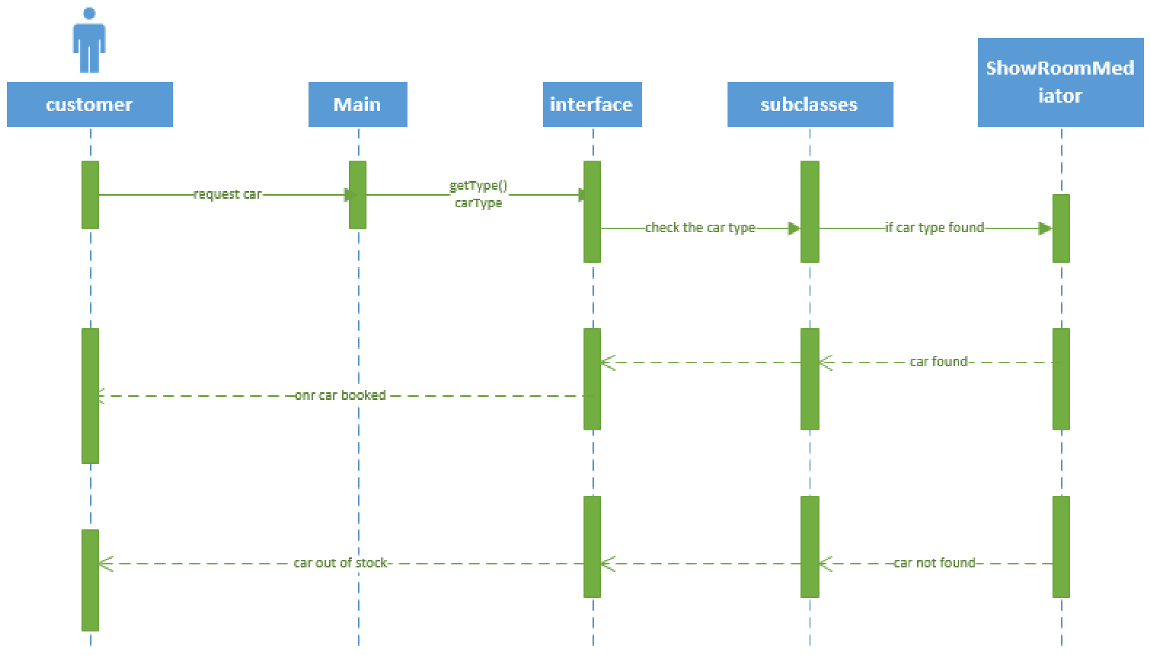
Architecture:



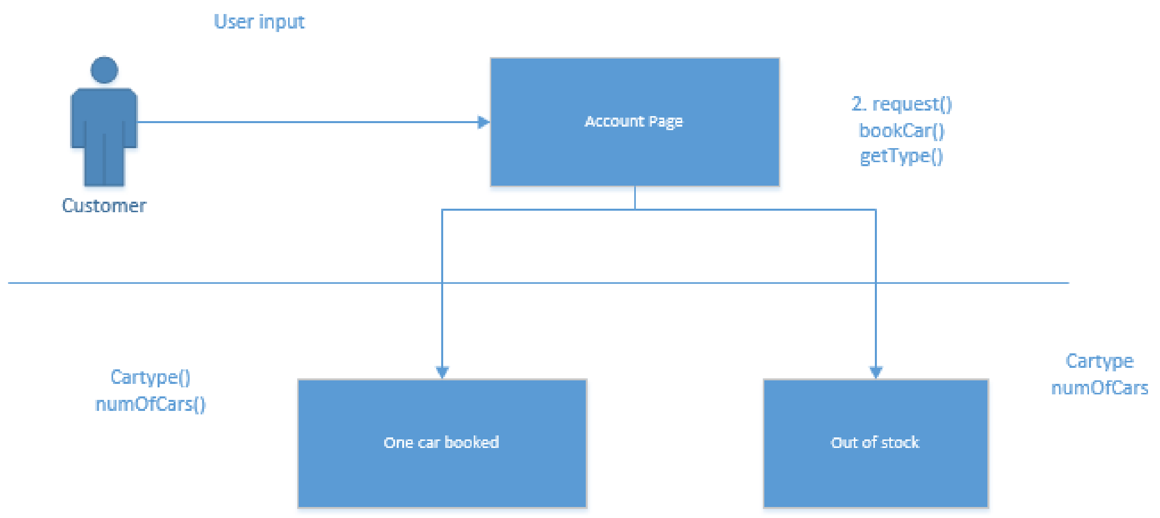
Class Diagram:



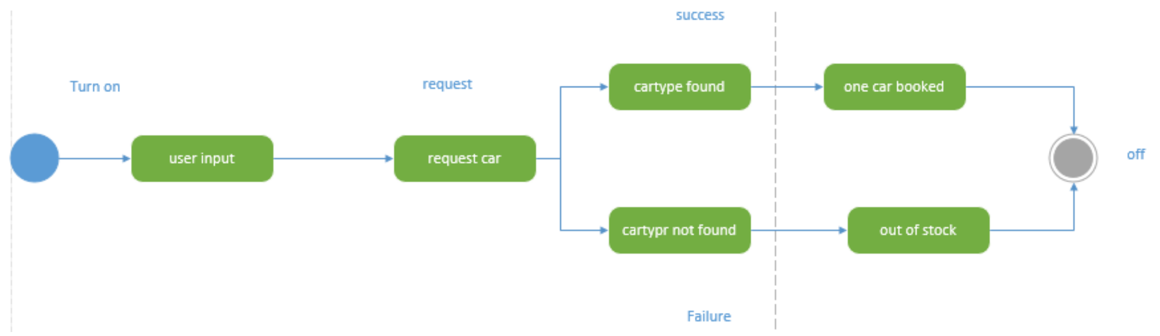
Sequence Diagram:



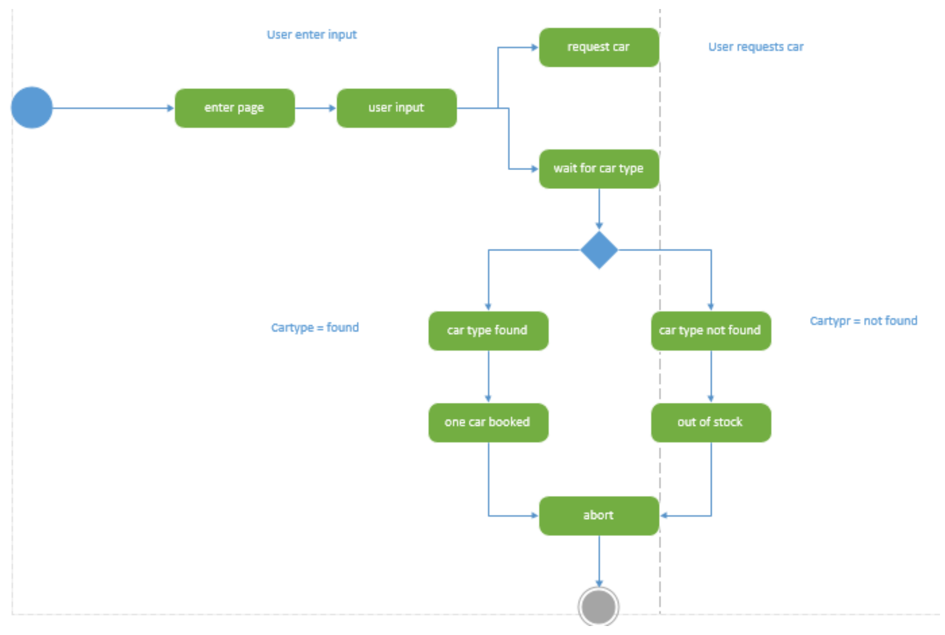
Communication or Collaboration Diagram:



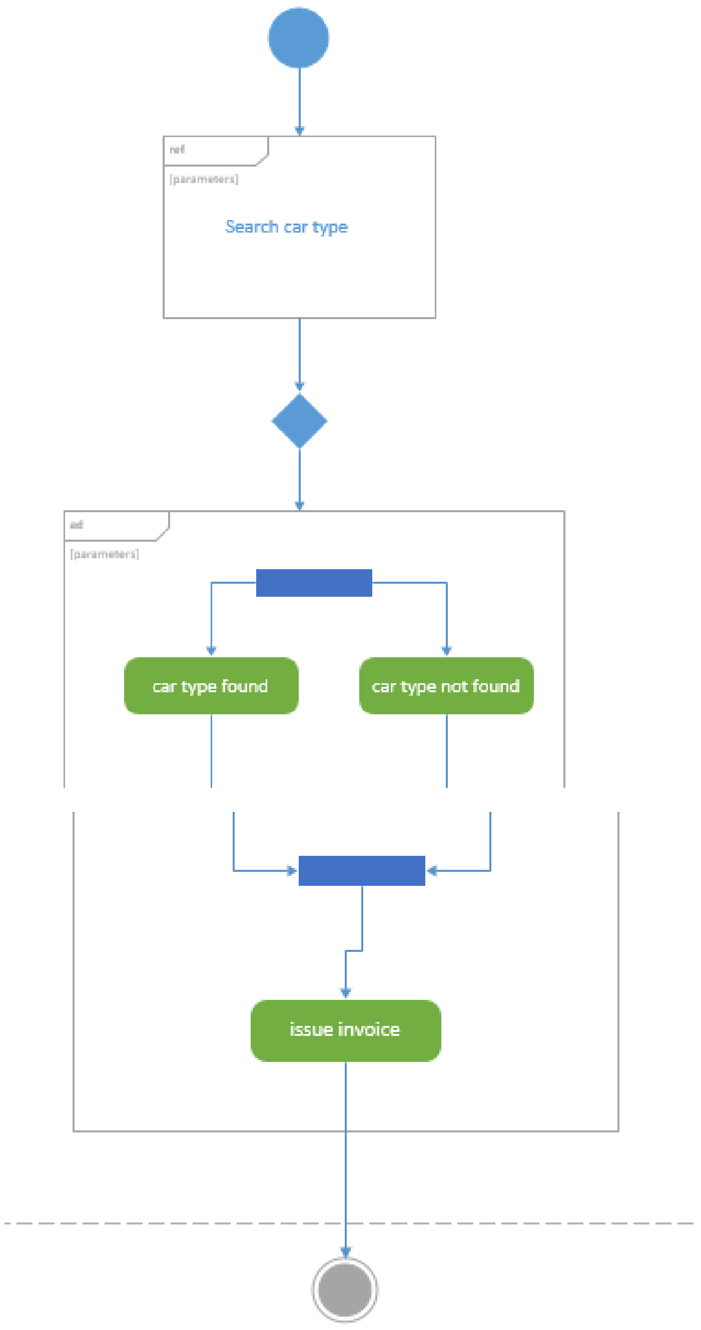
State Diagram:



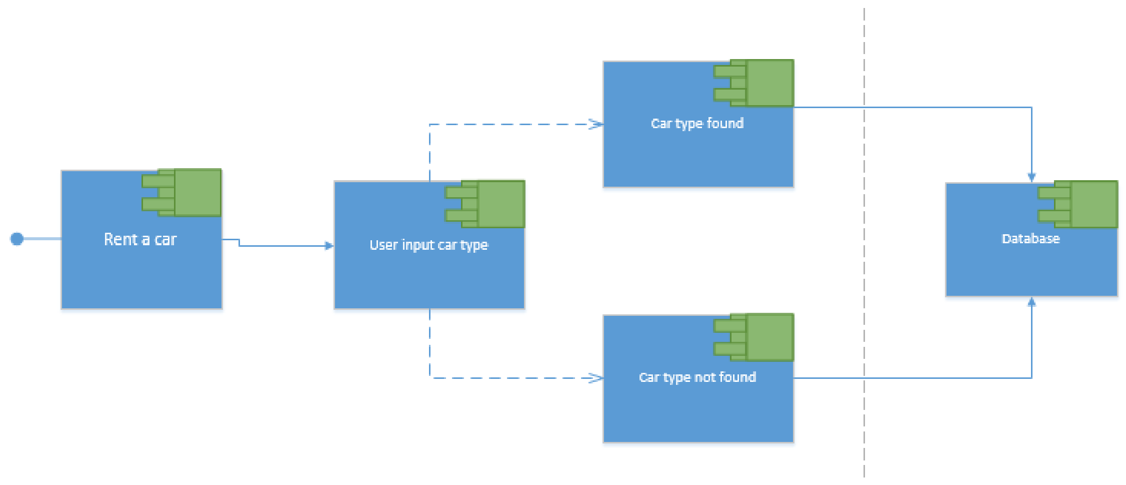
Activity Diagram:



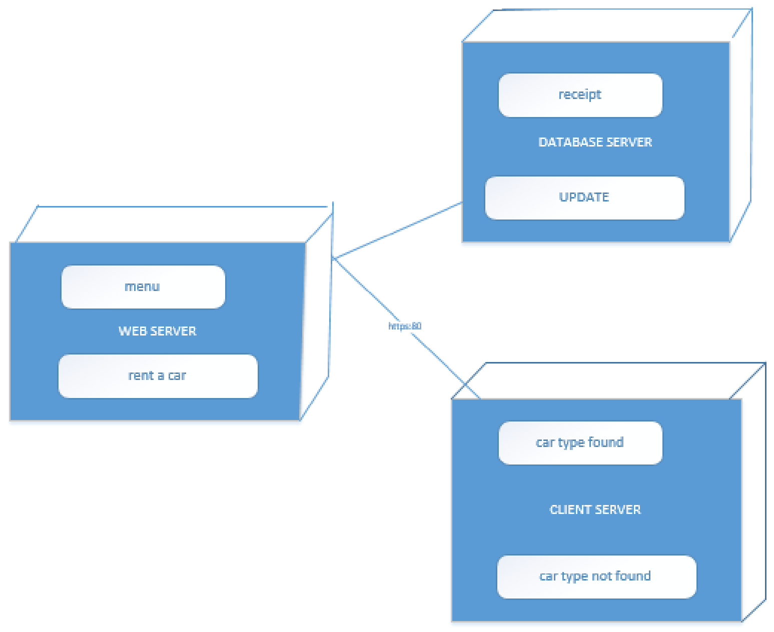
Interactive Overview Diagram:



Component Diagram:

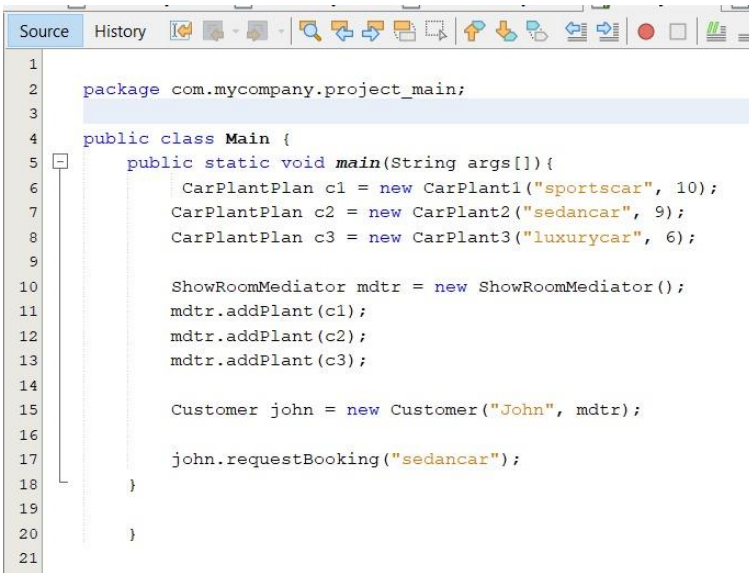


Deployment Diagram:



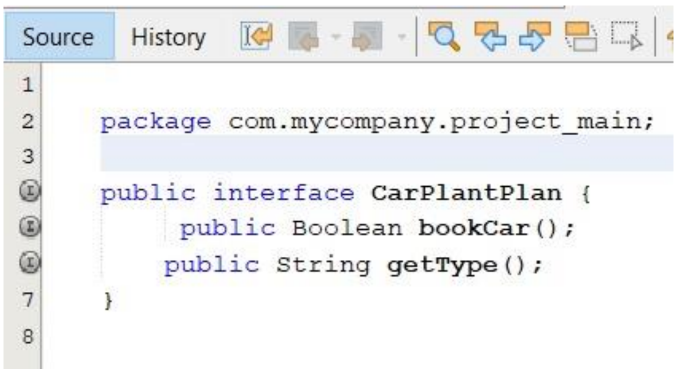
Source Code:

Main Class:



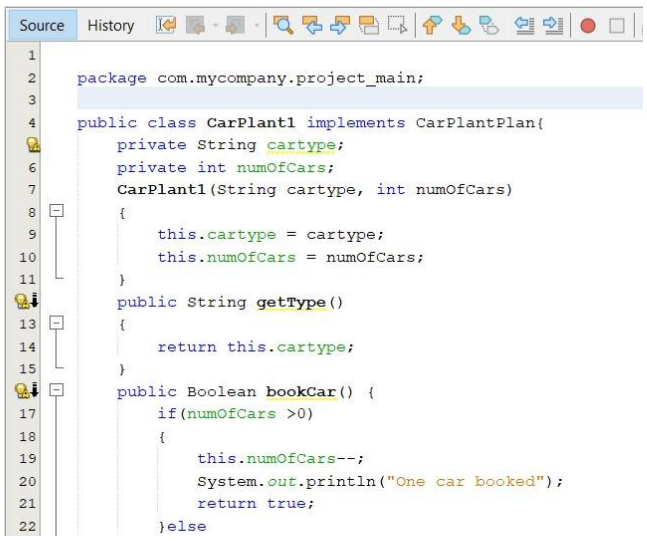
```
package com.mycompany.project_main;

public class Main {
    public static void main(String args[]){
         CarPlantPlan c1 = new CarPlant1("sportscar", 10);
        CarPlantPlan c2 = new CarPlant2("sedancar", 9);
        CarPlantPlan c3 = new CarPlant3("luxurycar", 6);

        ShowRoomMediator mdtr = new ShowRoomMediator();
        mdtr.addPlant(c1);
        mdtr.addPlant(c2);
        mdtr.addPlant(c3);

        Customer john = new Customer("John", mdtr);

        john.requestBooking("sedancar");
    }

    }
```

Interface:

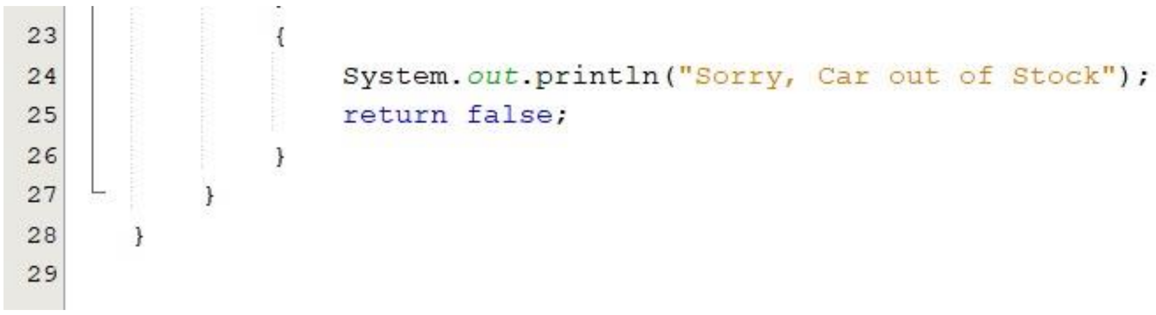


```
package com.mycompany.project_main;

public interface CarPlantPlan {
     public Boolean bookCar();
    public String getType();
}
```

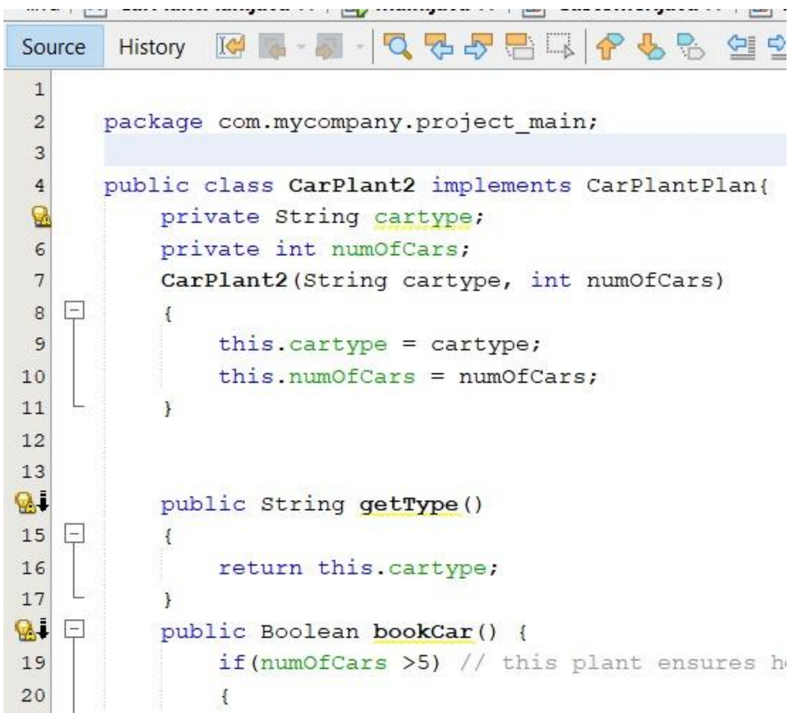
Sub Classes:



```
package com.mycompany.project_main;

public class CarPlant1 implements CarPlantPlan{
    private String cartype;
    private int numOfCars;
    CarPlant1(String cartype, int numOfCars)
    {
        this.cartype = cartype;
        this.numOfCars = numOfCars;
    }
    public String getType()
    {
        return this.cartype;
    }
    public Boolean bookCar() {
        if(numOfCars >0)
        {
            this.numOfCars--;
            System.out.println("One car booked");
            return true;
        }else
```

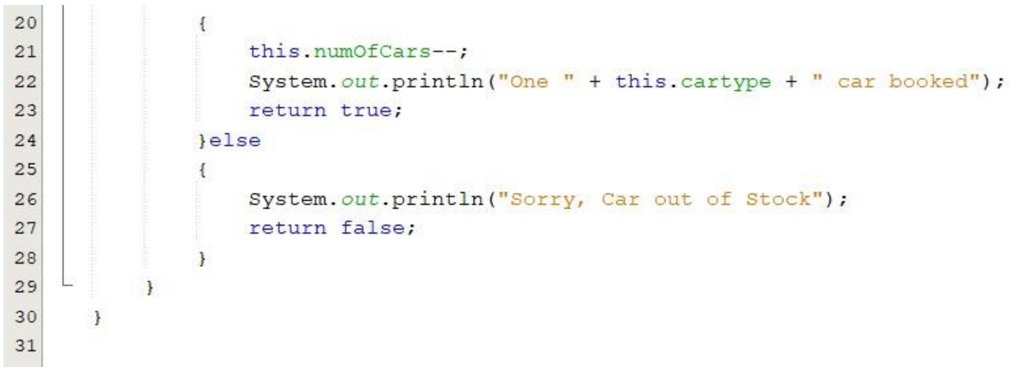


```
                {
                    System.out.println("Sorry, Car out of Stock");
                    return false;
                }
            }
    }
```

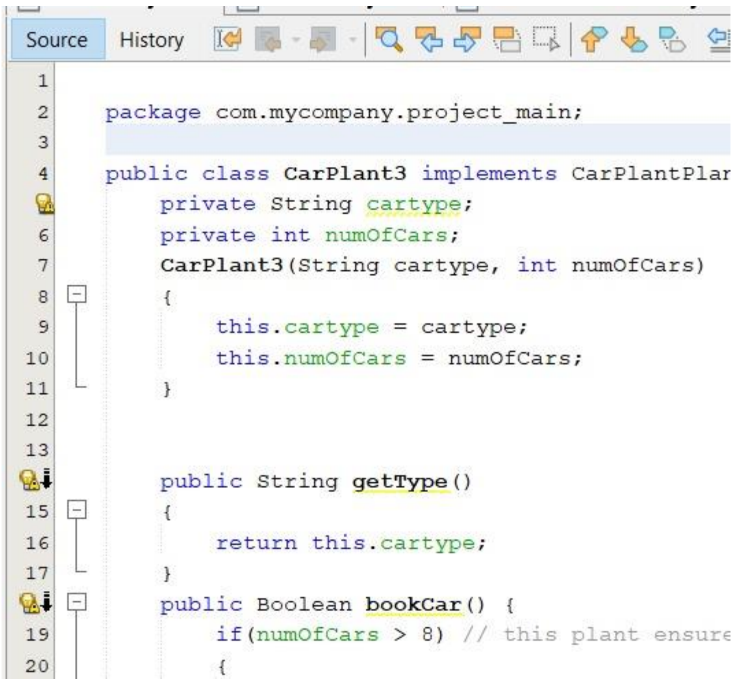


Source History

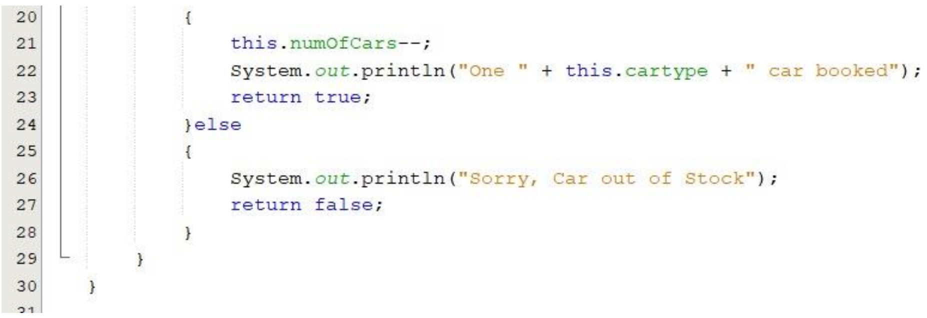
```

package com.mycompany.project_main;

public class CarPlant2 implements CarPlantPlan{
    private String cartype;
    private int numOfCars;
    CarPlant2(String cartype, int numOfCars)
    {
        this.cartype = cartype;
        this.numOfCars = numOfCars;
    }


    public String getType()
    {
        return this.cartype;
    }
    public Boolean bookCar() {
        if(numOfCars >5) // this plant ensures h
        {
```



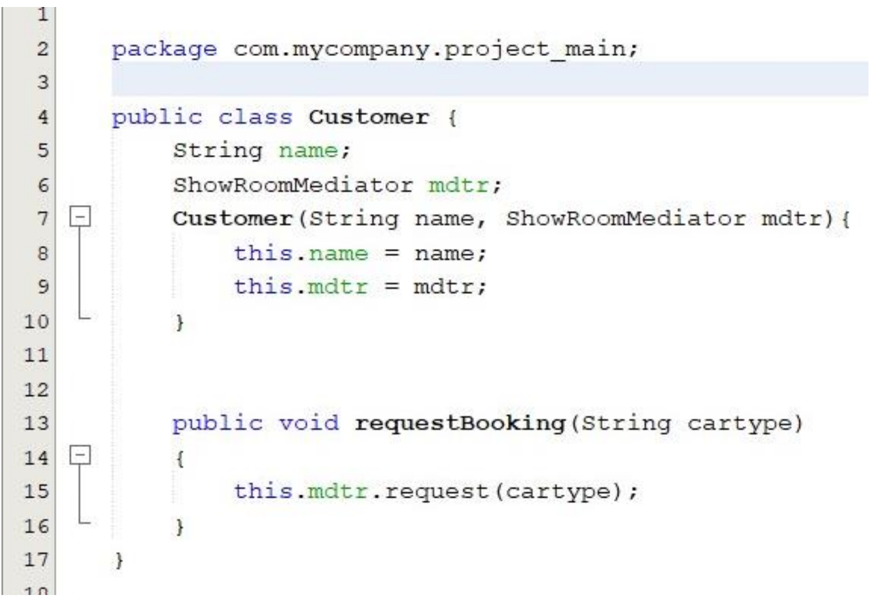
```
        {
            this.numOfCars--;
            System.out.println("One " + this.cartype + " car booked");
            return true;
        }else
        {
            System.out.println("Sorry, Car out of Stock");
            return false;
        }
    }
}
```



```

package com.mycompany.project_main;

public class CarPlant3 implements CarPlantPlar
    private String cartype;
    private int numOfCars;
    CarPlant3(String cartype, int numOfCars)
    {
        this.cartype = cartype;
        this.numOfCars = numOfCars;
    }


    public String getType()
    {
        return this.cartype;
    }
    public Boolean bookCar() {
        if(numOfCars > 8) // this plant ensure
        {
```

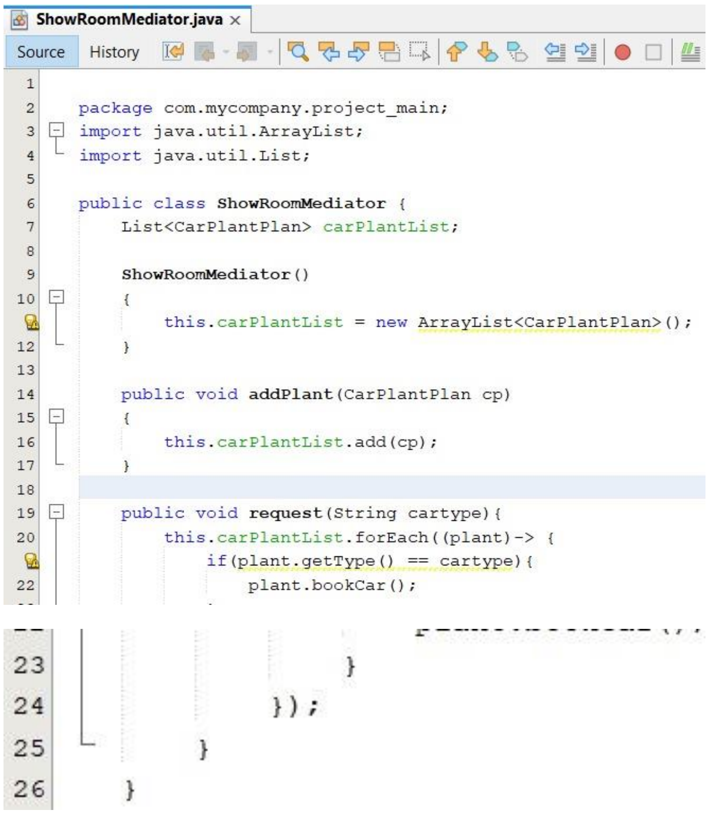


```
        {
            this.numOfCars--;
            System.out.println("One " + this.cartype + " car booked");
            return true;
        }else
        {
            System.out.println("Sorry, Car out of Stock");
            return false;
        }
    }
}
```



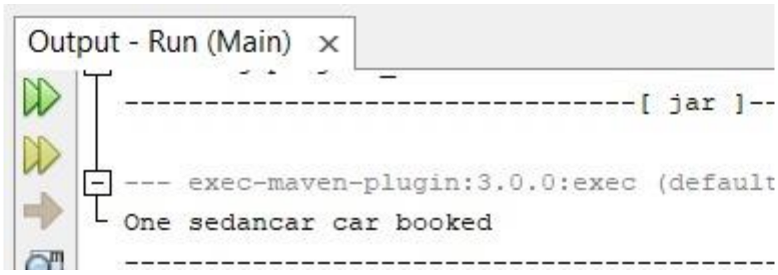
```

package com.mycompany.project_main;

public class Customer {
    String name;
    ShowRoomMediator mdtr;
    Customer(String name, ShowRoomMediator mdtr){
        this.name = name;
        this.mdtr = mdtr;
    }


    public void requestBooking(String cartype)
    {
        this.mdtr.request(cartype);
    }
}
```

MEDIATOR CLASS:

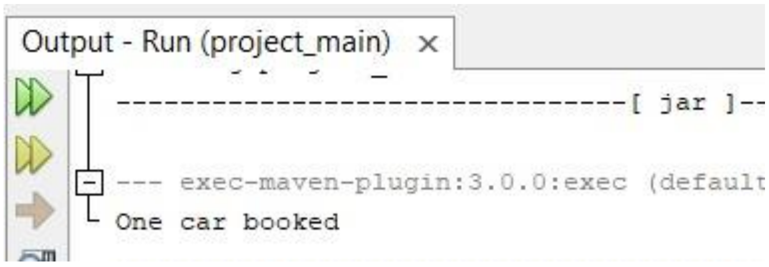


```java

package com.mycompany.project_main;
import java.util.ArrayList;
import java.util.List;

public class ShowRoomMediator {
    List<CarPlantPlan> carPlantList;

    ShowRoomMediator()
    {
        this.carPlantList = new ArrayList<CarPlantPlan>();
    }

    public void addPlant(CarPlantPlan cp)
    {
        this.carPlantList.add(cp);
    }

    public void request(String cartype){
        this.carPlantList.forEach((plant)-> {
            if(plant.getType() == cartype){
                plant.bookCar();
                }
            });
        }
    }
```

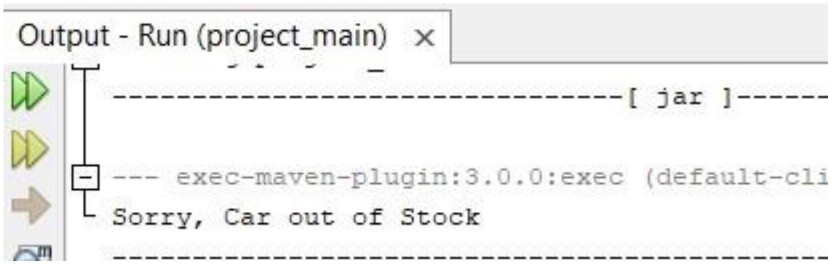
OUTPUT:



```
---------------------------------[ jar ]--

--- exec-maven-plugin:3.0.0:exec (default
One sedancar car booked
------------------------------------------
```



```
---------------------------------[ jar ]--

--- exec-maven-plugin:3.0.0:exec (default
One car booked
```



```
---------------------------------[ jar ]------

--- exec-maven-plugin:3.0.0:exec (default-cli
Sorry, Car out of Stock
----------------------------------------------
```