BSc. Artificial Intelligence & Data Science

Level 05

CM 2601

OBJECT ORIENTED PROGRAMMING

COURSEWORK REPORT-01

NIHAJ AMEER

IIT ID : 20200993

RGU USERNAME : 2017969

1) Characteristics of the OO design paradigm

a) Classes

The classes are such as the prototype of the houses.it contains all the details about the particular concept. we can create many objects from a class. The **class** keyword is used to create a class in java.

```
class ClassName {
```

Class keyword

1. Instance variable

Instance variable are used by objects to store their states. Variables that are defined **without** the **STATIC** keyword and are Outside any method declaration are Object-specific and are known as instance variables. They are called so because their values are instance-specific and are **not** shared among instances.If a class has an instance variable, then a new instance variable is created and initialized to a default value as part of the object creation of the class or subclass.

```
class Page {
  public String pageName;
  // instance variable with public access
  private int pageNumber;
  // instance variable with private access
}
```

2. Instance methods

Instance Methods are the group of codes that performs a particular task. Sometimes the program grows in size, and we want to separate the logic of the main method from other methods. A method is a function written inside the class. In java need to write a method inside some classes.

```
public void disp( ){
        int a= 10;
        System.out.println(a);
}
```

3. Static variable

The static variable can used to refer the common things. The static variable gets memory only once in the class area at the time of class loading.

```
class Student

{
    int rollno;
    String name;
    static String college ="GGGI"; //Static Variable gets memory once

    /*Constructor of Student class*/
    Student(int r,String n)
    {
        rollno = r;
        name = n;
    }

    /*Method For Displaying Student Details*/
    void display()
    {
        System.out.println(rollno+" "+name+" "+college); // print the value of roll no, name and college
    }

    public static void main(String args[])
    {
        Student s1 = new Student(101,"Gagan");
        Student s2 = new Student(102,"Raman");
        s1.display();  // call the display function using the s1 object
        s2.display();   // call the display function using the s2 object
    }

}
```

4. Static methods

Static methods can be called directly using class name. The syntax to call a static method.

```
className.methodName(); // Here, className is the name of a class.

For example:

   Student.add(); // Student is the name of class and add is a method.
```

5. **Constructers**

   The constructer is more similar to the method.it calls when an instance class created. when constructer is called the memory is allocated in the memory. It is a special type of method which is used to initialize the object.Every time an object is created using the new() keyword, at least one constructor is called.

```
1. //Java Program to create and call a default constructor
2. class Bike1{
3. //creating a default constructor
4. Bike1(){System.out.println("Bike is created");}
5. //main method
6. public static void main(String args[]){
7. //calling a default constructor
8. Bike1 b=new Bike1();
```

6. **Getters and Setters**

   Getter and setter methods used to access and manipulate the values of class fields. class fields are decorated with a private access specifier. If need to access them, public access specifiers are used with the getter and setter methods.

```
9. public class GetterSetterExample
```

```
10. {
11.     public salary;
12.
13.     public storeSalaryDB(int salary)
14.     {
15.         // code for storing the salary in the database
16.     }
17.
18.     // main method
19.     public static void main(String argvs[])
20.     {
21.       GetterSetterExample obj = new GetterSetterExample();
22.
23.       obj.salary = -50000;
24.
25.       // storing salary in database
26.       obj.storeSalaryDB(salary);
27.
28.     }
29. }
```

b) Objects

An object in Java is the physical as well as a logical entity, whereas, a class in Java is a logical entity only.

```
1.  //Java Program to illustrate how to define a class and fields
2.  //Defining a Student class.
3.  class Student{
4.   //defining fields
5.   int id;//field or data member or instance variable
6.   String name;
7.   //creating main method inside the Student class
8.   public static void main(String args[]){
9.    //Creating an object or instance
```

```
10. Student s1=new Student();//creating an object of Student
11. //Printing values of the object
12. System.out.println(s1.id);//accessing member through reference variable
13. System.out.println(s1.name);
14. }
15.}
```

c) OOP Concept

1. Abstraction

Abstraction method id define as the process of identifying only the required characteristics of an ignoring and the irrelevant details. an object differentiate it from other objects of similar type and also help in grouping the objects.

```
abstract class Shape {
    String color;

    // these are abstract methods
    abstract double area();
    public abstract String toString();

    // abstract class can have the constructor
    public Shape(String color)
    {
        System.out.println("Shape constructor called");
        this.color = color;
    }

    // this is a concrete method
    public String getColor() { return color; }
}
class Circle extends Shape {
    double radius;
```

2. Encapsulation

Encapsulation in Java is a *process of wrapping code and data together into a single unit*, for example, a capsule which is mixed of several medicines.Creating an encapsulated class in Java by making all the data members of the class private.Now we can use setter and getter methods to set and get the data in it.

```
1. //A Java class which is a fully encapsulated class.
2. //It has a private data member and getter and setter methods.
3. package com.javatpoint;
4. public class Student{
5. //private data member
6. private String name;
7. //getter method for name
8. public String getName(){
9. return name;
10. }
11. //setter method for name
12. public void setName(String name){
13. this.name=name
14. }
15. }
```

3. Inheritance

Inheritance is allows to create a new class from an existing class. The new class that is created is known as **subclass** (child or derived class) and the existing class from where the child class is derived is known as **superclass** (parent or base class).

```
class Animal {
  // methods and fields
}

// use of extends keyword
// to perform inheritance
class Dog extends Animal {

  // methods and fields of Animal
  // methods and fields of Dog
```

```
}
```

4. **Polymorphism**

In Polymorphism can create many classes that are related to each other by inheritance.Inheritance inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in 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");
  }
}
```

5. Abstract classes and Abstract methods

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.An abstract method is a method that is declared without an implementation.

```
public abstract class GraphicObject {
   // declare fields
   // declare nonabstract methods
   abstract void draw();
}
```

6. Interface

An interface can have methods and variables just like the class but the methods declared in interface are by default abstract (only method signature). Also, the variables declared in an interface are public, static & final by default.Interfaces are declared by specifying a keyword “interface”.

```
interface MyInterface
{
   /* All the methods are public abstract by default
    * As you see they have no body
    */
   public void method1();
   public void method2();
}
```

2) Different types of relationships between classes in the UML class diagram

Association- Indicate that instances of one model element are connected to instances of another model element

Dependencies-Indicate that a change to one model element can affect another model element

Eneralizations-Indicate that one model element is a specialization of another model element

Realizations-Indicate that one model element provides a specification that another model element implements

Transitions-Represent changes in state

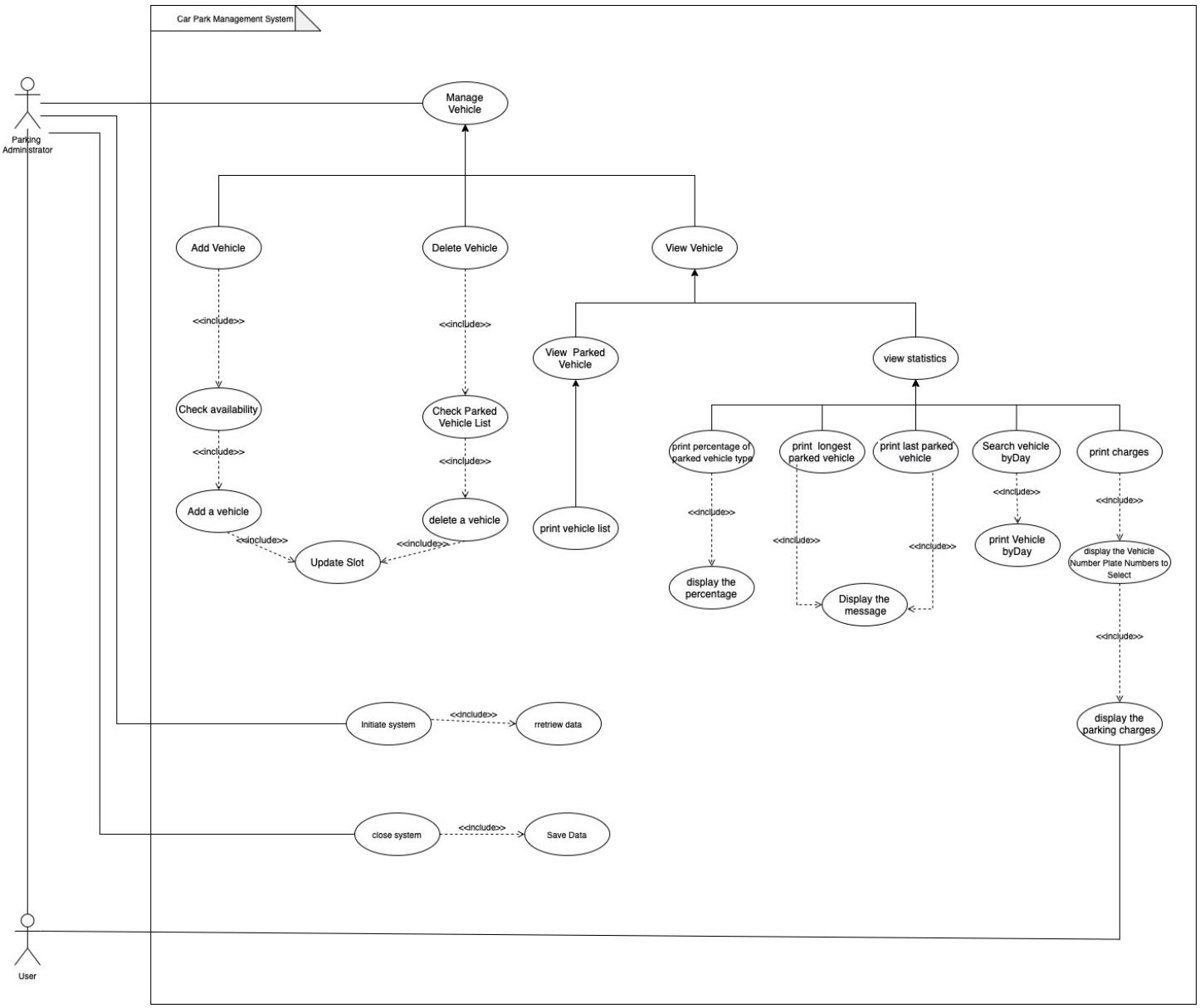
3) Design patterns and their relationship with the OOP design principals

Creational Design Patterns - Creational design patterns are concerned with the way of creating objects > Factory Pattern, Abstract Factory, Prototype, Singleton, Builder

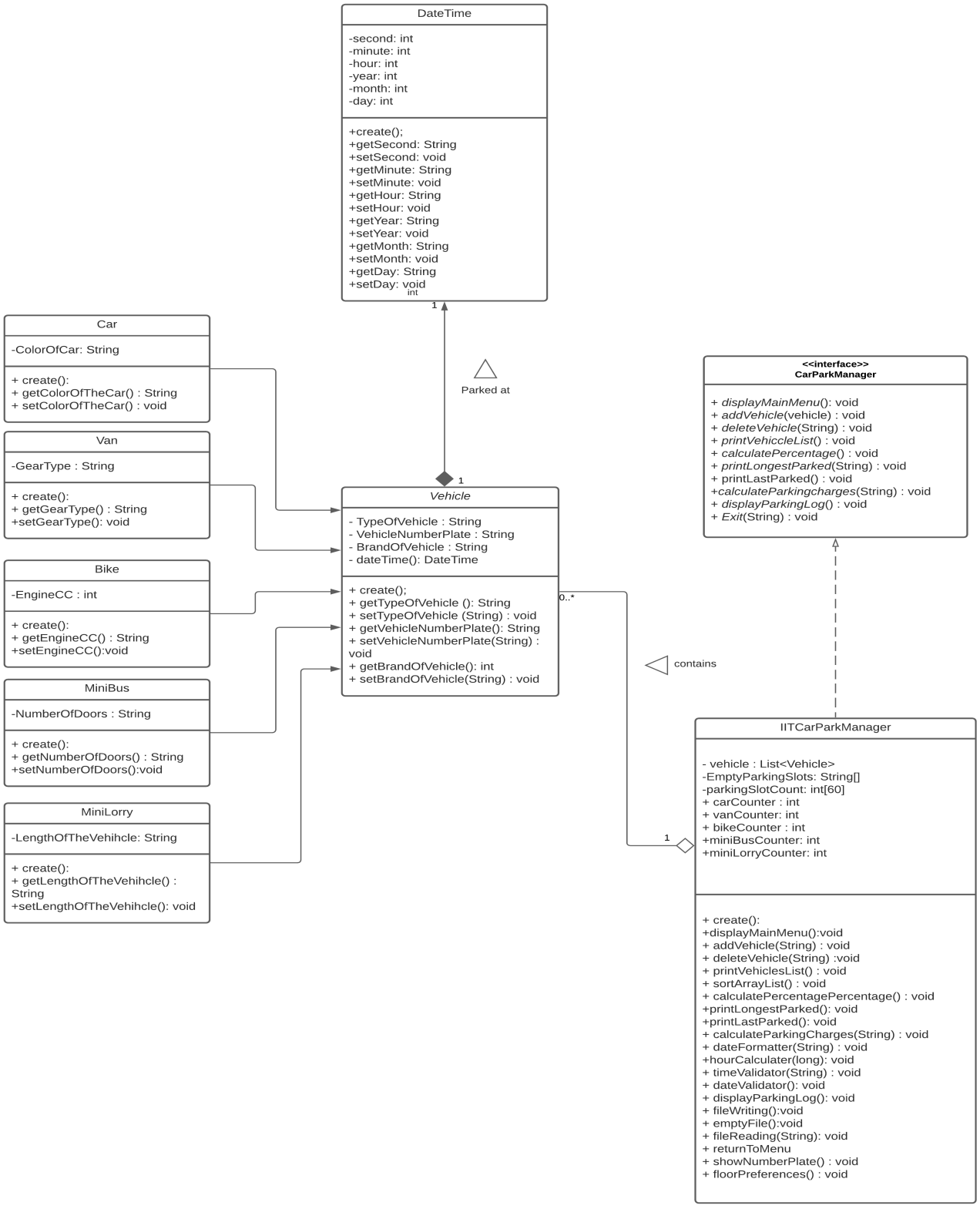
Structural Design pattern - Structural design patterns are concerned with how classes and objects can be composed, to form larger structures. > Adapter Pattern, Bridge, Composite, Decorator, Façade, Flyweight, proxy

Behavior Design Pattern - Behavioral design patterns are concerned with algorithms and the assignment of responsibilities between objects > Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, etc..

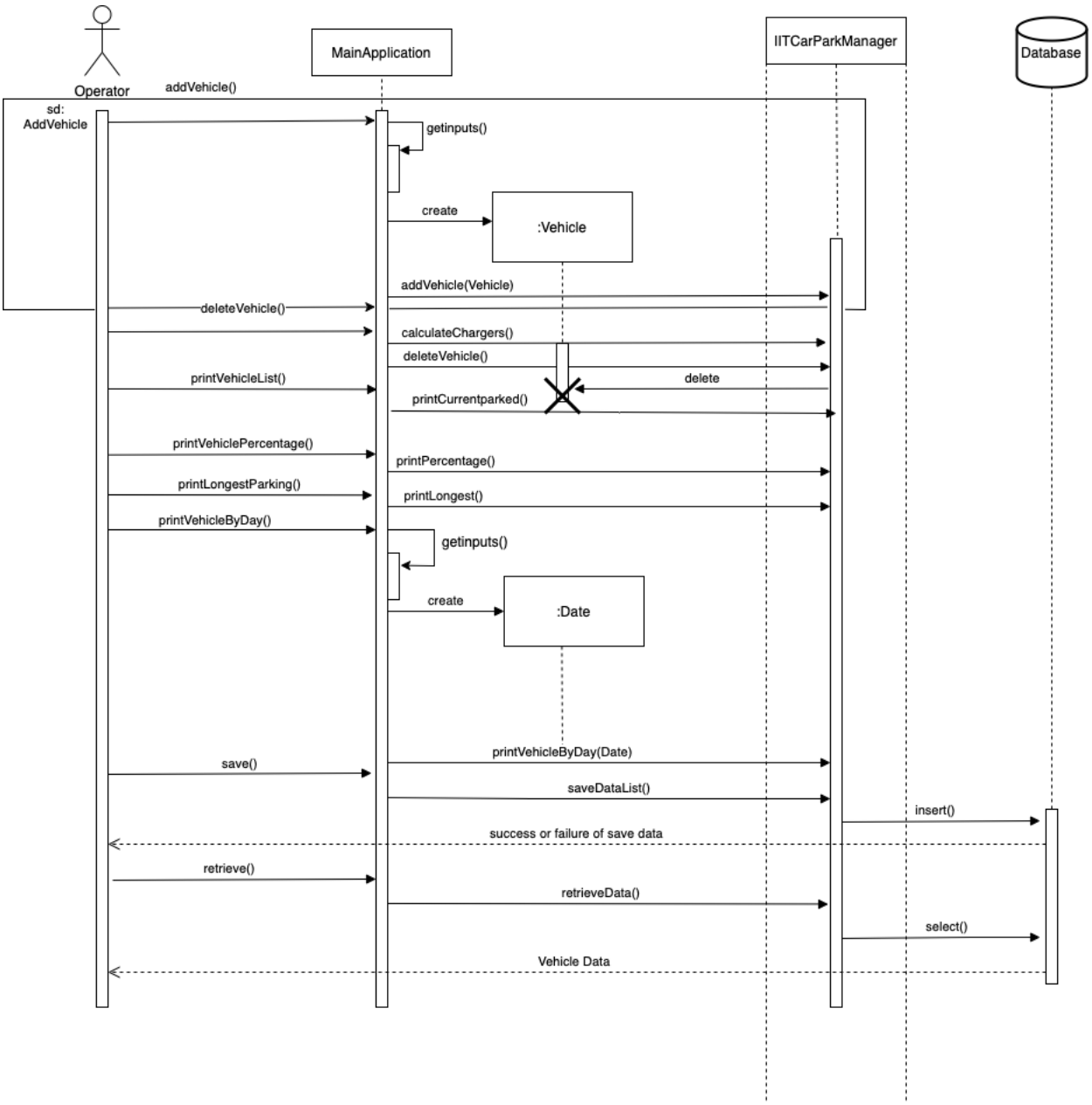
Use Case Diagram



Analysis Class Diagram

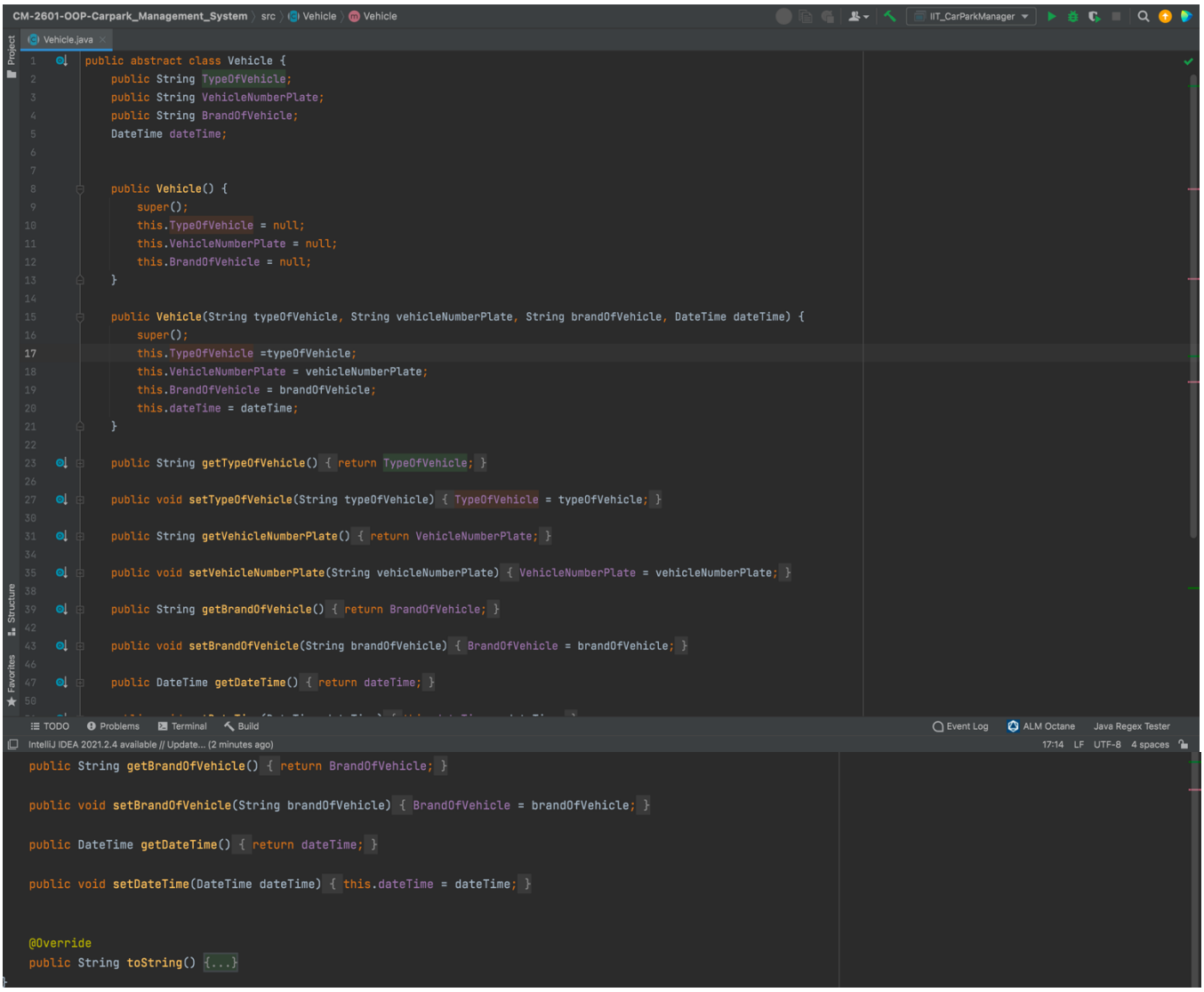


Sequence Diagram

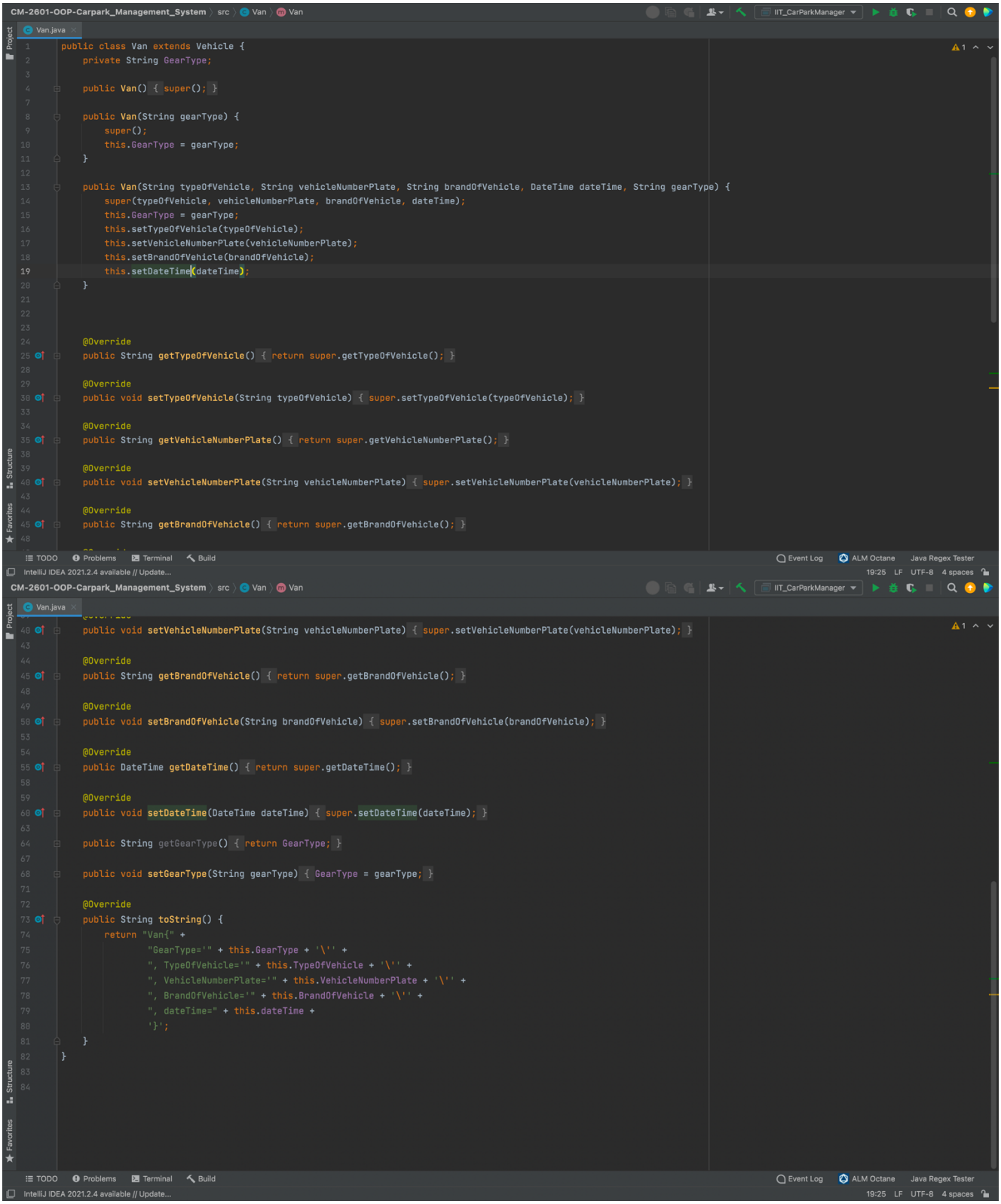


Code Snippets

1. Vehicle.java



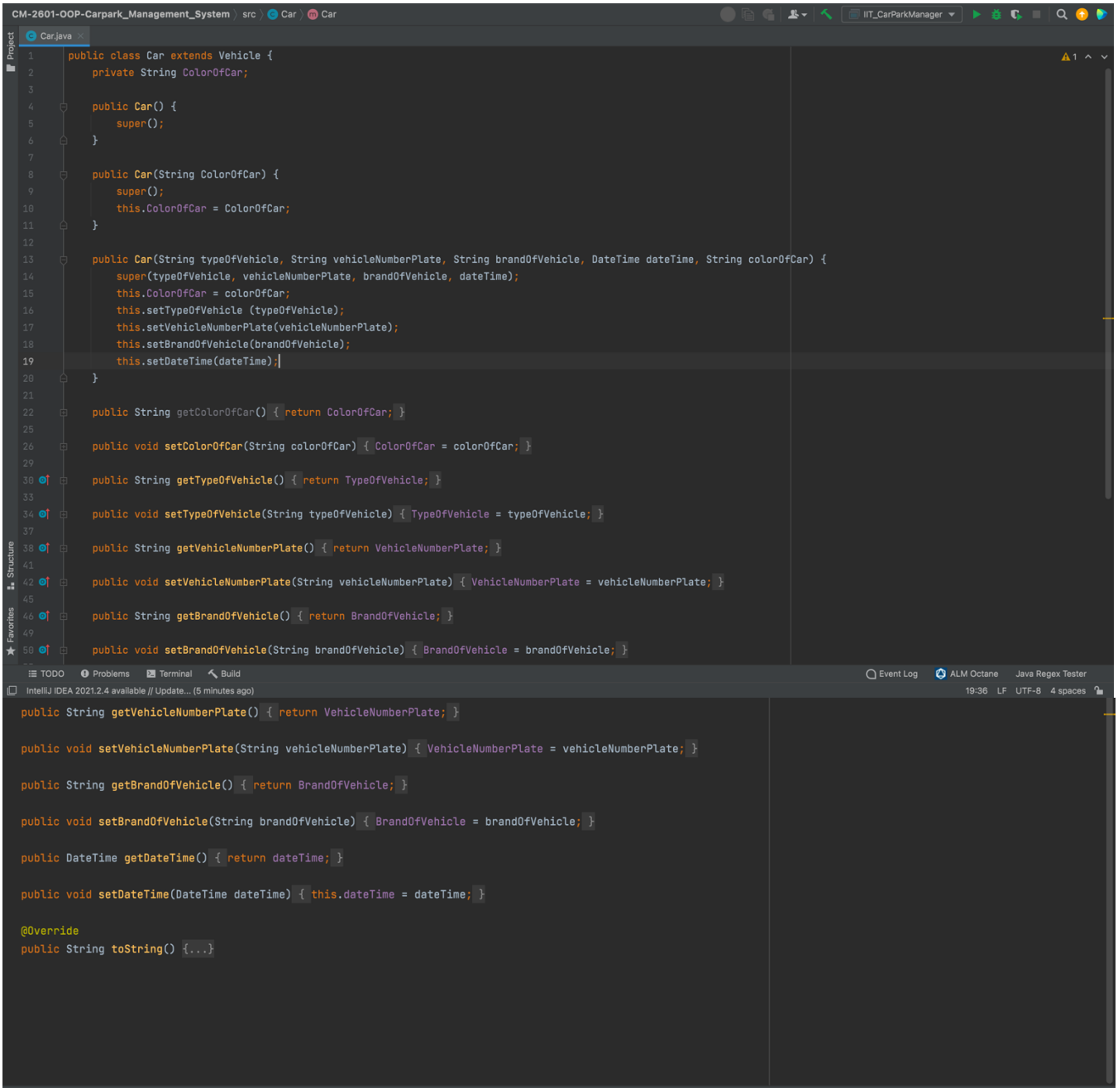
2. Van.java



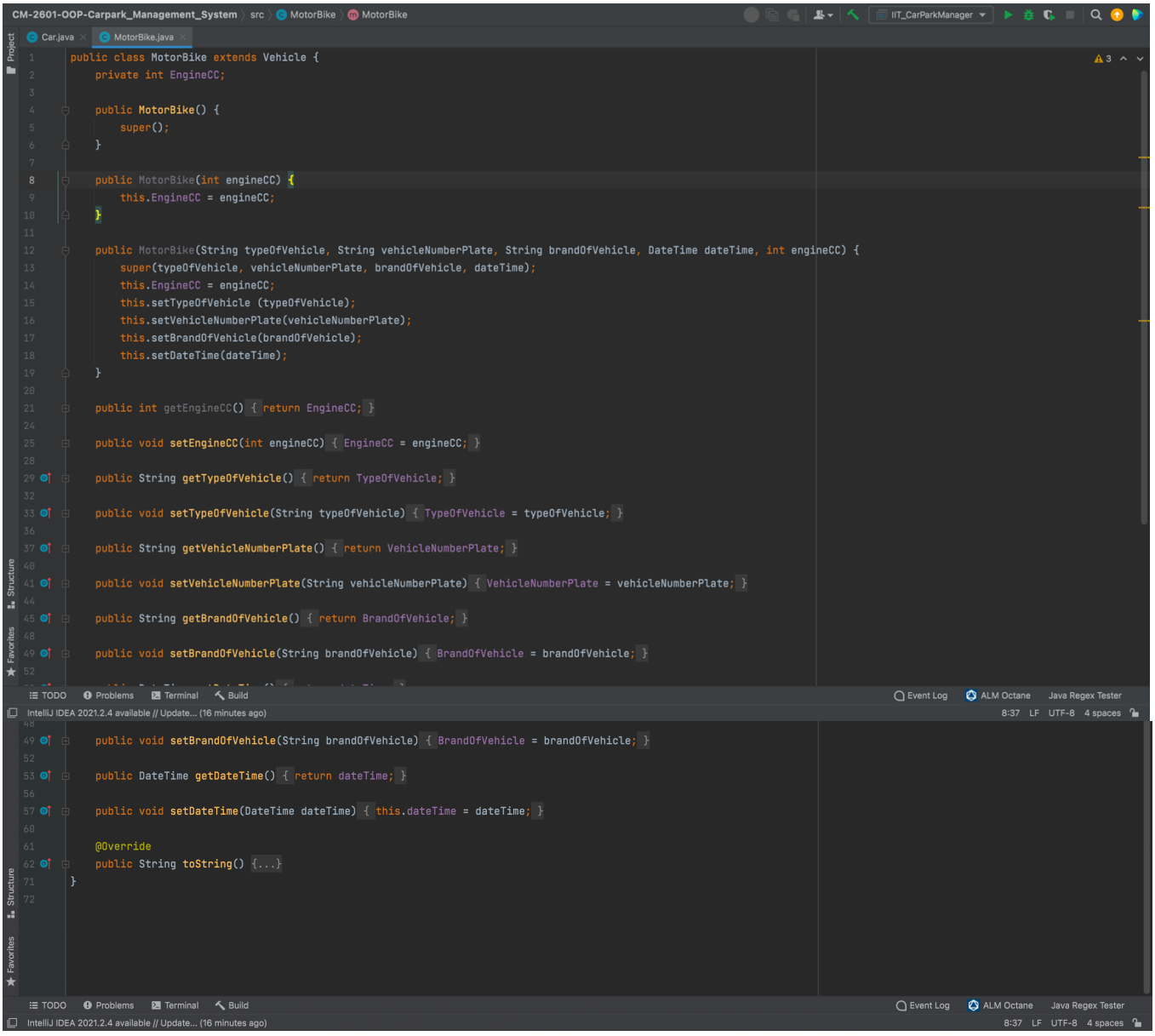
```
public class Van extends Vehicle {
    private String GearType;

    public Van() { super(); }

    public Van(String gearType) {
        super();
        this.GearType = gearType;
    }

    public Van(String typeOfVehicle, String vehicleNumberPlate, String brandOfVehicle, DateTime dateTime, String gearType) {
        super(typeOfVehicle, vehicleNumberPlate, brandOfVehicle, dateTime);
        this.GearType = gearType;
        this.setTypeOfVehicle(typeOfVehicle);
        this.setVehicleNumberPlate(vehicleNumberPlate);
        this.setBrandOfVehicle(brandOfVehicle);
        this.setDateTime(dateTime);
    }



    @Override
    public String getTypeOfVehicle() { return super.getTypeOfVehicle(); }

    @Override
    public void setTypeOfVehicle(String typeOfVehicle) { super.setTypeOfVehicle(typeOfVehicle); }

    @Override
    public String getVehicleNumberPlate() { return super.getVehicleNumberPlate(); }

    @Override
    public void setVehicleNumberPlate(String vehicleNumberPlate) { super.setVehicleNumberPlate(vehicleNumberPlate); }

    @Override
    public String getBrandOfVehicle() { return super.getBrandOfVehicle(); }
```

```
    public void setVehicleNumberPlate(String vehicleNumberPlate) { super.setVehicleNumberPlate(vehicleNumberPlate); }

    @Override
    public String getBrandOfVehicle() { return super.getBrandOfVehicle(); }

    @Override
    public void setBrandOfVehicle(String brandOfVehicle) { super.setBrandOfVehicle(brandOfVehicle); }

    @Override
    public DateTime getDateTime() { return super.getDateTime(); }

    @Override
    public void setDateTime(DateTime dateTime) { super.setDateTime(dateTime); }

    public String getGearType() { return GearType; }

    public void setGearType(String gearType) { GearType = gearType; }

    @Override
    public String toString() {
        return "Van{" +
                "GearType='" + this.GearType + '\'' +
                ", TypeOfVehicle='" + this.TypeOfVehicle + '\'' +
                ", VehicleNumberPlate='" + this.VehicleNumberPlate + '\'' +
                ", BrandOfVehicle='" + this.BrandOfVehicle + '\'' +
                ", dateTime=" + this.dateTime +
                '}';
    }
}
```

3. Car.java

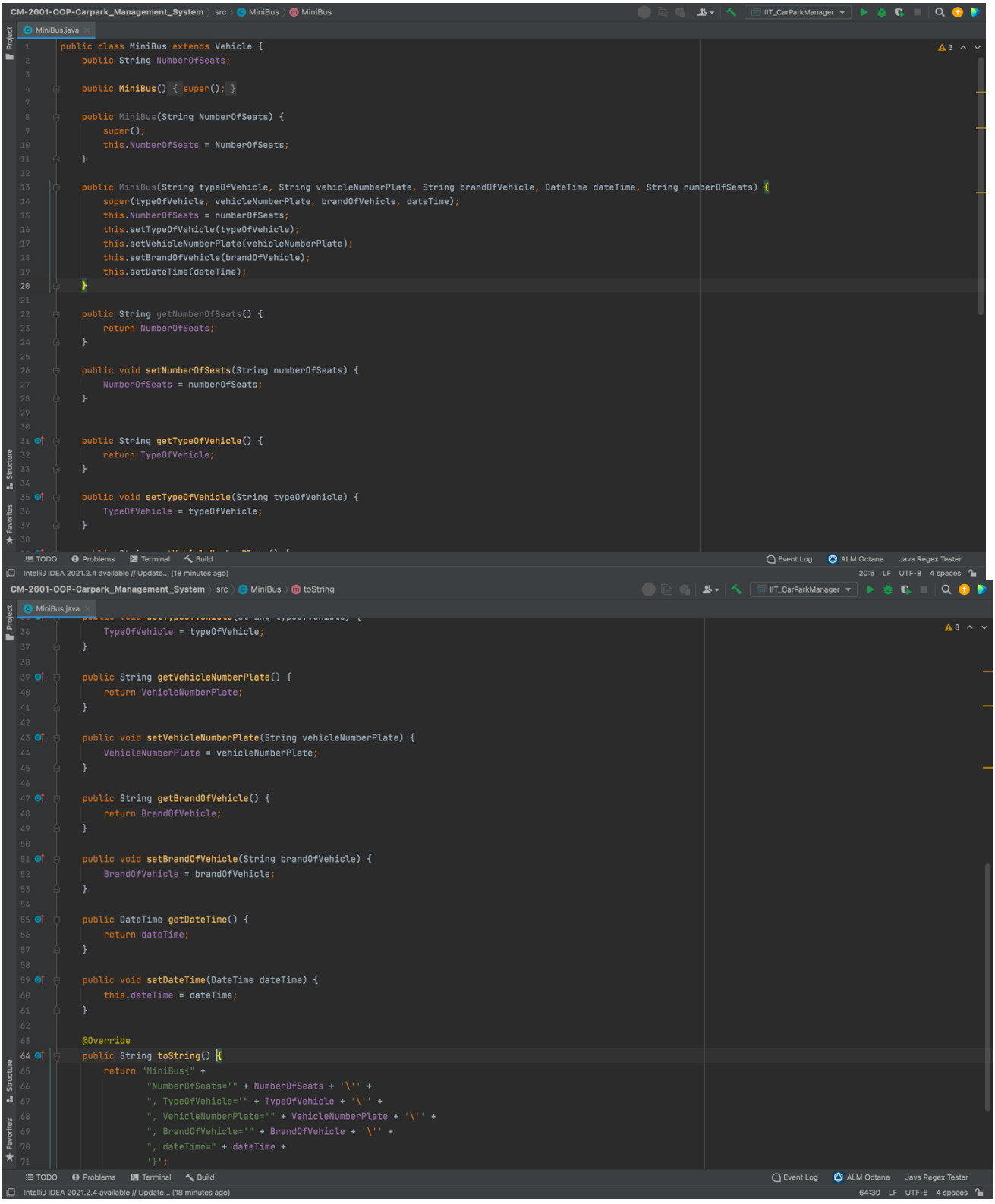

CM-2601-OOP-Carpark_Management_System
src
Car
Car
IIT_CarParkManager
Car.java
public class Car extends Vehicle {
private String ColorOfCar;
public Car() {
super();
}
public Car(String ColorOfCar) {
super();
this.ColorOfCar = ColorOfCar;
}
public Car(String typeOfVehicle, String vehicleNumberPlate, String brandOfVehicle, DateTime dateTime, String colorOfCar) {
super(typeOfVehicle, vehicleNumberPlate, brandOfVehicle, dateTime);
this.ColorOfCar = colorOfCar;
this.setTypeOfVehicle (typeOfVehicle);
this.setVehicleNumberPlate(vehicleNumberPlate);
this.setBrandOfVehicle(brandOfVehicle);
this.setDateTime(dateTime);
}
public String getColorOfCar() { return ColorOfCar; }
public void setColorOfCar(String colorOfCar) { ColorOfCar = colorOfCar; }
public String getTypeOfVehicle() { return TypeOfVehicle; }
public void setTypeOfVehicle(String typeOfVehicle) { TypeOfVehicle = typeOfVehicle; }
public String getVehicleNumberPlate() { return VehicleNumberPlate; }
public void setVehicleNumberPlate(String vehicleNumberPlate) { VehicleNumberPlate = vehicleNumberPlate; }
public String getBrandOfVehicle() { return BrandOfVehicle; }
public void setBrandOfVehicle(String brandOfVehicle) { BrandOfVehicle = brandOfVehicle; }
TODO
Problems
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (5 minutes ago)
19:36 LF UTF-8 4 spaces
public String getVehicleNumberPlate() { return VehicleNumberPlate; }
public void setVehicleNumberPlate(String vehicleNumberPlate) { VehicleNumberPlate = vehicleNumberPlate; }
public String getBrandOfVehicle() { return BrandOfVehicle; }
public void setBrandOfVehicle(String brandOfVehicle) { BrandOfVehicle = brandOfVehicle; }
public DateTime getDateTime() { return dateTime; }
public void setDateTime(DateTime dateTime) { this.dateTime = dateTime; }
@Override
public String toString() {...}

4. motorbike.java



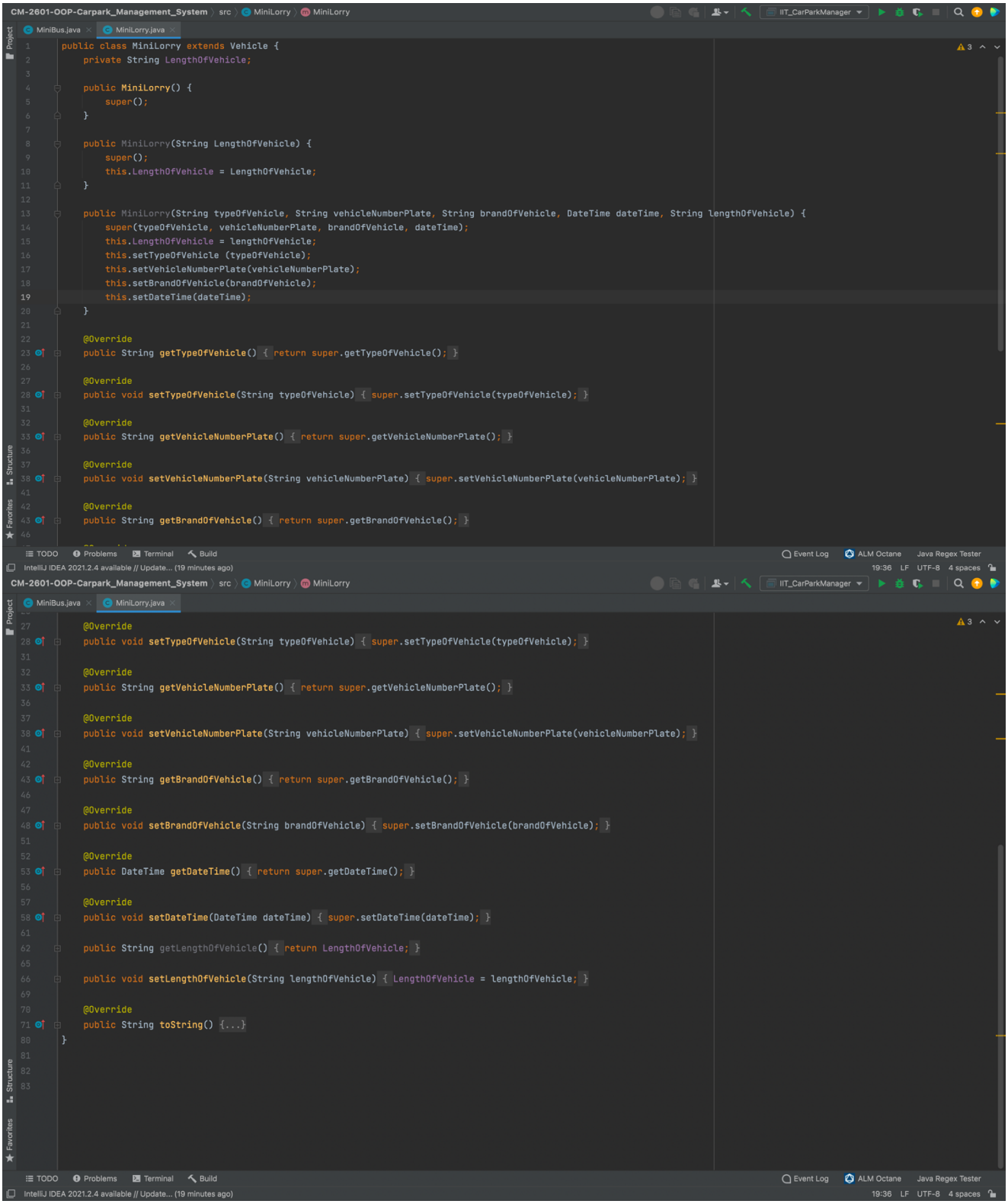
```
public class MotorBike extends Vehicle {
    private int EngineCC;

    public MotorBike() {
        super();
    }

    public MotorBike(int engineCC) {
        this.EngineCC = engineCC;
    }

    public MotorBike(String typeOfVehicle, String vehicleNumberPlate, String brandOfVehicle, DateTime dateTime, int engineCC) {
        super(typeOfVehicle, vehicleNumberPlate, brandOfVehicle, dateTime);
        this.EngineCC = engineCC;
        this.setTypeOfVehicle (typeOfVehicle);
        this.setVehicleNumberPlate(vehicleNumberPlate);
        this.setBrandOfVehicle(brandOfVehicle);
        this.setDateTime(dateTime);
    }

    public int getEngineCC() { return EngineCC; }

    public void setEngineCC(int engineCC) { EngineCC = engineCC; }

    public String getTypeOfVehicle() { return TypeOfVehicle; }

    public void setTypeOfVehicle(String typeOfVehicle) { TypeOfVehicle = typeOfVehicle; }

    public String getVehicleNumberPlate() { return VehicleNumberPlate; }

    public void setVehicleNumberPlate(String vehicleNumberPlate) { VehicleNumberPlate = vehicleNumberPlate; }

    public String getBrandOfVehicle() { return BrandOfVehicle; }

    public void setBrandOfVehicle(String brandOfVehicle) { BrandOfVehicle = brandOfVehicle; }

    public void setBrandOfVehicle(String brandOfVehicle) { BrandOfVehicle = brandOfVehicle; }

    public DateTime getDateTime() { return dateTime; }

    public void setDateTime(DateTime dateTime) { this.dateTime = dateTime; }

    @Override
    public String toString() {...}
}
```

5. minibus.java



```java
public class MiniBus extends Vehicle {
    public String NumberOfSeats;

    public MiniBus() { super(); }

    public MiniBus(String NumberOfSeats) {
        super();
        this.NumberOfSeats = NumberOfSeats;
    }

    public MiniBus(String typeOfVehicle, String vehicleNumberPlate, String brandOfVehicle, DateTime dateTime, String numberOfSeats) {
        super(typeOfVehicle, vehicleNumberPlate, brandOfVehicle, dateTime);
        this.NumberOfSeats = numberOfSeats;
        this.setTypeOfVehicle(typeOfVehicle);
        this.setVehicleNumberPlate(vehicleNumberPlate);
        this.setBrandOfVehicle(brandOfVehicle);
        this.setDateTime(dateTime);
    }

    public String getNumberOfSeats() {
        return NumberOfSeats;
    }

    public void setNumberOfSeats(String numberOfSeats) {
        NumberOfSeats = numberOfSeats;
    }


    public String getTypeOfVehicle() {
        return TypeOfVehicle;
    }

    public void setTypeOfVehicle(String typeOfVehicle) {
        TypeOfVehicle = typeOfVehicle;
    }

    public String getVehicleNumberPlate() {
        return VehicleNumberPlate;
    }

    public void setVehicleNumberPlate(String vehicleNumberPlate) {
        VehicleNumberPlate = vehicleNumberPlate;
    }

    public String getBrandOfVehicle() {
        return BrandOfVehicle;
    }

    public void setBrandOfVehicle(String brandOfVehicle) {
        BrandOfVehicle = brandOfVehicle;
    }

    public DateTime getDateTime() {
        return dateTime;
    }

    public void setDateTime(DateTime dateTime) {
        this.dateTime = dateTime;
    }

    @Override
    public String toString() {
        return "MiniBus{" +
                "NumberOfSeats='" + NumberOfSeats + '\'' +
                ", TypeOfVehicle='" + TypeOfVehicle + '\'' +
                ", VehicleNumberPlate='" + VehicleNumberPlate + '\'' +
                ", BrandOfVehicle='" + BrandOfVehicle + '\'' +
                ", dateTime=" + dateTime +
                '}';
    }
```

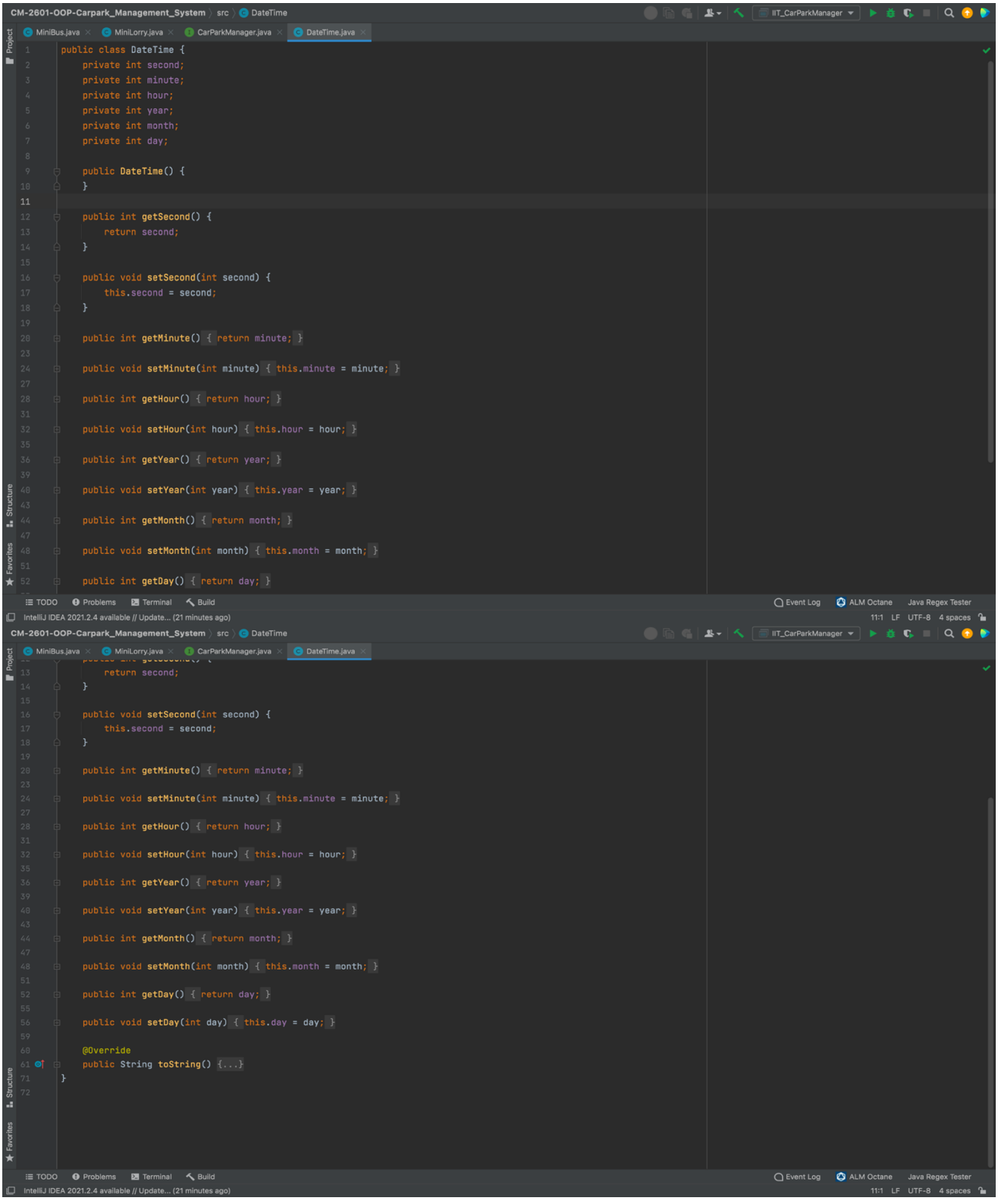
6. miniLorry.java



```
public class MiniLorry extends Vehicle {
    private String LengthOfVehicle;

    public MiniLorry() {
        super();
    }

    public MiniLorry(String LengthOfVehicle) {
        super();
        this.LengthOfVehicle = LengthOfVehicle;
    }

    public MiniLorry(String typeOfVehicle, String vehicleNumberPlate, String brandOfVehicle, DateTime dateTime, String lengthOfVehicle) {
        super(typeOfVehicle, vehicleNumberPlate, brandOfVehicle, dateTime);
        this.LengthOfVehicle = lengthOfVehicle;
        this.setTypeOfVehicle (typeOfVehicle);
        this.setVehicleNumberPlate(vehicleNumberPlate);
        this.setBrandOfVehicle(brandOfVehicle);
        this.setDateTime(dateTime);
    }

    @Override
    public String getTypeOfVehicle() { return super.getTypeOfVehicle(); }

    @Override
    public void setTypeOfVehicle(String typeOfVehicle) { super.setTypeOfVehicle(typeOfVehicle); }

    @Override
    public String getVehicleNumberPlate() { return super.getVehicleNumberPlate(); }

    @Override
    public void setVehicleNumberPlate(String vehicleNumberPlate) { super.setVehicleNumberPlate(vehicleNumberPlate); }

    @Override
    public String getBrandOfVehicle() { return super.getBrandOfVehicle(); }
```

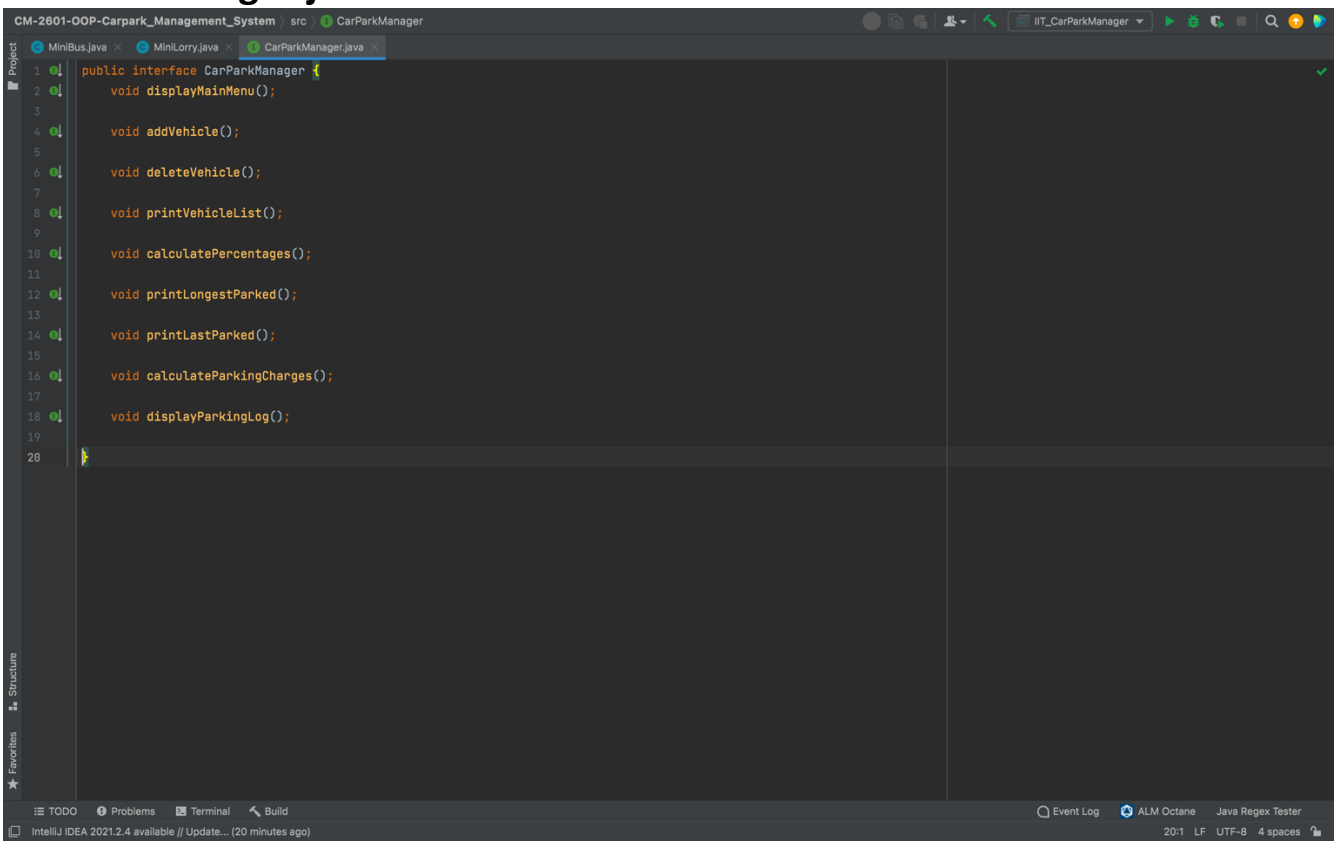
```
    @Override
    public void setTypeOfVehicle(String typeOfVehicle) { super.setTypeOfVehicle(typeOfVehicle); }

    @Override
    public String getVehicleNumberPlate() { return super.getVehicleNumberPlate(); }

    @Override
    public void setVehicleNumberPlate(String vehicleNumberPlate) { super.setVehicleNumberPlate(vehicleNumberPlate); }

    @Override
    public String getBrandOfVehicle() { return super.getBrandOfVehicle(); }

    @Override
    public void setBrandOfVehicle(String brandOfVehicle) { super.setBrandOfVehicle(brandOfVehicle); }

    @Override
    public DateTime getDateTime() { return super.getDateTime(); }

    @Override
    public void setDateTime(DateTime dateTime) { super.setDateTime(dateTime); }

    public String getLengthOfVehicle() { return LengthOfVehicle; }

    public void setLengthOfVehicle(String lengthOfVehicle) { LengthOfVehicle = lengthOfVehicle; }

    @Override
    public String toString() {...}
}
```

7. DataTime.java



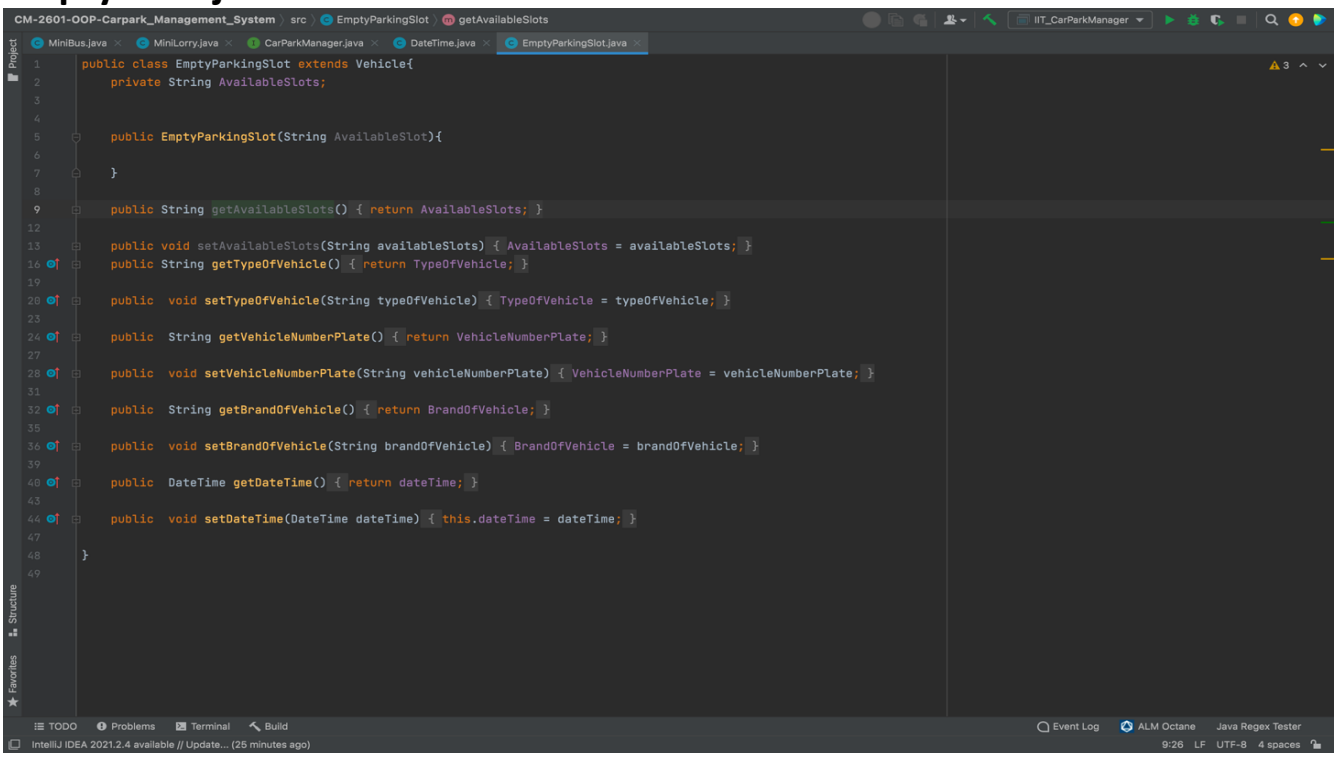
```
public class DateTime {
    private int second;
    private int minute;
    private int hour;
    private int year;
    private int month;
    private int day;

    public DateTime() {
    }

    public int getSecond() {
        return second;
    }

    public void setSecond(int second) {
        this.second = second;
    }

    public int getMinute() { return minute; }

    public void setMinute(int minute) { this.minute = minute; }

    public int getHour() { return hour; }

    public void setHour(int hour) { this.hour = hour; }

    public int getYear() { return year; }

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

    public int getMonth() { return month; }

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

    public int getDay() { return day; }

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

    @Override
    public String toString() {...}
}
```

8. carParkManager.java



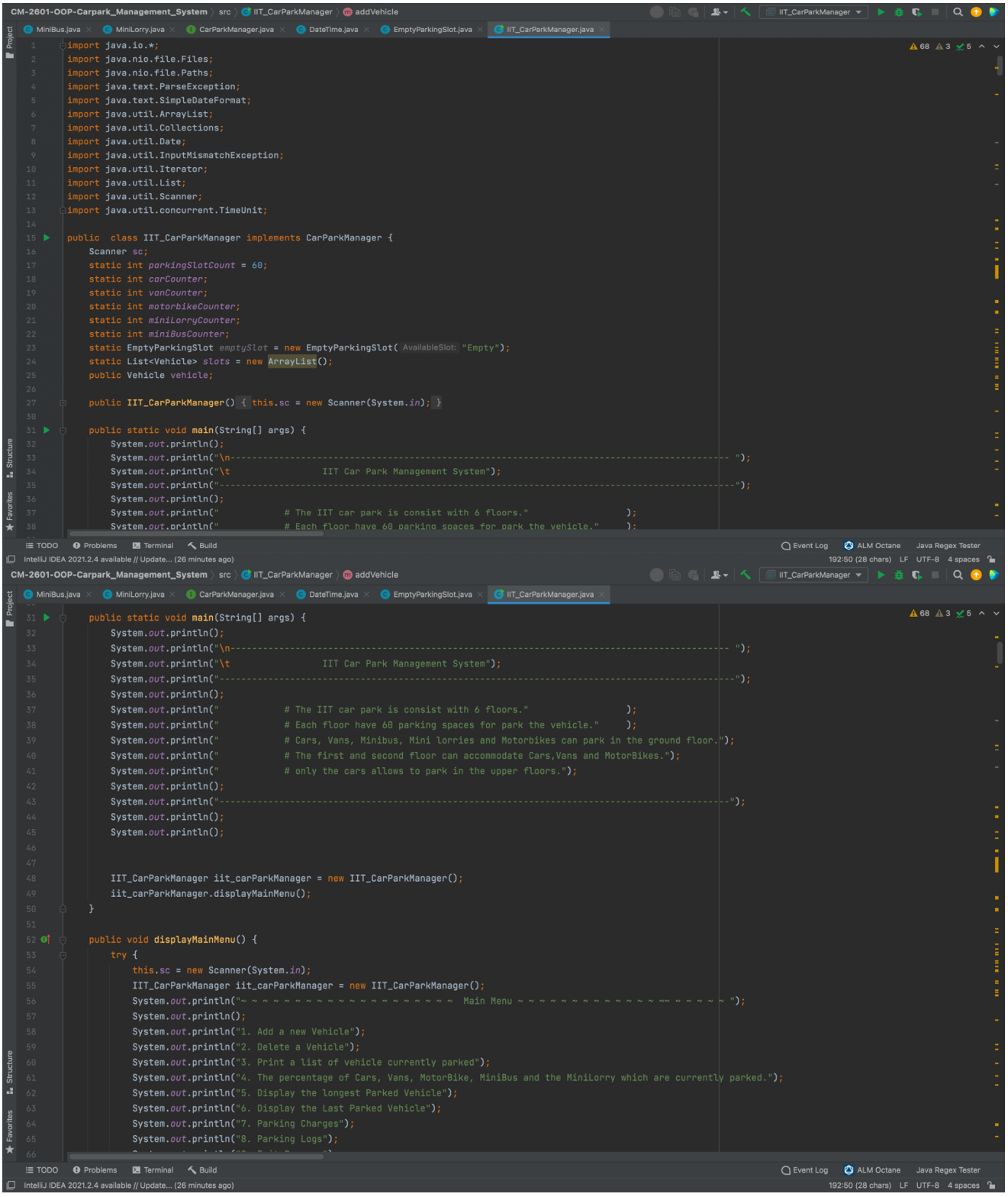
```
public interface CarParkManager {
    void displayMainMenu();

    void addVehicle();

    void deleteVehicle();

    void printVehicleList();

    void calculatePercentages();

    void printLongestParked();

    void printLastParked();

    void calculateParkingCharges();

    void displayParkingLog();

}
```

9. EmptySlots.java

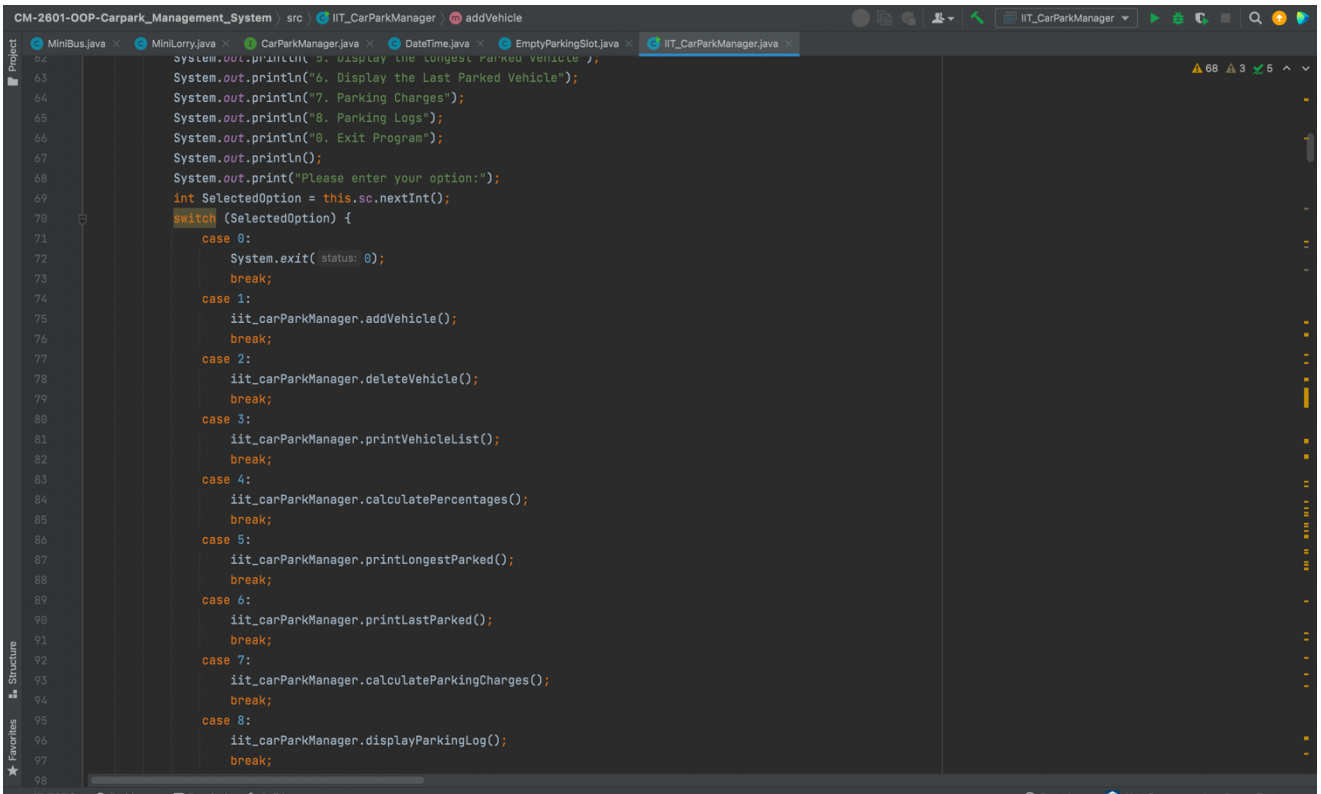


```
public class EmptyParkingSlot extends Vehicle{
    private String AvailableSlots;


    public EmptyParkingSlot(String AvailableSlot){

    }

    public String getAvailableSlots() { return AvailableSlots; }

    public void setAvailableSlots(String availableSlots) { AvailableSlots = availableSlots; }
    public String getTypeOfVehicle() { return TypeOfVehicle; }

    public  void setTypeOfVehicle(String typeOfVehicle) { TypeOfVehicle = typeOfVehicle; }

    public  String getVehicleNumberPlate() { return VehicleNumberPlate; }

    public  void setVehicleNumberPlate(String vehicleNumberPlate) { VehicleNumberPlate = vehicleNumberPlate; }

    public  String getBrandOfVehicle() { return BrandOfVehicle; }

    public  void setBrandOfVehicle(String brandOfVehicle) { BrandOfVehicle = brandOfVehicle; }

    public  DateTime getDateTime() { return dateTime; }

    public  void setDateTime(DateTime dateTime) { this.dateTime = dateTime; }

}
```

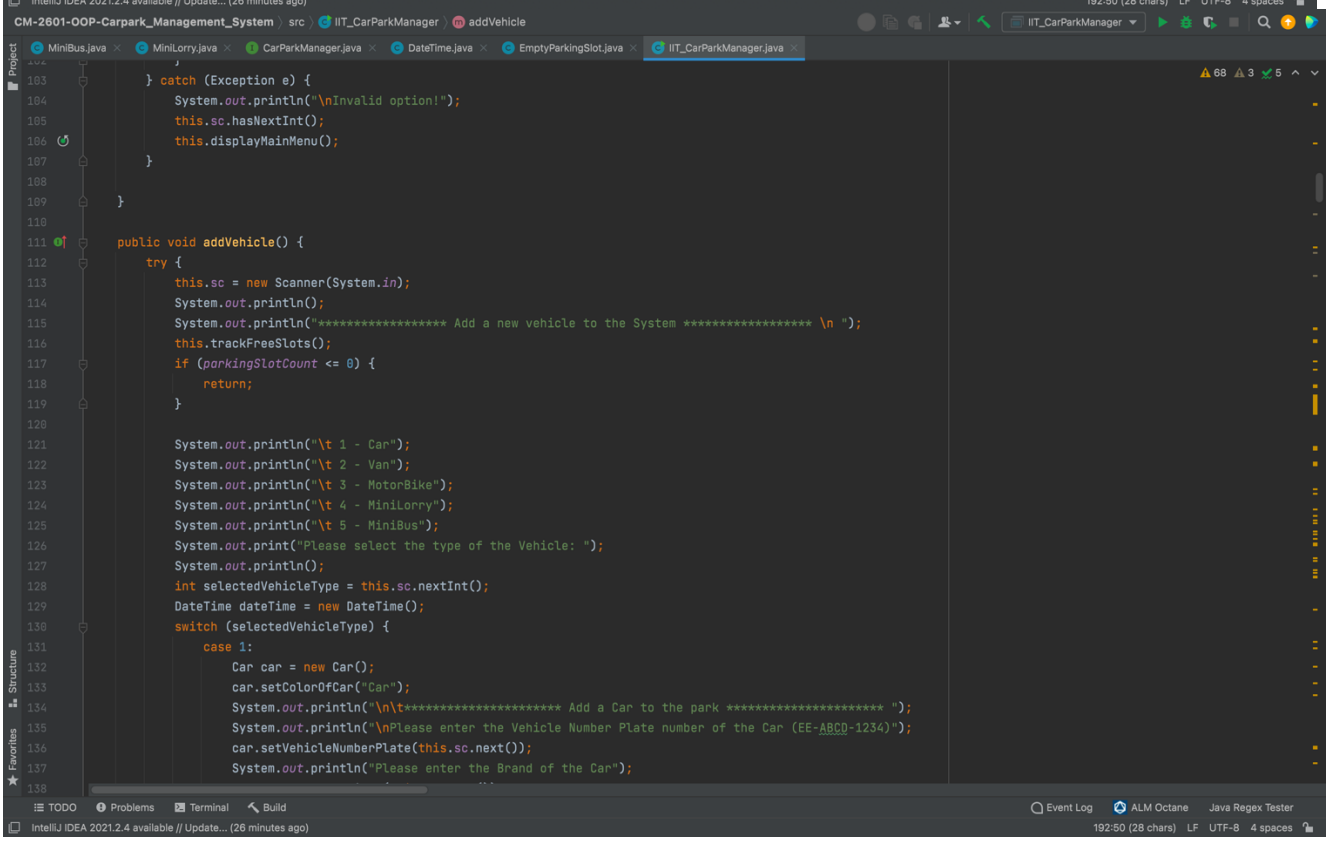
10. IITcarParkManager.java



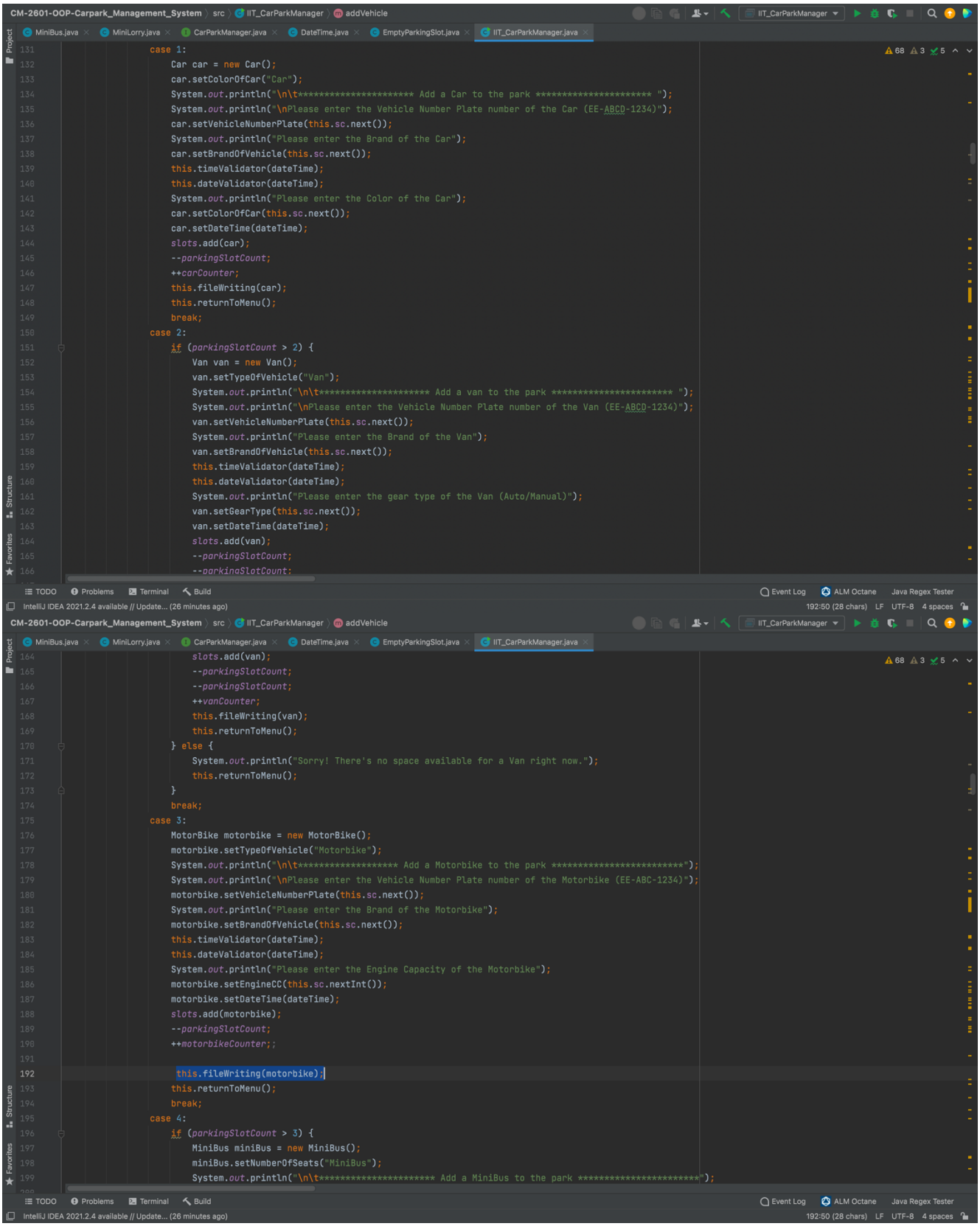
```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

public  class IIT_CarParkManager implements CarParkManager {
    Scanner sc;
    static int parkingSlotCount = 60;
    static int carCounter;
    static int vanCounter;
    static int motorbikeCounter;
    static int miniLorryCounter;
    static int miniBusCounter;
    static EmptyParkingSlot emptySlot = new EmptyParkingSlot( AvailableSlot: "Empty");
    static List<Vehicle> slots = new ArrayList();
    public Vehicle vehicle;

    public IIT_CarParkManager() { this.sc = new Scanner(System.in); }

    public static void main(String[] args) {
        System.out.println();
        System.out.println("\n---------------------------------------------------------------------------------------------- ");
        System.out.println("\t                IIT Car Park Management System");
        System.out.println("----------------------------------------------------------------------------------------------");
        System.out.println();
        System.out.println("           # The IIT car park is consist with 6 floors."                );
        System.out.println("           # Each floor have 60 parking spaces for park the vehicle."    );
        System.out.println("           # Cars, Vans, Minibus, Mini lorries and Motorbikes can park in the ground floor.");
        System.out.println("           # The first and second floor can accommodate Cars,Vans and MotorBikes.");
        System.out.println("           # only the cars allows to park in the upper floors.");
        System.out.println();
        System.out.println("----------------------------------------------------------------------------------------------");
        System.out.println();
        System.out.println();



        IIT_CarParkManager iit_carParkManager = new IIT_CarParkManager();
        iit_carParkManager.displayMainMenu();
    }

    public void displayMainMenu() {
        try {
            this.sc = new Scanner(System.in);
            IIT_CarParkManager iit_carParkManager = new IIT_CarParkManager();
            System.out.println("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~  Main Menu ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ");
            System.out.println();
            System.out.println("1. Add a new Vehicle");
            System.out.println("2. Delete a Vehicle");
            System.out.println("3. Print a list of vehicle currently parked");
            System.out.println("4. The percentage of Cars, Vans, MotorBike, MiniBus and the MiniLorry which are currently parked.");
            System.out.println("5. Display the longest Parked Vehicle");
            System.out.println("6. Display the Last Parked Vehicle");
            System.out.println("7. Parking Charges");
            System.out.println("8. Parking Logs");
```



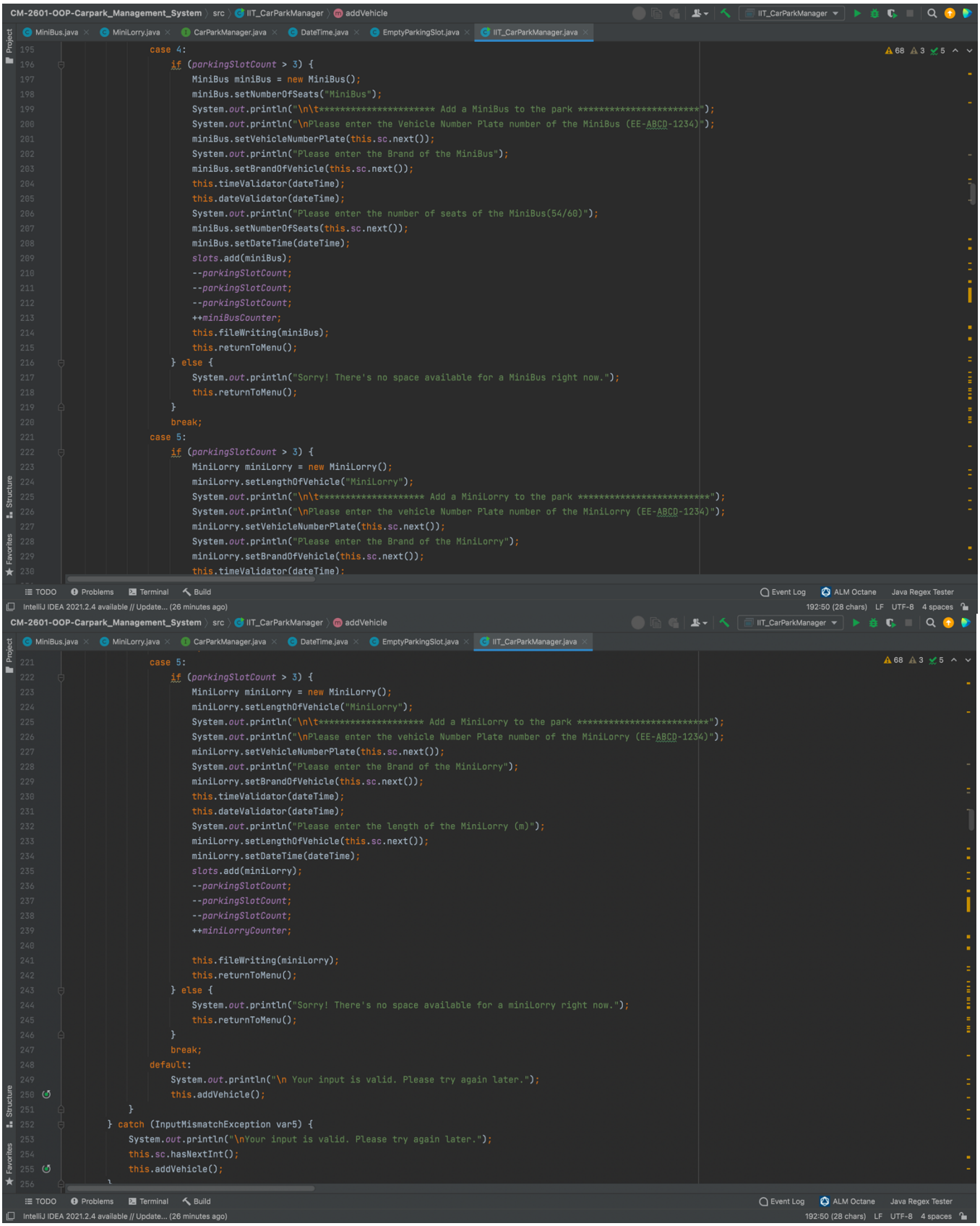
```
System.out.println("6. Display the Last Parked Vehicle");
System.out.println("7. Parking Charges");
System.out.println("8. Parking Logs");
System.out.println("0. Exit Program");
System.out.println();
System.out.print("Please enter your option:");
int SelectedOption = this.sc.nextInt();
switch (SelectedOption) {
    case 0:
        System.exit( status: 0);
        break;
    case 1:
        iit_carParkManager.addVehicle();
        break;
    case 2:
        iit_carParkManager.deleteVehicle();
        break;
    case 3:
        iit_carParkManager.printVehicleList();
        break;
    case 4:
        iit_carParkManager.calculatePercentages();
        break;
    case 5:
        iit_carParkManager.printLongestParked();
        break;
    case 6:
        iit_carParkManager.printLastParked();
        break;
    case 7:
        iit_carParkManager.calculateParkingCharges();
        break;
    case 8:
        iit_carParkManager.displayParkingLog();
        break;
```



```
    } catch (Exception e) {
        System.out.println("\nInvalid option!");
        this.sc.hasNextInt();
        this.displayMainMenu();
    }

}

public void addVehicle() {
    try {
        this.sc = new Scanner(System.in);
        System.out.println();
        System.out.println("****************** Add a new vehicle to the System ****************** \n ");
        this.trackFreeSlots();
        if (parkingSlotCount <= 0) {
            return;
        }

        System.out.println("\t 1 - Car");
        System.out.println("\t 2 - Van");
        System.out.println("\t 3 - MotorBike");
        System.out.println("\t 4 - MiniLorry");
        System.out.println("\t 5 - MiniBus");
        System.out.print("Please select the type of the Vehicle: ");
        System.out.println();
        int selectedVehicleType = this.sc.nextInt();
        DateTime dateTime = new DateTime();
        switch (selectedVehicleType) {
            case 1:
                Car car = new Car();
                car.setColorOfCar("Car");
                System.out.println("\n\t********************** Add a Car to the park ********************** ");
                System.out.println("\nPlease enter the Vehicle Number Plate number of the Car (EE-ABCD-1234)");
                car.setVehicleNumberPlate(this.sc.next());
                System.out.println("Please enter the Brand of the Car");
```

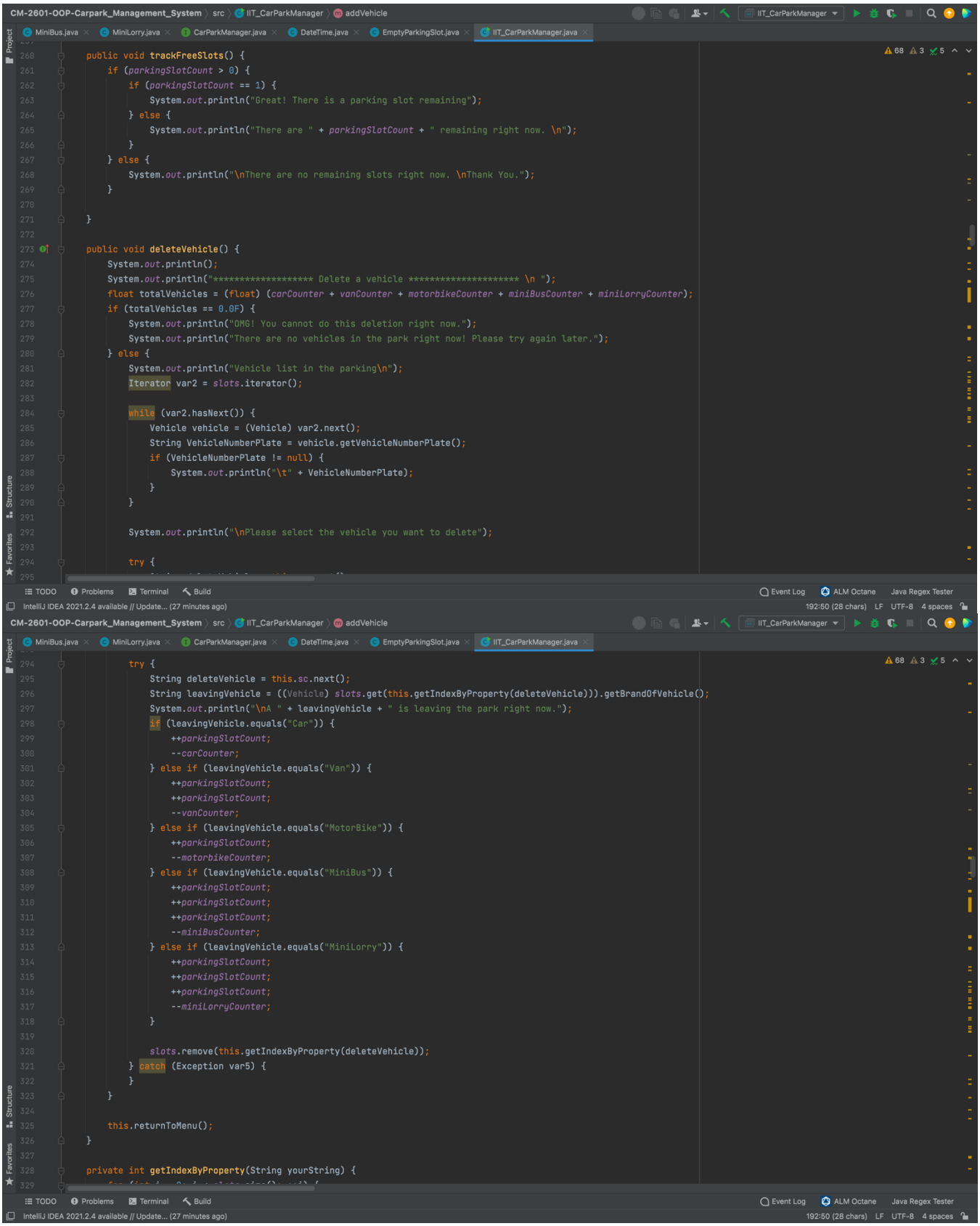


```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
131                 case 1:
132                     Car car = new Car();
133                     car.setColorOfCar("Car");
134                     System.out.println("\n\t********************* Add a Car to the park ********************* ");
135                     System.out.println("\nPlease enter the Vehicle Number Plate number of the Car (EE-ABCD-1234)");
136                     car.setVehicleNumberPlate(this.sc.next());
137                     System.out.println("Please enter the Brand of the Car");
138                     car.setBrandOfVehicle(this.sc.next());
139                     this.timeValidator(dateTime);
140                     this.dateValidator(dateTime);
141                     System.out.println("Please enter the Color of the Car");
142                     car.setColorOfCar(this.sc.next());
143                     car.setDateTime(dateTime);
144                     slots.add(car);
145                     --parkingSlotCount;
146                     ++carCounter;
147                     this.fileWriting(car);
148                     this.returnToMenu();
149                     break;
150                 case 2:
151                     if (parkingSlotCount > 2) {
152                         Van van = new Van();
153                         van.setTypeOfVehicle("Van");
154                         System.out.println("\n\t********************* Add a van to the park ********************* ");
155                         System.out.println("\nPlease enter the Vehicle Number Plate number of the Van (EE-ABCD-1234)");
156                         van.setVehicleNumberPlate(this.sc.next());
157                         System.out.println("Please enter the Brand of the Van");
158                         van.setBrandOfVehicle(this.sc.next());
159                         this.timeValidator(dateTime);
160                         this.dateValidator(dateTime);
161                         System.out.println("Please enter the gear type of the Van (Auto/Manual)");
162                         van.setGearType(this.sc.next());
163                         van.setDateTime(dateTime);
164                         slots.add(van);
165                         --parkingSlotCount;
166                         --parkingSlotCount;
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (26 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
164                     slots.add(van);
165                     --parkingSlotCount;
166                     --parkingSlotCount;
167                     ++vanCounter;
168                     this.fileWriting(van);
169                     this.returnToMenu();
170                 } else {
171                     System.out.println("Sorry! There's no space available for a Van right now.");
172                     this.returnToMenu();
173                 }
174                 break;
175             case 3:
176                 MotorBike motorbike = new MotorBike();
177                 motorbike.setTypeOfVehicle("Motorbike");
178                 System.out.println("\n\t********************* Add a Motorbike to the park ************************");
179                 System.out.println("\nPlease enter the Vehicle Number Plate number of the Motorbike (EE-ABC-1234)");
180                 motorbike.setVehicleNumberPlate(this.sc.next());
181                 System.out.println("Please enter the Brand of the Motorbike");
182                 motorbike.setBrandOfVehicle(this.sc.next());
183                 this.timeValidator(dateTime);
184                 this.dateValidator(dateTime);
185                 System.out.println("Please enter the Engine Capacity of the Motorbike");
186                 motorbike.setEngineCC(this.sc.nextInt());
187                 motorbike.setDateTime(dateTime);
188                 slots.add(motorbike);
189                 --parkingSlotCount;
190                 ++motorbikeCounter;;
191
192                 this.fileWriting(motorbike);
193                 this.returnToMenu();
194                 break;
195             case 4:
196                 if (parkingSlotCount > 3) {
197                     MiniBus miniBus = new MiniBus();
198                     miniBus.setNumberOfSeats("MiniBus");
199                     System.out.println("\n\t********************* Add a MiniBus to the park ************************");
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (26 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```



```
                case 4:
                    if (parkingSlotCount > 3) {
                        MiniBus miniBus = new MiniBus();
                        miniBus.setNumberOfSeats("MiniBus");
                        System.out.println("\n\t********************** Add a MiniBus to the park **********************");
                        System.out.println("\nPlease enter the Vehicle Number Plate number of the MiniBus (EE-ABCD-1234)");
                        miniBus.setVehicleNumberPlate(this.sc.next());
                        System.out.println("Please enter the Brand of the MiniBus");
                        miniBus.setBrandOfVehicle(this.sc.next());
                        this.timeValidator(dateTime);
                        this.dateValidator(dateTime);
                        System.out.println("Please enter the number of seats of the MiniBus(54/60)");
                        miniBus.setNumberOfSeats(this.sc.next());
                        miniBus.setDateTime(dateTime);
                        slots.add(miniBus);
                        --parkingSlotCount;
                        --parkingSlotCount;
                        --parkingSlotCount;
                        ++miniBusCounter;
                        this.fileWriting(miniBus);
                        this.returnToMenu();
                    } else {
                        System.out.println("Sorry! There's no space available for a MiniBus right now.");
                        this.returnToMenu();
                    }
                    break;
                case 5:
                    if (parkingSlotCount > 3) {
                        MiniLorry miniLorry = new MiniLorry();
                        miniLorry.setLengthOfVehicle("MiniLorry");
                        System.out.println("\n\t******************** Add a MiniLorry to the park ************************");
                        System.out.println("\nPlease enter the vehicle Number Plate number of the MiniLorry (EE-ABCD-1234)");
                        miniLorry.setVehicleNumberPlate(this.sc.next());
                        System.out.println("Please enter the Brand of the MiniLorry");
                        miniLorry.setBrandOfVehicle(this.sc.next());
                        this.timeValidator(dateTime);
```

```
                case 5:
                    if (parkingSlotCount > 3) {
                        MiniLorry miniLorry = new MiniLorry();
                        miniLorry.setLengthOfVehicle("MiniLorry");
                        System.out.println("\n\t******************** Add a MiniLorry to the park ************************");
                        System.out.println("\nPlease enter the vehicle Number Plate number of the MiniLorry (EE-ABCD-1234)");
                        miniLorry.setVehicleNumberPlate(this.sc.next());
                        System.out.println("Please enter the Brand of the MiniLorry");
                        miniLorry.setBrandOfVehicle(this.sc.next());
                        this.timeValidator(dateTime);
                        this.dateValidator(dateTime);
                        System.out.println("Please enter the length of the MiniLorry (m)");
                        miniLorry.setLengthOfVehicle(this.sc.next());
                        miniLorry.setDateTime(dateTime);
                        slots.add(miniLorry);
                        --parkingSlotCount;
                        --parkingSlotCount;
                        --parkingSlotCount;
                        ++miniLorryCounter;

                        this.fileWriting(miniLorry);
                        this.returnToMenu();
                    } else {
                        System.out.println("Sorry! There's no space available for a miniLorry right now.");
                        this.returnToMenu();
                    }
                    break;
                default:
                    System.out.println("\n Your input is valid. Please try again later.");
                    this.addVehicle();
                }
            } catch (InputMismatchException var5) {
                System.out.println("\nYour input is valid. Please try again later.");
                this.sc.hasNextInt();
                this.addVehicle();
```



```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

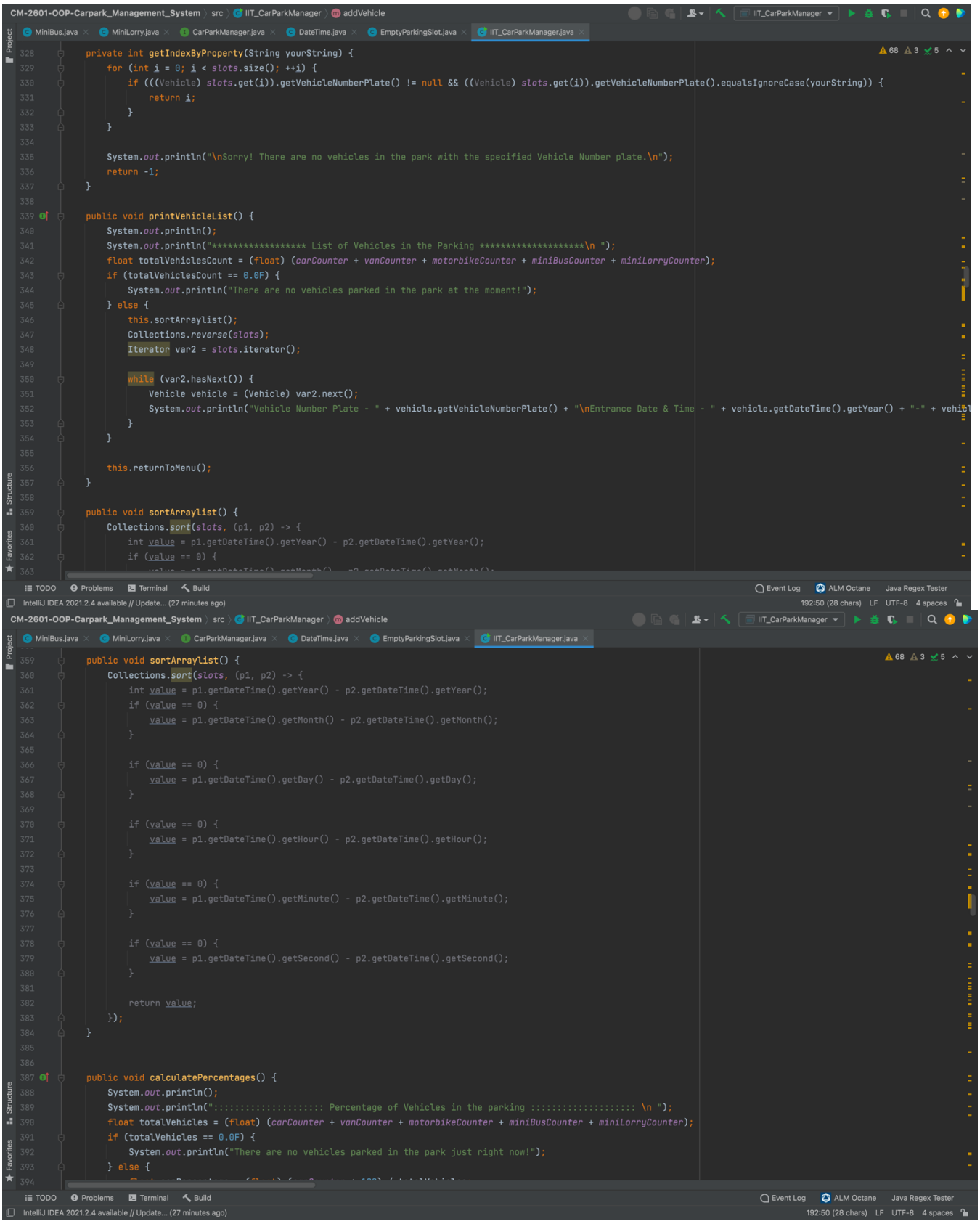
```
public void trackFreeSlots() {
    if (parkingSlotCount > 0) {
        if (parkingSlotCount == 1) {
            System.out.println("Great! There is a parking slot remaining");
        } else {
            System.out.println("There are " + parkingSlotCount + " remaining right now. \n");
        }
    } else {
        System.out.println("\nThere are no remaining slots right now. \nThank You.");
    }

}

public void deleteVehicle() {
    System.out.println();
    System.out.println("****************** Delete a vehicle ******************** \n ");
    float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
    if (totalVehicles == 0.0F) {
        System.out.println("OMG! You cannot do this deletion right now.");
        System.out.println("There are no vehicles in the park right now! Please try again later.");
    } else {
        System.out.println("Vehicle list in the parking\n");
        Iterator var2 = slots.iterator();

        while (var2.hasNext()) {
            Vehicle vehicle = (Vehicle) var2.next();
            String VehicleNumberPlate = vehicle.getVehicleNumberPlate();
            if (VehicleNumberPlate != null) {
                System.out.println("\t" + VehicleNumberPlate);
            }
        }

        System.out.println("\nPlease select the vehicle you want to delete");

        try {
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (27 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```

```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

```
        try {
            String deleteVehicle = this.sc.next();
            String leavingVehicle = ((Vehicle) slots.get(this.getIndexByProperty(deleteVehicle))).getBrandOfVehicle();
            System.out.println("\nA " + leavingVehicle + " is leaving the park right now.");
            if (leavingVehicle.equals("Car")) {
                ++parkingSlotCount;
                --carCounter;
            } else if (leavingVehicle.equals("Van")) {
                ++parkingSlotCount;
                ++parkingSlotCount;
                --vanCounter;
            } else if (leavingVehicle.equals("MotorBike")) {
                ++parkingSlotCount;
                --motorbikeCounter;
            } else if (leavingVehicle.equals("MiniBus")) {
                ++parkingSlotCount;
                ++parkingSlotCount;
                ++parkingSlotCount;
                --miniBusCounter;
            } else if (leavingVehicle.equals("MiniLorry")) {
                ++parkingSlotCount;
                ++parkingSlotCount;
                ++parkingSlotCount;
                --miniLorryCounter;
            }

            slots.remove(this.getIndexByProperty(deleteVehicle));
        } catch (Exception var5) {
        }
    }

    this.returnToMenu();
}

private int getIndexByProperty(String yourString) {
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (27 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```



```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

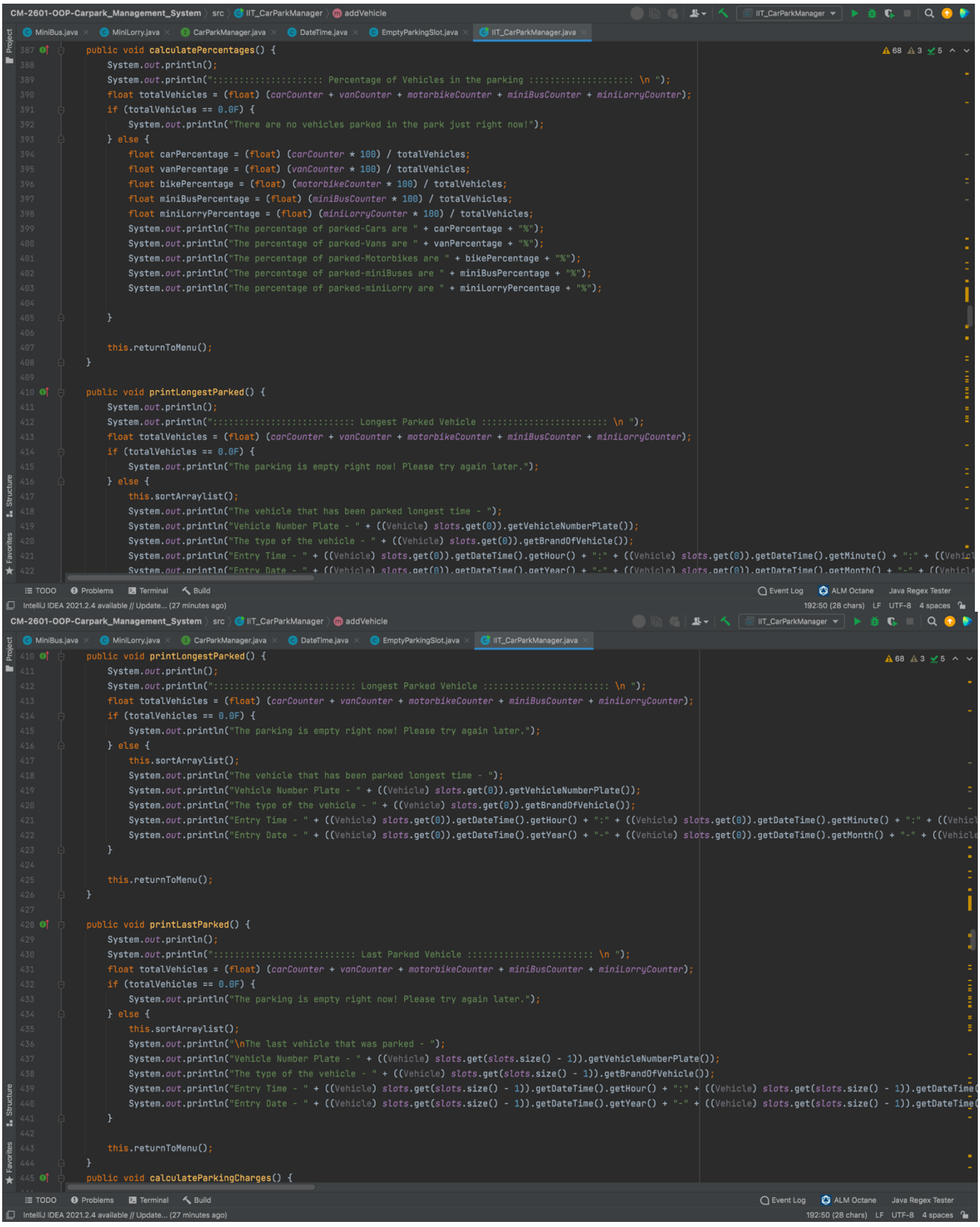
```
328     private int getIndexByProperty(String yourString) {
329         for (int i = 0; i < slots.size(); ++i) {
330             if (((Vehicle) slots.get(i)).getVehicleNumberPlate() != null && ((Vehicle) slots.get(i)).getVehicleNumberPlate().equalsIgnoreCase(yourString)) {
331                 return i;
332             }
333         }
334
335         System.out.println("\nSorry! There are no vehicles in the park with the specified Vehicle Number plate.\n");
336         return -1;
337     }
338
339     public void printVehicleList() {
340         System.out.println();
341         System.out.println("***************** List of Vehicles in the Parking ******************\n ");
342         float totalVehiclesCount = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
343         if (totalVehiclesCount == 0.0F) {
344             System.out.println("There are no vehicles parked in the park at the moment!");
345         } else {
346             this.sortArraylist();
347             Collections.reverse(slots);
348             Iterator var2 = slots.iterator();
349
350             while (var2.hasNext()) {
351                 Vehicle vehicle = (Vehicle) var2.next();
352                 System.out.println("Vehicle Number Plate - " + vehicle.getVehicleNumberPlate() + "\nEntrance Date & Time - " + vehicle.getDateTime().getYear() + "-" + vehicl
353             }
354         }
355
356         this.returnToMenu();
357     }
358
359     public void sortArraylist() {
360         Collections.sort(slots, (p1, p2) -> {
361             int value = p1.getDateTime().getYear() - p2.getDateTime().getYear();
362             if (value == 0) {
363
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (27 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```

```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

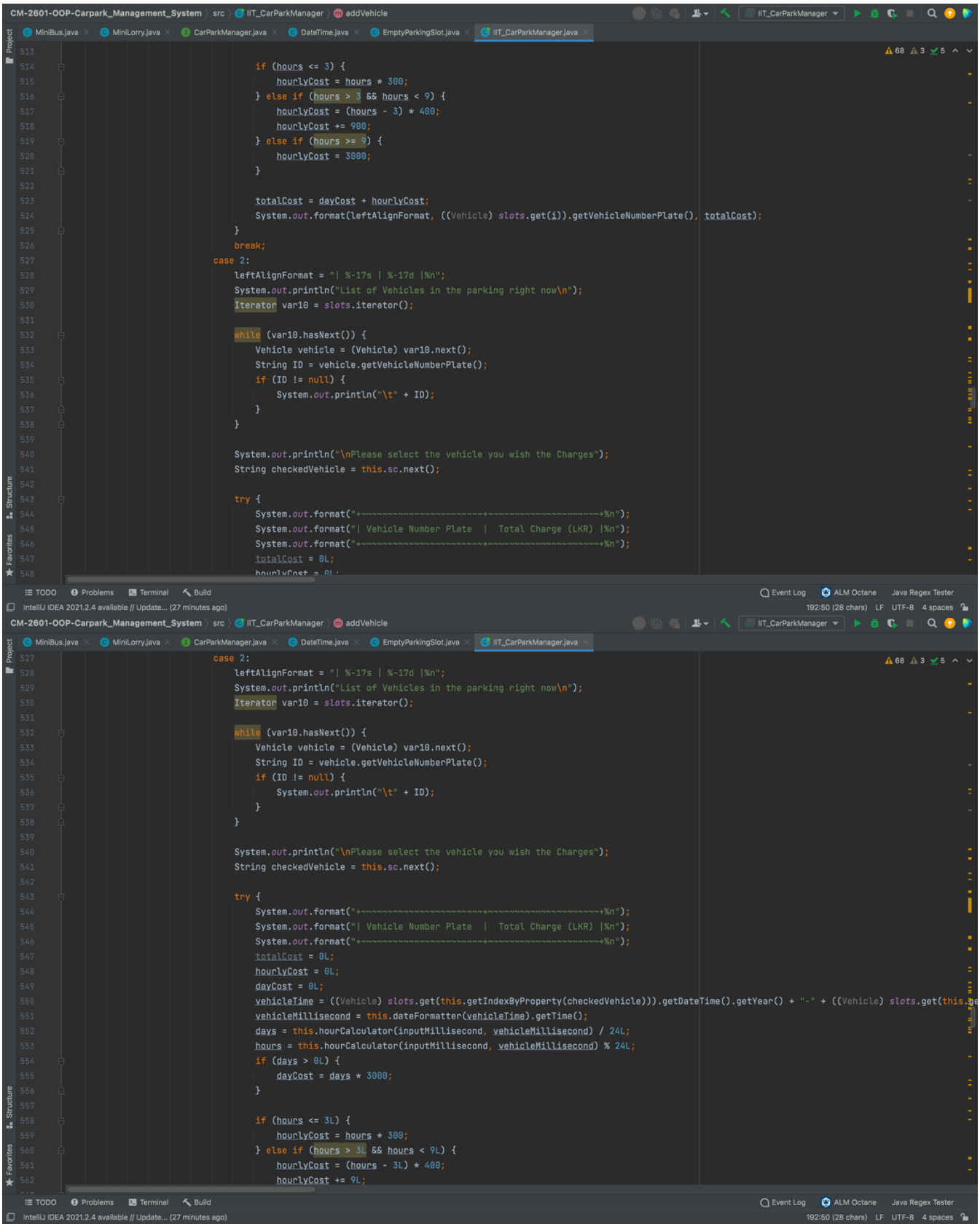
```
359     public void sortArraylist() {
360         Collections.sort(slots, (p1, p2) -> {
361             int value = p1.getDateTime().getYear() - p2.getDateTime().getYear();
362             if (value == 0) {
363                 value = p1.getDateTime().getMonth() - p2.getDateTime().getMonth();
364             }
365
366             if (value == 0) {
367                 value = p1.getDateTime().getDay() - p2.getDateTime().getDay();
368             }
369
370             if (value == 0) {
371                 value = p1.getDateTime().getHour() - p2.getDateTime().getHour();
372             }
373
374             if (value == 0) {
375                 value = p1.getDateTime().getMinute() - p2.getDateTime().getMinute();
376             }
377
378             if (value == 0) {
379                 value = p1.getDateTime().getSecond() - p2.getDateTime().getSecond();
380             }
381
382             return value;
383         });
384     }
385
386
387     public void calculatePercentages() {
388         System.out.println();
389         System.out.println("::::::::::::::::::::: Percentage of Vehicles in the parking ::::::::::::::::::: \n ");
390         float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
391         if (totalVehicles == 0.0F) {
392             System.out.println("There are no vehicles parked in the park just right now!");
393         } else {
394
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (27 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```



```
public void calculatePercentages() {
    System.out.println();
    System.out.println("::::::::::::::::::::: Percentage of Vehicles in the parking ::::::::::::::::::::: \n ");
    float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
    if (totalVehicles == 0.0F) {
        System.out.println("There are no vehicles parked in the park just right now!");
    } else {
        float carPercentage = (float) (carCounter * 100) / totalVehicles;
        float vanPercentage = (float) (vanCounter * 100) / totalVehicles;
        float bikePercentage = (float) (motorbikeCounter * 100) / totalVehicles;
        float miniBusPercentage = (float) (miniBusCounter * 100) / totalVehicles;
        float miniLorryPercentage = (float) (miniLorryCounter * 100) / totalVehicles;
        System.out.println("The percentage of parked-Cars are " + carPercentage + "%");
        System.out.println("The percentage of parked-Vans are " + vanPercentage + "%");
        System.out.println("The percentage of parked-Motorbikes are " + bikePercentage + "%");
        System.out.println("The percentage of parked-miniBuses are " + miniBusPercentage + "%");
        System.out.println("The percentage of parked-miniLorry are " + miniLorryPercentage + "%");


    }

    this.returnToMenu();
}

public void printLongestParked() {
    System.out.println();
    System.out.println("::::::::::::::::::::::::::: Longest Parked Vehicle ::::::::::::::::::::::::: \n ");
    float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
    if (totalVehicles == 0.0F) {
        System.out.println("The parking is empty right now! Please try again later.");
    } else {
        this.sortArraylist();
        System.out.println("The vehicle that has been parked longest time - ");
        System.out.println("Vehicle Number Plate - " + ((Vehicle) slots.get(0)).getVehicleNumberPlate());
        System.out.println("The type of the vehicle - " + ((Vehicle) slots.get(0)).getBrandOfVehicle());
        System.out.println("Entry Time - " + ((Vehicle) slots.get(0)).getDateTime().getHour() + ":" + ((Vehicle) slots.get(0)).getDateTime().getMinute() + ":" + ((Vehicl
        System.out.println("Entry Date - " + ((Vehicle) slots.get(0)).getDateTime().getYear() + "-" + ((Vehicle) slots.get(0)).getDateTime().getMonth() + "-" + ((Vehicle
    }

    this.returnToMenu();
}

public void printLastParked() {
    System.out.println();
    System.out.println("::::::::::::::::::::::::::: Last Parked Vehicle ::::::::::::::::::::::::: \n ");
    float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
    if (totalVehicles == 0.0F) {
        System.out.println("The parking is empty right now! Please try again later.");
    } else {
        this.sortArraylist();
        System.out.println("\nThe last vehicle that was parked - ");
        System.out.println("Vehicle Number Plate - " + ((Vehicle) slots.get(slots.size() - 1)).getVehicleNumberPlate());
        System.out.println("The type of the vehicle - " + ((Vehicle) slots.get(slots.size() - 1)).getBrandOfVehicle());
        System.out.println("Entry Time - " + ((Vehicle) slots.get(slots.size() - 1)).getDateTime().getHour() + ":" + ((Vehicle) slots.get(slots.size() - 1)).getDateTime(
        System.out.println("Entry Date - " + ((Vehicle) slots.get(slots.size() - 1)).getDateTime().getYear() + "-" + ((Vehicle) slots.get(slots.size() - 1)).getDateTime(
    }

    this.returnToMenu();
}
public void calculateParkingCharges() {
```

```java
}
public void calculateParkingCharges() {
    System.out.println();
    System.out.println(":::::::::::::::::::: Parking Charges ::::::::::::::::::::: \n ");
    float totalVehicles = (float) (carCounter + vanCounter + motorbikeCounter + miniBusCounter + miniLorryCounter);
    if (totalVehicles == 0.0F) {
        System.out.println("The parking is empty right now! Please try again later.");
    } else {
        while (true) {
            System.out.println("Please enter the current Time in the format of HH:MM:SS");
            String timeString = this.sc.next();
            String[] time = timeString.split( regex: ":");
            if (Integer.parseInt(time[0]) >= 24 || Integer.parseInt(time[1]) >= 60 || Integer.parseInt(time[2]) >= 60) {
                System.out.println("\n Invalid time format. Please try again.\n ");
            }

            if (Integer.parseInt(time[0]) < 24 && Integer.parseInt(time[1]) < 60 && Integer.parseInt(time[2]) < 60) {
                String[] date;
                do {
                    System.out.println("Please enter the current Date in the format of YYYY-MM-DD");
                    String dateString = this.sc.next();
                    date = dateString.split( regex: "-");
                    if (Integer.parseInt(date[1]) <= 12 && Integer.parseInt(date[2]) <= 31) {
                        if ((int) Math.log10((double) Integer.parseInt(date[0])) + 1 < 4) {
                            System.out.println("\n The year you typed appears to be invalid. Please try again.\n ");
                        }
                    } else {
                        System.out.println("\n Invalid date. Please try again.\n ");
                    }
                } while ((int) Math.log10((double) Integer.parseInt(date[0])) + 1 < 4 || Integer.parseInt(date[1]) > 12 || Integer.parseInt(date[2]) > 31);

                System.out.println();
                String inputTime = date[0] + "-" + date[1] + "-" + date[2] + "T" + time[0] + ":" + time[1] + ":" + time[2] + "Z";
                long inputMillisecond = this.dateFormatter(inputTime).getTime();
                System.out.println("\t\t\tParking Charges\n");

                try {
                    System.out.println("\n\tHow would you like to view the charges?\n");
                    System.out.println("\t1. Whole Car Park \t 2. Specific Vehicle");
                    String leftAlignFormat;
                    long hourlyCost;
                    long dayCost;
                    long vehicleMillisecond;
                    long days;
                    long hours;
                    String vehicleTime;
                    long totalCost;
                    switch (this.sc.nextInt()) {
                        case 1:
                            leftAlignFormat = "| %-17s | %-17d |%n";
                            System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~+%n");
                            System.out.format("|  Vehicle Number Plate  |  Total Charge (LKR) |%n");
                            System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~+%n");

                            for (
                                    int i = 0;
                                    i < slots.size();
                                    ++i
                            ) {
                                totalCost = 0;
                                hourlyCost = 0;
                                dayCost = 0;
                                vehicleTime = ((Vehicle) slots.get(i)).getDateTime().getYear() + "-" + ((Vehicle) slots.get(i)).getDateTime().getMonth() + "-" + ((Vehicl
                                vehicleMillisecond = this.dateFormatter(vehicleTime).getTime();
                                days = this.hourCalculator(inputMillisecond, vehicleMillisecond) / 24;
                                hours = this.hourCalculator(inputMillisecond, vehicleMillisecond) % 24;
                                if (days > 0) {
                                    dayCost = days * 3000;
                                }

                                if (hours <= 3) {
```

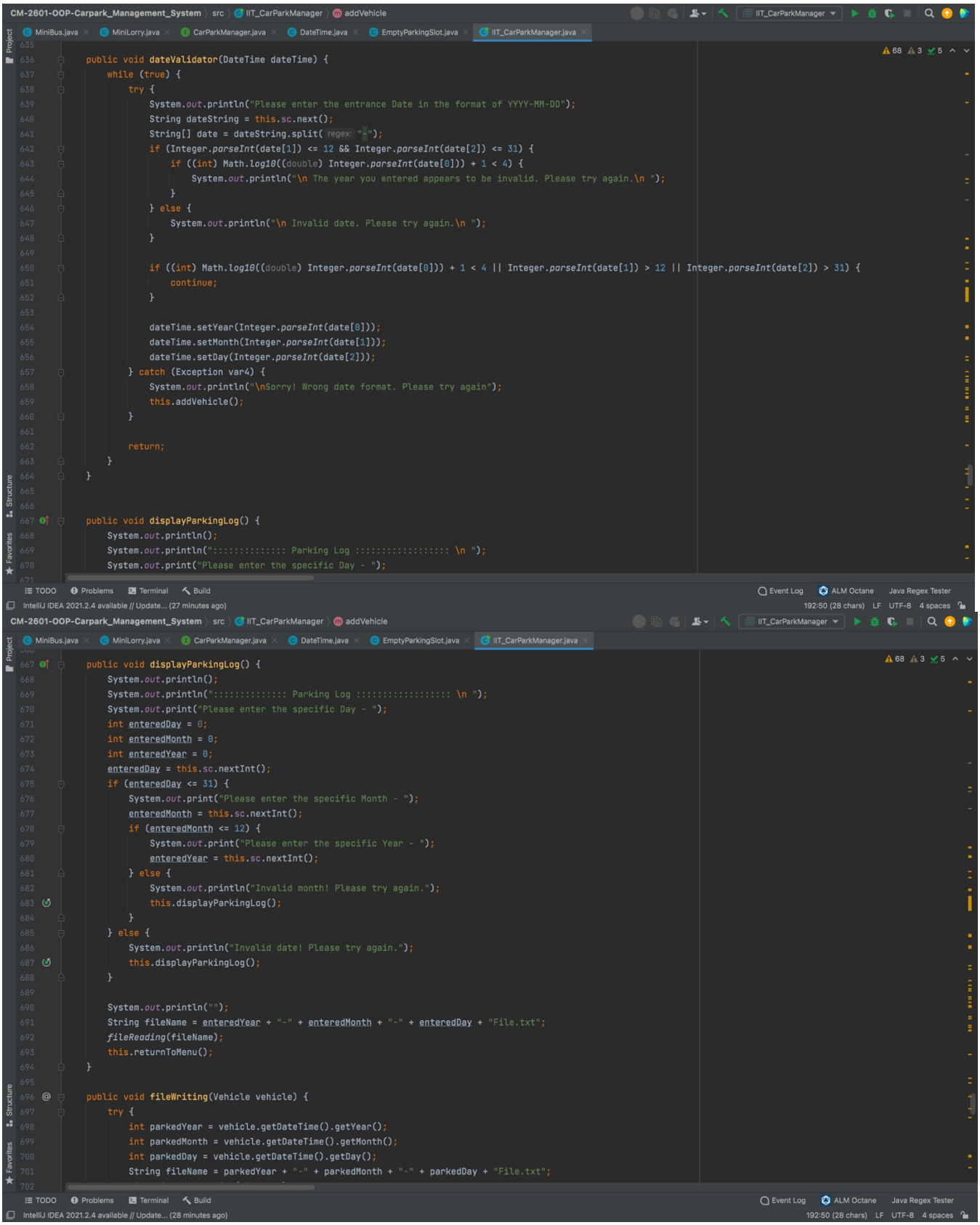


```
                if (hours <= 3) {
                    hourlyCost = hours * 300;
                } else if (hours > 3 && hours < 9) {
                    hourlyCost = (hours - 3) * 400;
                    hourlyCost += 900;
                } else if (hours >= 9) {
                    hourlyCost = 3000;
                }

                totalCost = dayCost + hourlyCost;
                System.out.format(leftAlignFormat, ((Vehicle) slots.get(i)).getVehicleNumberPlate(), totalCost);
            }
            break;
        case 2:
            leftAlignFormat = "| %-17s | %-17d |%n";
            System.out.println("List of Vehicles in the parking right now\n");
            Iterator var10 = slots.iterator();

            while (var10.hasNext()) {
                Vehicle vehicle = (Vehicle) var10.next();
                String ID = vehicle.getVehicleNumberPlate();
                if (ID != null) {
                    System.out.println("\t" + ID);
                }
            }

            System.out.println("\nPlease select the vehicle you wish the Charges");
            String checkedVehicle = this.sc.next();

            try {
                System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+%n");
                System.out.format("| Vehicle Number Plate  |  Total Charge (LKR) |%n");
                System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+%n");
                totalCost = 0L;
                hourlyCost = 0L;
```

```
        case 2:
            leftAlignFormat = "| %-17s | %-17d |%n";
            System.out.println("List of Vehicles in the parking right now\n");
            Iterator var10 = slots.iterator();

            while (var10.hasNext()) {
                Vehicle vehicle = (Vehicle) var10.next();
                String ID = vehicle.getVehicleNumberPlate();
                if (ID != null) {
                    System.out.println("\t" + ID);
                }
            }

            System.out.println("\nPlease select the vehicle you wish the Charges");
            String checkedVehicle = this.sc.next();

            try {
                System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+%n");
                System.out.format("| Vehicle Number Plate  |  Total Charge (LKR) |%n");
                System.out.format("+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~~~~~~~~~~+%n");
                totalCost = 0L;
                hourlyCost = 0L;
                dayCost = 0L;
                vehicleTime = ((Vehicle) slots.get(this.getIndexByProperty(checkedVehicle))).getDateTime().getYear() + "-" + ((Vehicle) slots.get(this.ge
                vehicleMillisecond = this.dateFormatter(vehicleTime).getTime();
                days = this.hourCalculator(inputMillisecond, vehicleMillisecond) / 24L;
                hours = this.hourCalculator(inputMillisecond, vehicleMillisecond) % 24L;
                if (days > 0L) {
                    dayCost = days * 3000;
                }

                if (hours <= 3L) {
                    hourlyCost = hours * 300;
                } else if (hours > 3L && hours < 9L) {
                    hourlyCost = (hours - 3L) * 400;
                    hourlyCost += 9L;
```

```java
                        if (hours <= 3L) {
                            hourlyCost = hours * 300;
                        } else if (hours > 3L && hours < 9L) {
                            hourlyCost = (hours - 3L) * 400;
                            hourlyCost += 9L;
                        } else if (hours >= 9L) {
                            hourlyCost = 300;
                        }

                        totalCost = dayCost + hourlyCost;
                        System.out.format(leftAlignFormat, ((Vehicle) slots.get(this.getIndexByProperty(checkedVehicle))).getVehicleNumberPlate(), totalCost);
                    } catch (Exception var24) {
                    }
                    break;
                default:
                    System.out.println("\nInvalid option!");
                    this.displayMainMenu();
            }
        } catch (Exception var25) {
            System.out.println("\nInvalid option!");
            this.displayMainMenu();
        }

        System.out.format("+~~~~~~~~~~~~~~~~~~~~+~~~~~~~~~~~~~~~~~~~~+%n");
        System.out.println("\nRates - First 3 hours - LKR 300 per hour , Starting from 4th Hour - LKR 400  per hour\n\n\t NOTE - The maximum charge for any 24 h
        break;
            }
        }
    }

    this.returnToMenu();
}

public Date dateFormatter(String dateInString) {
    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ssZ");
    Date date = null;
```

```java
public Date dateFormatter(String dateInString) {
    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ssZ");
    Date date = null;

    try {
        date = formatter.parse(dateInString.replaceAll( regex: "Z$",  replacement: "+0000"));
    } catch (ParseException var5) {
        var5.printStackTrace();
    }

    return date;
}

public long hourCalculator(long inputTime, long vehicleTime) {
    long duration = inputTime - vehicleTime;
    long hours = TimeUnit.MILLISECONDS.toHours(duration);
    return hours;
}

public void timeValidator(DateTime dateTime) {
    while (true) {
        try {
            System.out.println("Please enter the entrance Time in the format of HH:MM:SS");
            String timeString = this.sc.next();
            String[] time = timeString.split( regex: ":");
            if (Integer.parseInt(time[0]) >= 24 || Integer.parseInt(time[1]) >= 60 || Integer.parseInt(time[2]) >= 60) {
                System.out.println("\n Invalid time. Please try again.\n ");
            }

            if (Integer.parseInt(time[0]) >= 24 || Integer.parseInt(time[1]) >= 60 || Integer.parseInt(time[2]) >= 60) {
                continue;
            }

            dateTime.setHour(Integer.parseInt(time[0]));
            dateTime.setMinute(Integer.parseInt(time[1]));
            dateTime.setSecond(Integer.parseInt(time[2]));
```



```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

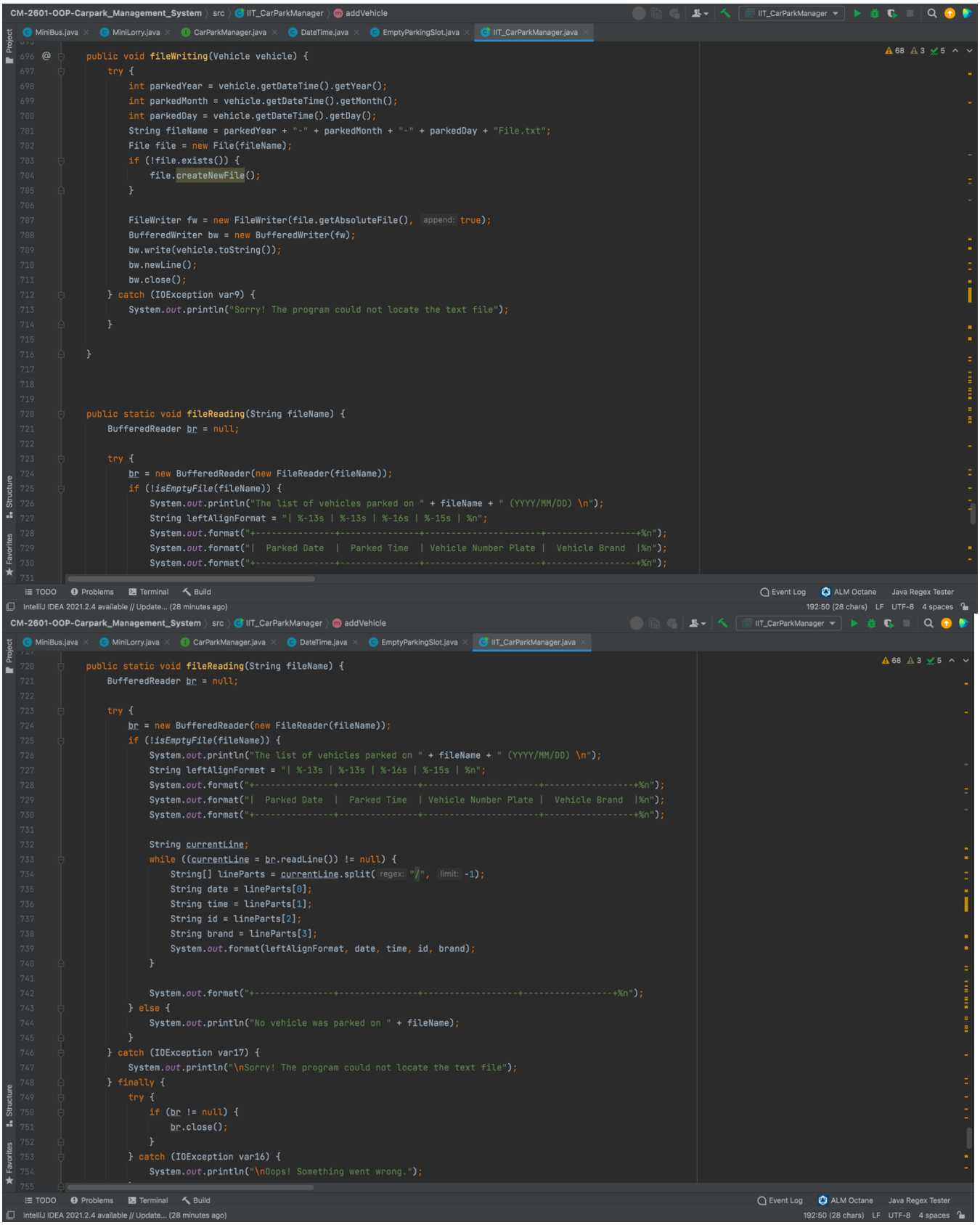
```
635
636     public void dateValidator(DateTime dateTime) {
637         while (true) {
638             try {
639                 System.out.println("Please enter the entrance Date in the format of YYYY-MM-DD");
640                 String dateString = this.sc.next();
641                 String[] date = dateString.split( regex: "-");
642                 if (Integer.parseInt(date[1]) <= 12 && Integer.parseInt(date[2]) <= 31) {
643                     if ((int) Math.log10((double) Integer.parseInt(date[0])) + 1 < 4) {
644                         System.out.println("\n The year you entered appears to be invalid. Please try again.\n ");
645                     }
646                 } else {
647                     System.out.println("\n Invalid date. Please try again.\n ");
648                 }
649
650                 if ((int) Math.log10((double) Integer.parseInt(date[0])) + 1 < 4 || Integer.parseInt(date[1]) > 12 || Integer.parseInt(date[2]) > 31) {
651                     continue;
652                 }
653
654                 dateTime.setYear(Integer.parseInt(date[0]));
655                 dateTime.setMonth(Integer.parseInt(date[1]));
656                 dateTime.setDay(Integer.parseInt(date[2]));
657             } catch (Exception var4) {
658                 System.out.println("\nSorry! Wrong date format. Please try again");
659                 this.addVehicle();
660             }
661
662             return;
663         }
664     }
665
666
667     public void displayParkingLog() {
668         System.out.println();
669         System.out.println("::::::::::::::: Parking Log :::::::::::::::::: \n ");
670         System.out.print("Please enter the specific Day - ");
671
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (27 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```

```
CM-2601-OOP-Carpark_Management_System › src › IIT_CarParkManager › addVehicle
MiniBus.java | MiniLorry.java | CarParkManager.java | DateTime.java | EmptyParkingSlot.java | IIT_CarParkManager.java
```

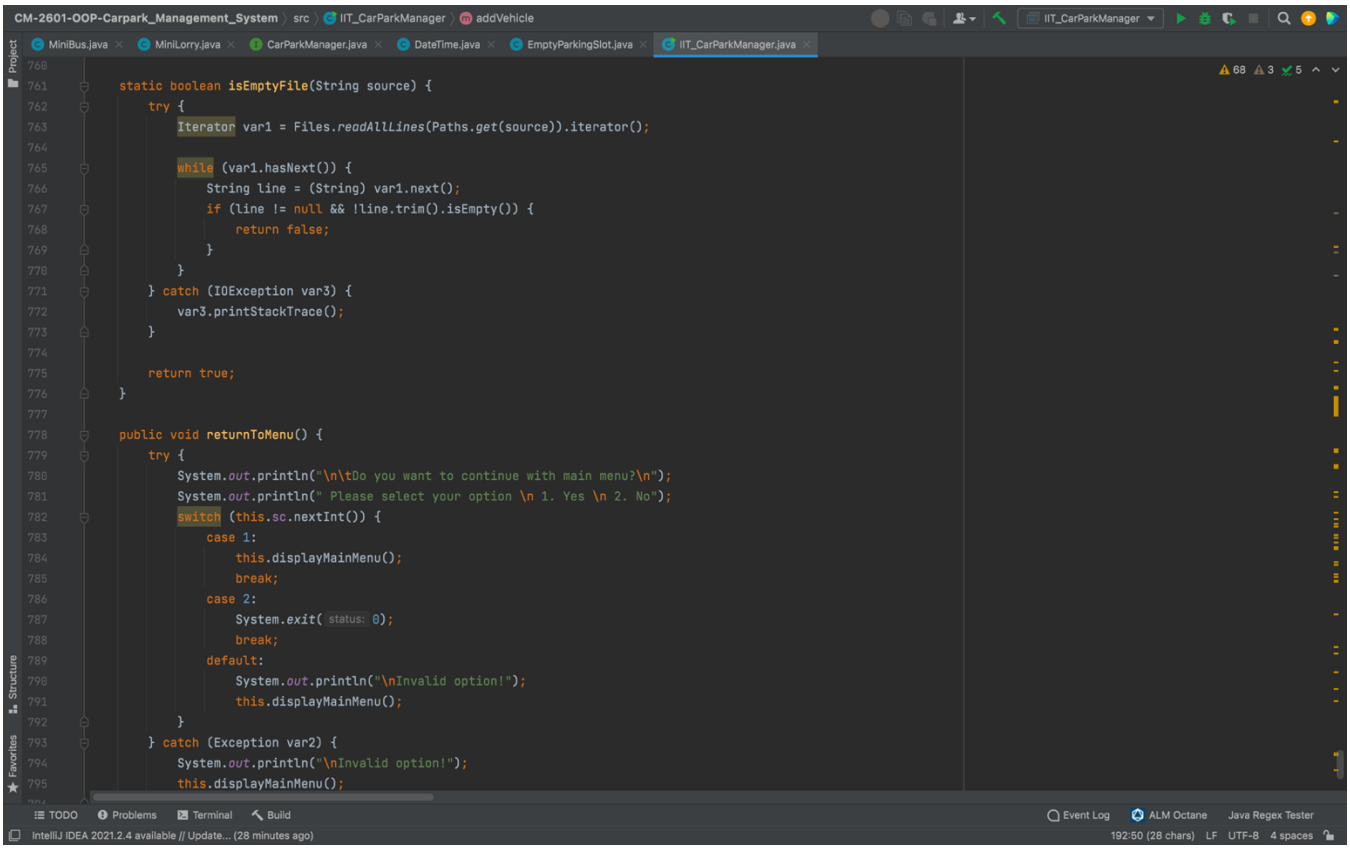
```
667     public void displayParkingLog() {
668         System.out.println();
669         System.out.println("::::::::::::::: Parking Log :::::::::::::::::: \n ");
670         System.out.print("Please enter the specific Day - ");
671         int enteredDay = 0;
672         int enteredMonth = 0;
673         int enteredYear = 0;
674         enteredDay = this.sc.nextInt();
675         if (enteredDay <= 31) {
676             System.out.print("Please enter the specific Month - ");
677             enteredMonth = this.sc.nextInt();
678             if (enteredMonth <= 12) {
679                 System.out.print("Please enter the specific Year - ");
680                 enteredYear = this.sc.nextInt();
681             } else {
682                 System.out.println("Invalid month! Please try again.");
683                 this.displayParkingLog();
684             }
685         } else {
686             System.out.println("Invalid date! Please try again.");
687             this.displayParkingLog();
688         }
689
690         System.out.println("");
691         String fileName = enteredYear + "-" + enteredMonth + "-" + enteredDay + "File.txt";
692         fileReading(fileName);
693         this.returnToMenu();
694     }
695
696     public void fileWriting(Vehicle vehicle) {
697         try {
698             int parkedYear = vehicle.getDateTime().getYear();
699             int parkedMonth = vehicle.getDateTime().getMonth();
700             int parkedDay = vehicle.getDateTime().getDay();
701             String fileName = parkedYear + "-" + parkedMonth + "-" + parkedDay + "File.txt";
702
```

```
TODO | Problems | Terminal | Build | Event Log | ALM Octane | Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (28 minutes ago)
192:50 (28 chars) LF UTF-8 4 spaces
```



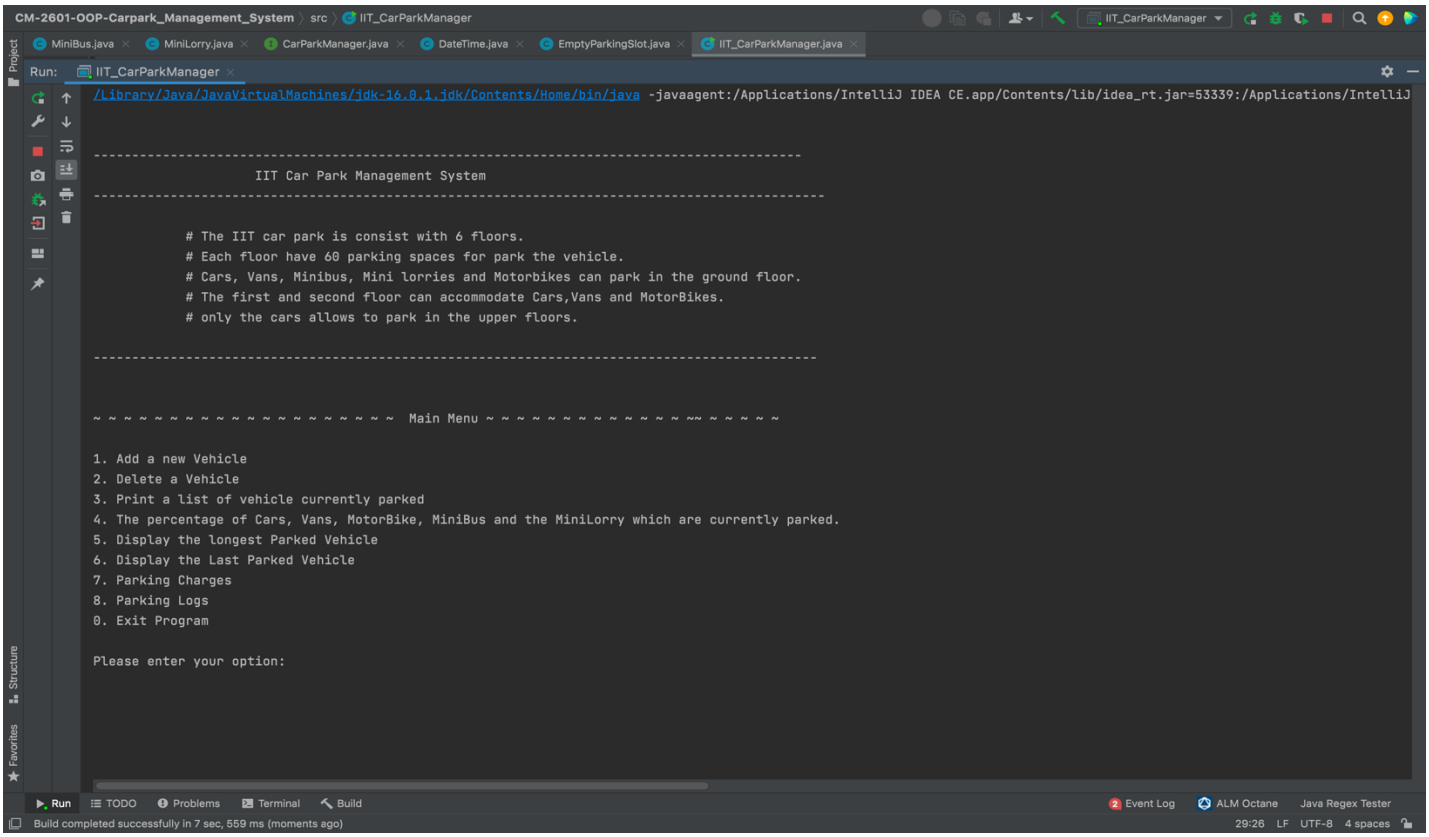
```
public void fileWriting(Vehicle vehicle) {
    try {
        int parkedYear = vehicle.getDateTime().getYear();
        int parkedMonth = vehicle.getDateTime().getMonth();
        int parkedDay = vehicle.getDateTime().getDay();
        String fileName = parkedYear + "-" + parkedMonth + "-" + parkedDay + "File.txt";
        File file = new File(fileName);
        if (!file.exists()) {
            file.createNewFile();
        }

        FileWriter fw = new FileWriter(file.getAbsoluteFile(), append: true);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(vehicle.toString());
        bw.newLine();
        bw.close();
    } catch (IOException var9) {
        System.out.println("Sorry! The program could not locate the text file");
    }

}



public static void fileReading(String fileName) {
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader(fileName));
        if (!isEmptyFile(fileName)) {
            System.out.println("The list of vehicles parked on " + fileName + " (YYYY/MM/DD) \n");
            String leftAlignFormat = "| %-13s | %-13s | %-16s | %-15s | %n";
            System.out.format("+---------------+---------------+----------------------+-----------------+%n");
            System.out.format("|  Parked Date  |  Parked Time  | Vehicle Number Plate |  Vehicle Brand  |%n");
            System.out.format("+---------------+---------------+----------------------+-----------------+%n");
```

```
public static void fileReading(String fileName) {
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader(fileName));
        if (!isEmptyFile(fileName)) {
            System.out.println("The list of vehicles parked on " + fileName + " (YYYY/MM/DD) \n");
            String leftAlignFormat = "| %-13s | %-13s | %-16s | %-15s | %n";
            System.out.format("+---------------+---------------+----------------------+-----------------+%n");
            System.out.format("|  Parked Date  |  Parked Time  | Vehicle Number Plate |  Vehicle Brand  |%n");
            System.out.format("+---------------+---------------+----------------------+-----------------+%n");

            String currentLine;
            while ((currentLine = br.readLine()) != null) {
                String[] lineParts = currentLine.split( regex: "/", limit: -1);
                String date = lineParts[0];
                String time = lineParts[1];
                String id = lineParts[2];
                String brand = lineParts[3];
                System.out.format(leftAlignFormat, date, time, id, brand);
            }

            System.out.format("+---------------+---------------+----------------------+-----------------+%n");
        } else {
            System.out.println("No vehicle was parked on " + fileName);
        }
    } catch (IOException var17) {
        System.out.println("\nSorry! The program could not locate the text file");
    } finally {
        try {
            if (br != null) {
                br.close();
            }
        } catch (IOException var16) {
            System.out.println("\nOops! Something went wrong.");
```

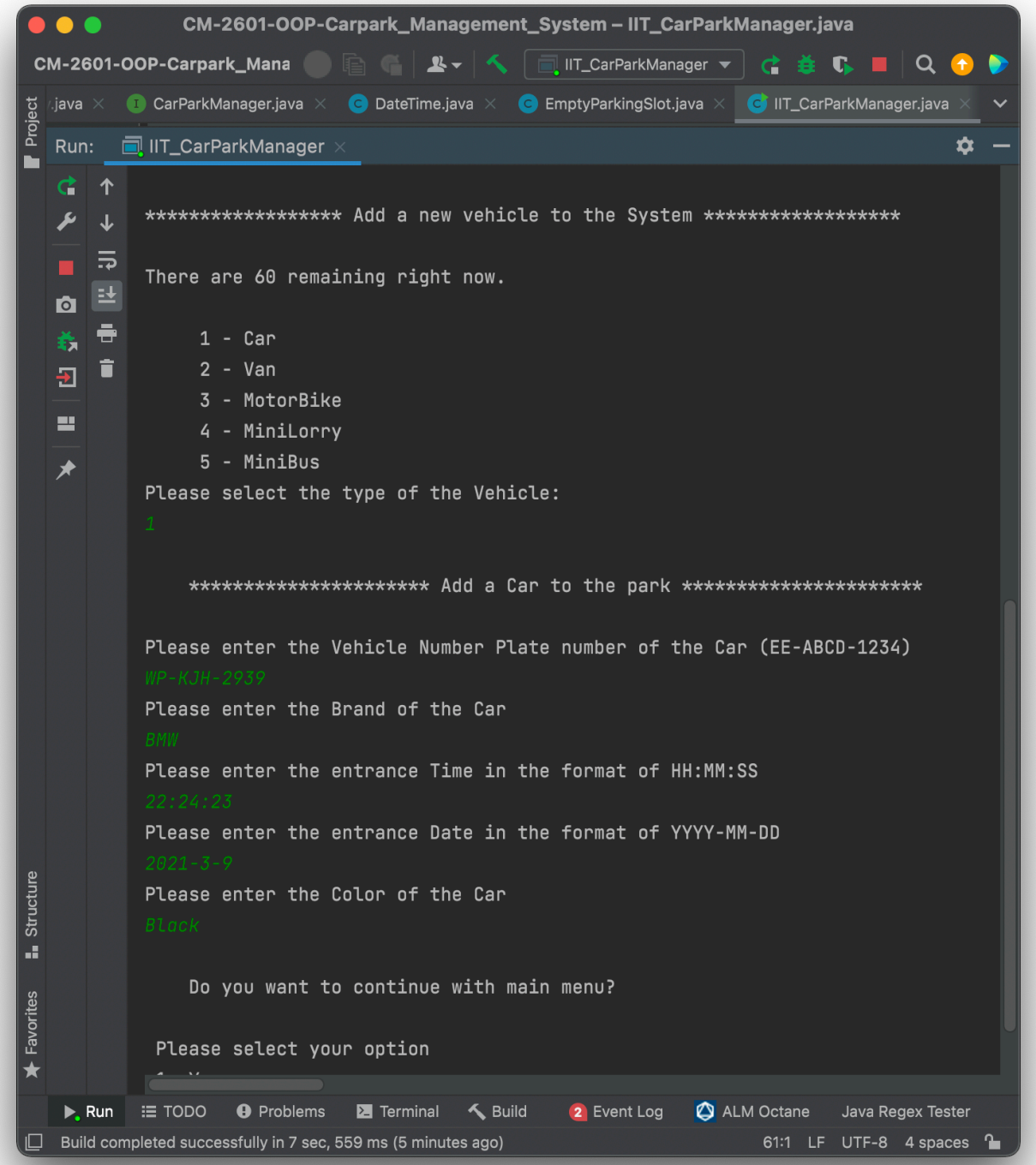

CM-2601-OOP-Carpark_Management_System
src
IIT_CarParkManager
addVehicle
MiniBus.java
MiniLorry.java
CarParkManager.java
DateTime.java
EmptyParkingSlot.java
IIT_CarParkManager.java
static boolean isEmptyFile(String source) {
try {
Iterator var1 = Files.readAllLines(Paths.get(source)).iterator();
while (var1.hasNext()) {
String line = (String) var1.next();
if (line != null && !line.trim().isEmpty()) {
return false;
}
}
} catch (IOException var3) {
var3.printStackTrace();
}
return true;
}
public void returnToMenu() {
try {
System.out.println("\n\tDo you want to continue with main menu?\n");
System.out.println(" Please select your option \n 1. Yes \n 2. No");
switch (this.sc.nextInt()) {
case 1:
this.displayMainMenu();
break;
case 2:
System.exit(status: 0);
break;
default:
System.out.println("\nInvalid option!");
this.displayMainMenu();
}
} catch (Exception var2) {
System.out.println("\nInvalid option!");
this.displayMainMenu();
TODO
Problems
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
IntelliJ IDEA 2021.2.4 available // Update... (28 minutes ago)
192:50 (28 chars)
LF
UTF-8
4 spaces

Output snippets

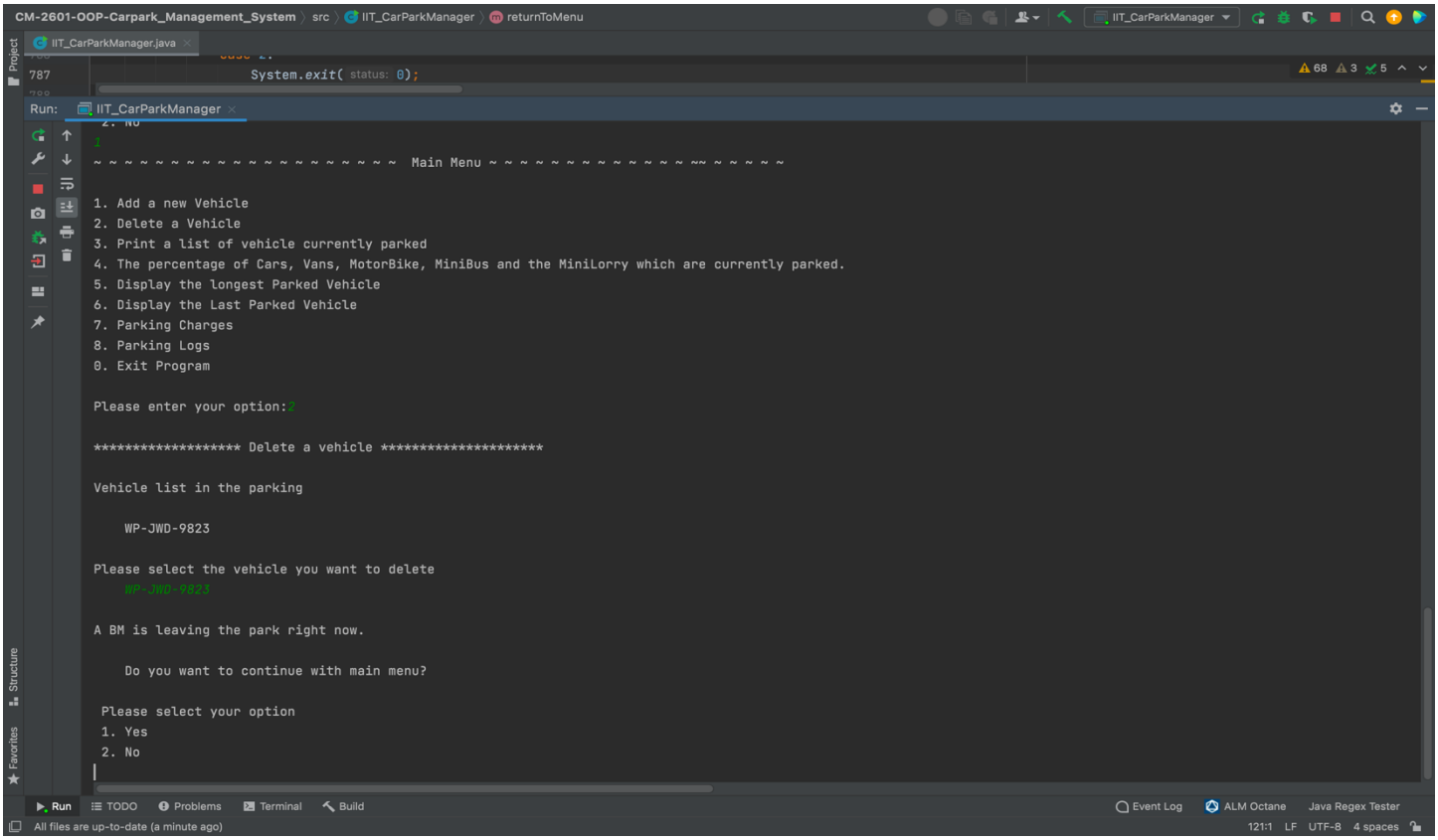
Main Menu



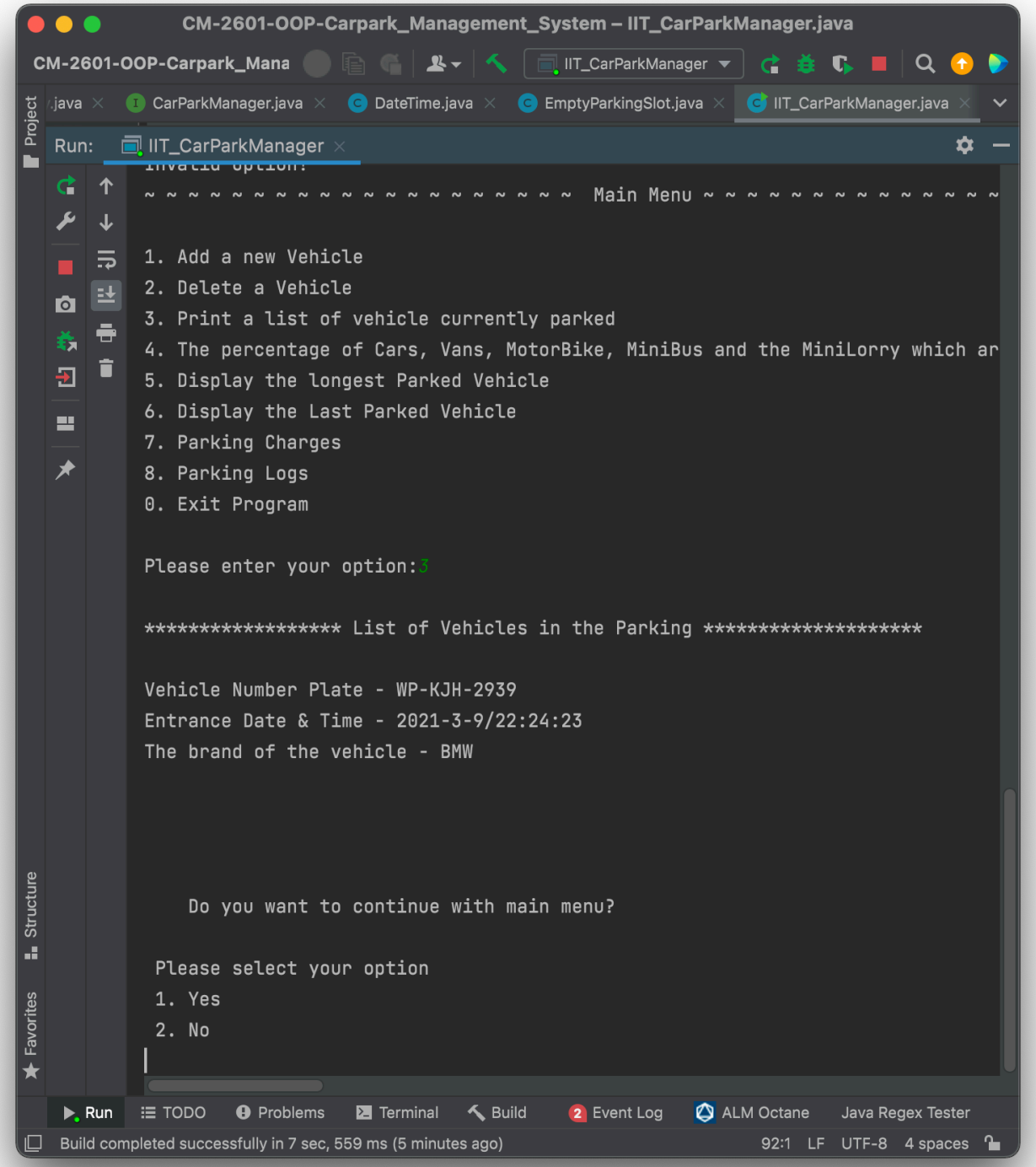
Add Vehicle


CM-2601-OOP-Carpark_Management_System – IIT_CarParkManager.java
CM-2601-OOP-Carpark_Mana
IIT_CarParkManager
.java
CarParkManager.java
DateTime.java
EmptyParkingSlot.java
IIT_CarParkManager.java
Run:
IIT_CarParkManager
********** Add a new vehicle to the System **********
There are 60 remaining right now.
1 - Car
2 - Van
3 - MotorBike
4 - MiniLorry
5 - MiniBus
Please select the type of the Vehicle:
1
********** Add a Car to the park **********
Please enter the Vehicle Number Plate number of the Car (EE-ABCD-1234)
WP-KJH-2939
Please enter the Brand of the Car
BMW
Please enter the entrance Time in the format of HH:MM:SS
22:24:23
Please enter the entrance Date in the format of YYYY-MM-DD
2021-3-9
Please enter the Color of the Car
Black
Do you want to continue with main menu?
Please select your option
Run
TODO
Problems
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
Build completed successfully in 7 sec, 559 ms (5 minutes ago)
61:1
LF
UTF-8
4 spaces
Project
Structure
Favorites

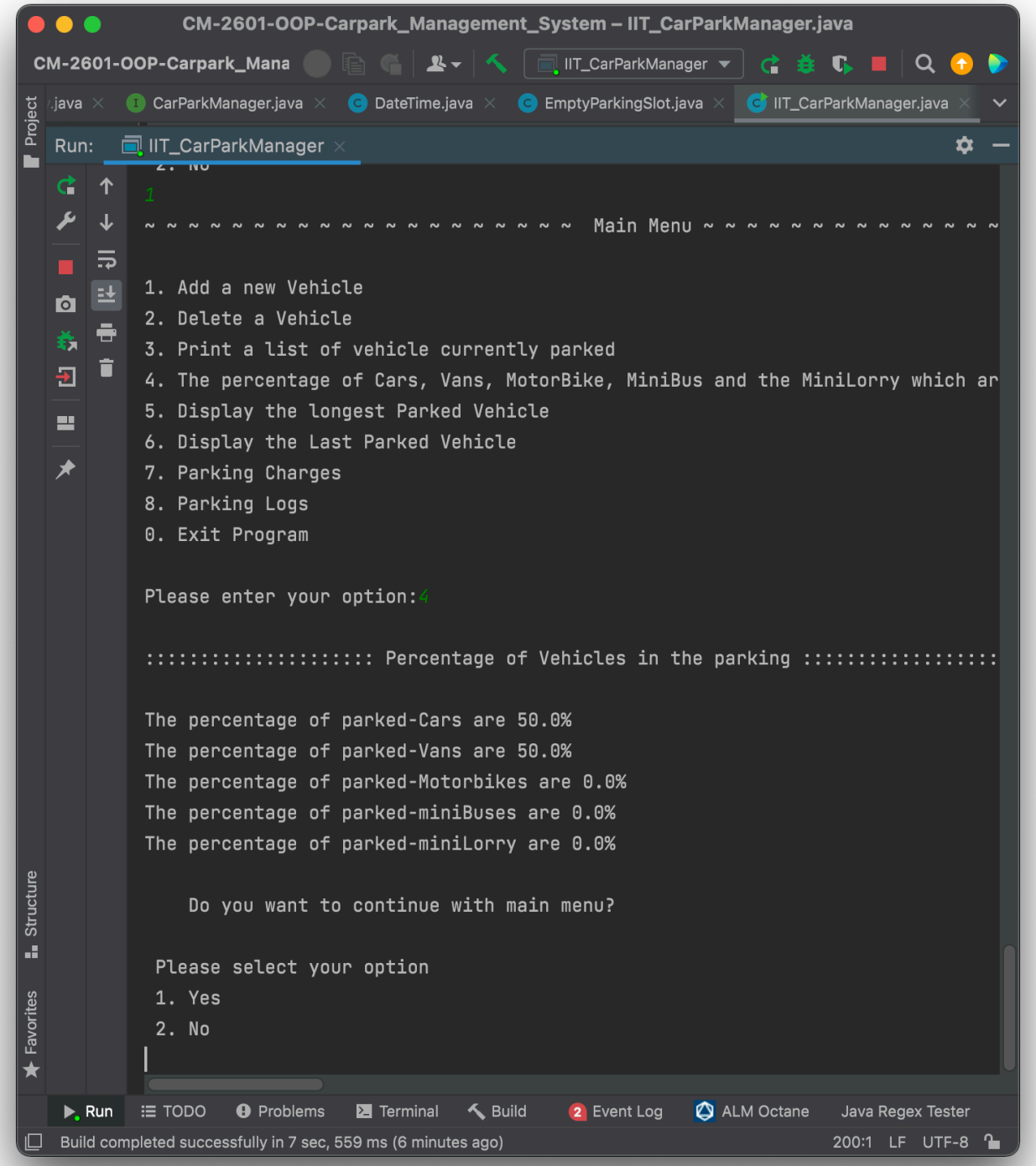
Delete Vehicle



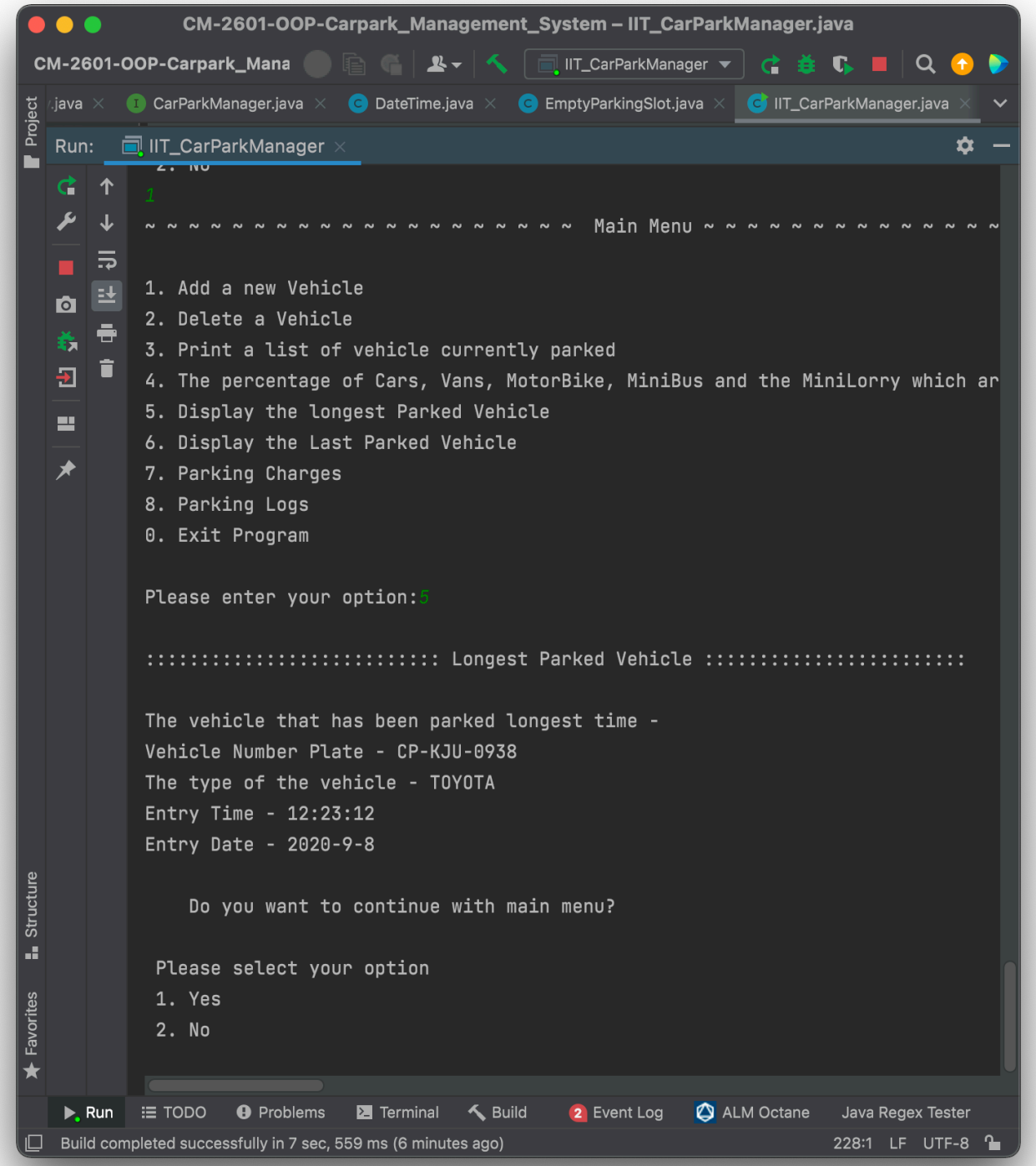
currently Parked Vehicle



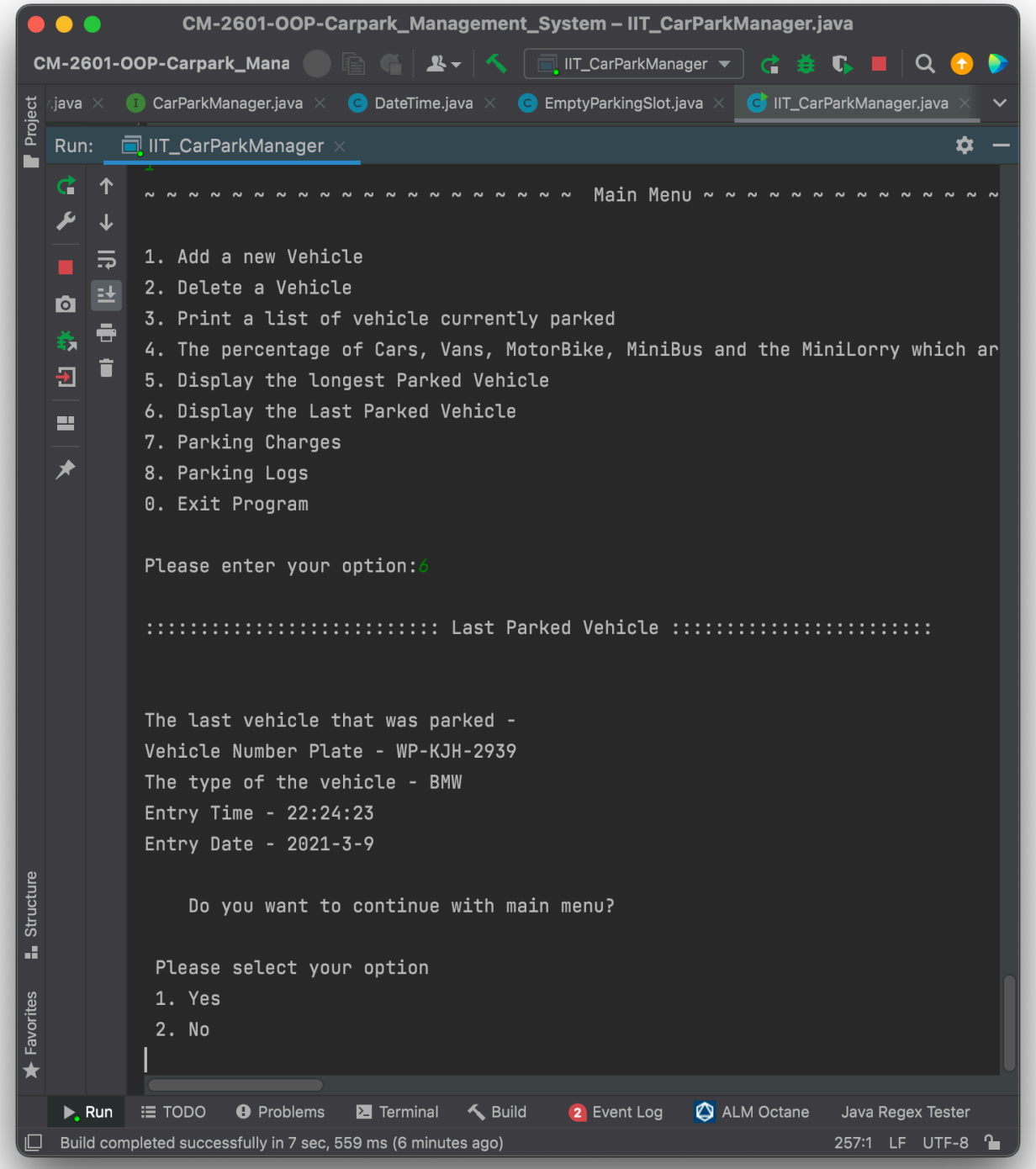
Percentage of Vehicles



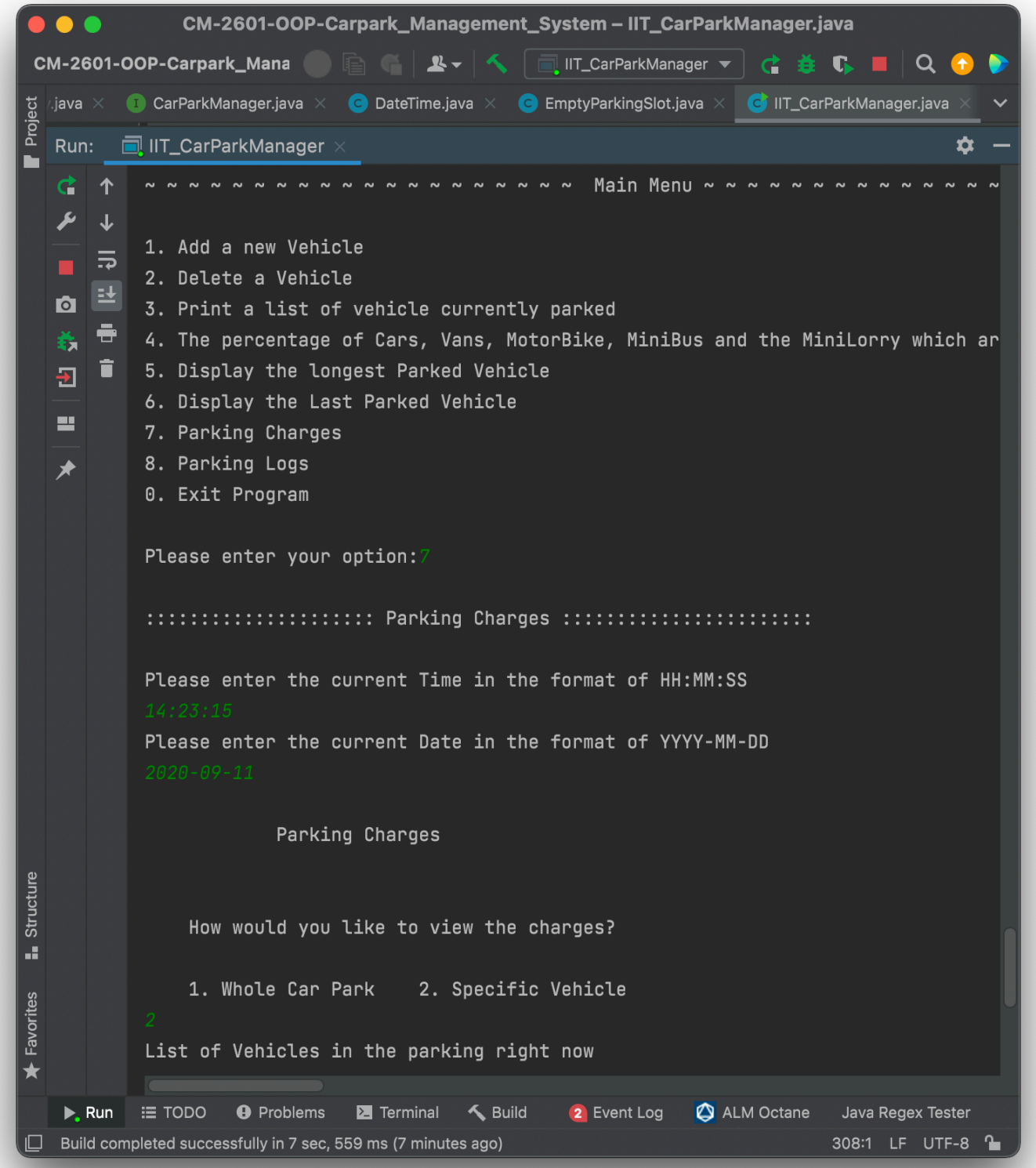
Longest Parked Vehicle

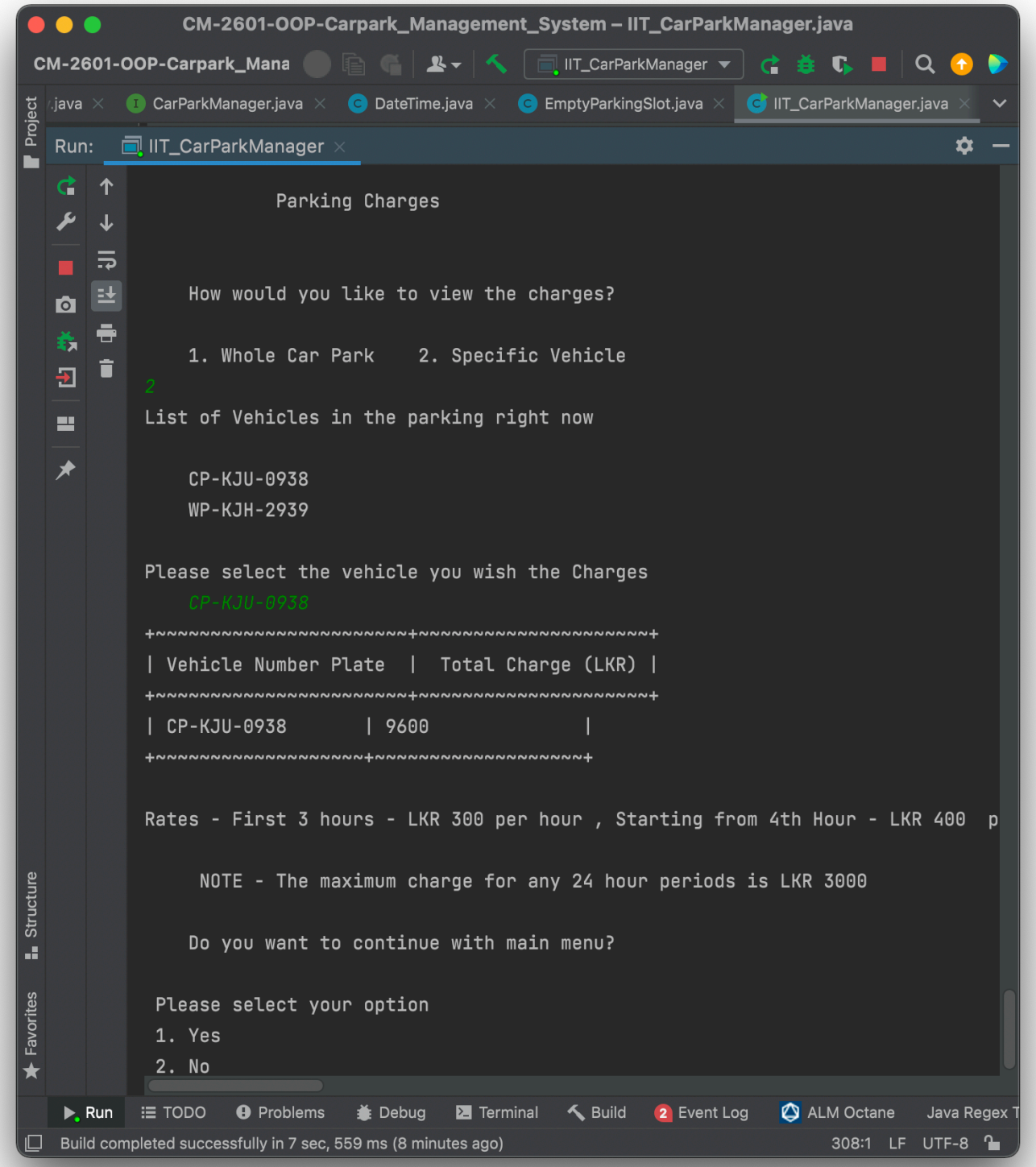


Last Parked Vehicle

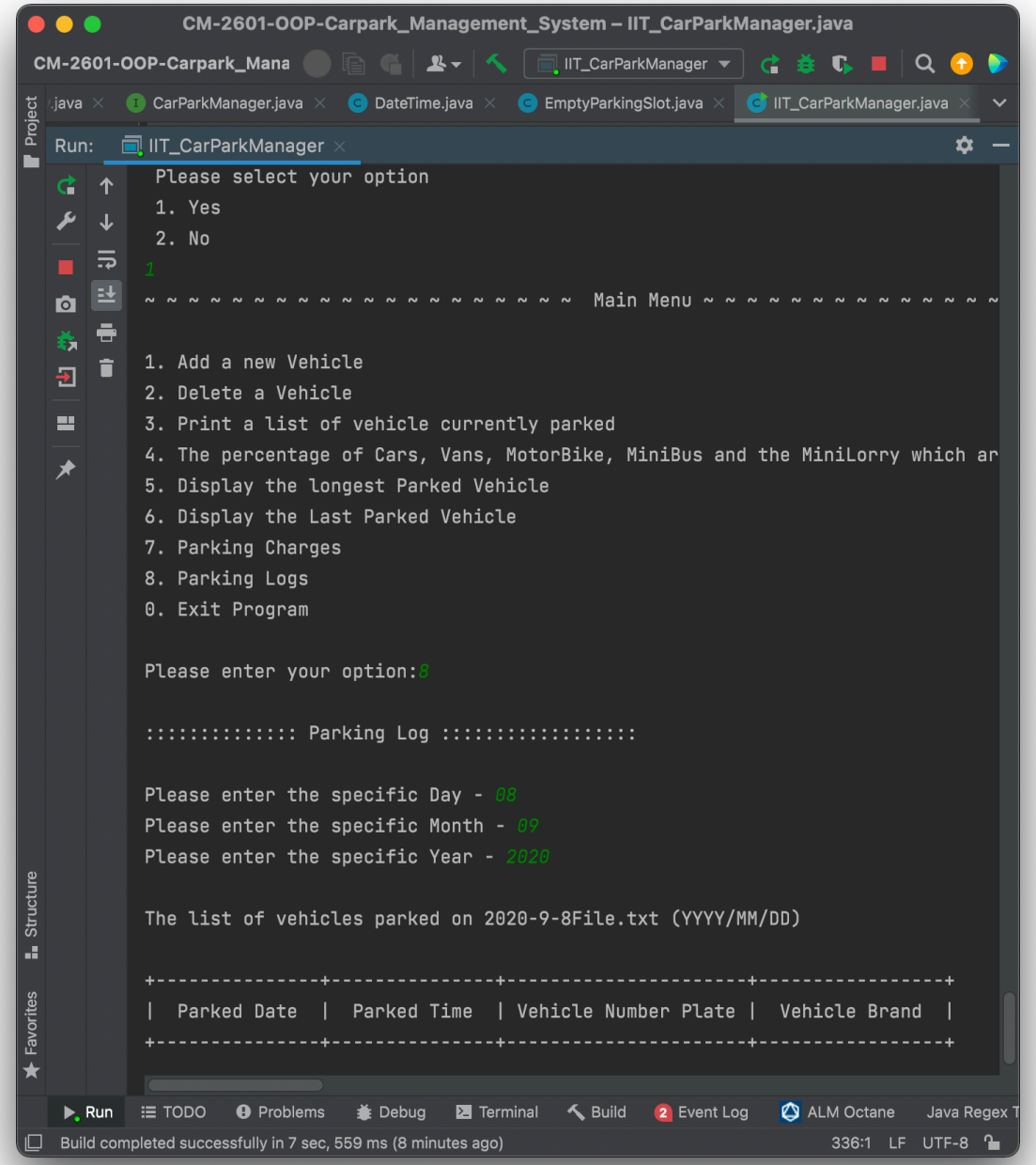


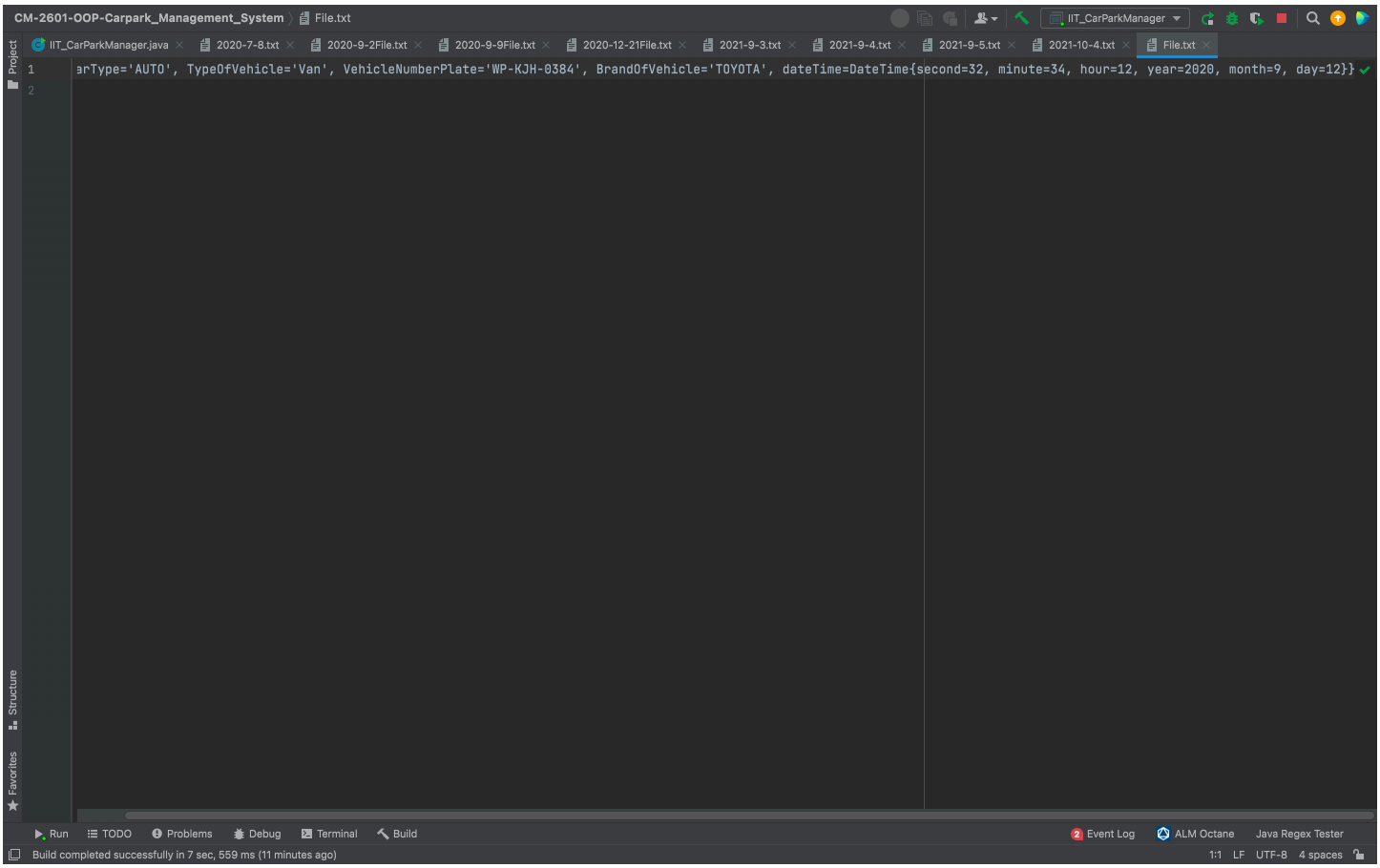
Print Parking Charges


CM-2601-OOP-Carpark_Management_System – IIT_CarParkManager.java
CM-2601-OOP-Carpark_Mana
IIT_CarParkManager
.java
CarParkManager.java
DateTime.java
EmptyParkingSlot.java
IIT_CarParkManager.java
Run:
IIT_CarParkManager
~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ Main Menu ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1. Add a new Vehicle
2. Delete a Vehicle
3. Print a list of vehicle currently parked
4. The percentage of Cars, Vans, MotorBike, MiniBus and the MiniLorry which ar
5. Display the longest Parked Vehicle
6. Display the Last Parked Vehicle
7. Parking Charges
8. Parking Logs
0. Exit Program
Please enter your option:7
:::::::::::::::::::: Parking Charges ::::::::::::::::::::
Please enter the current Time in the format of HH:MM:SS
14:23:15
Please enter the current Date in the format of YYYY-MM-DD
2020-09-11
Parking Charges
How would you like to view the charges?
1. Whole Car Park 2. Specific Vehicle
2
List of Vehicles in the parking right now
Run
TODO
Problems
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
Build completed successfully in 7 sec, 559 ms (7 minutes ago)
308:1
LF
UTF-8
Project
Structure
Favorites


CM-2601-OOP-Carpark_Management_System – IIT_CarParkManager.java
CM-2601-OOP-Carpark_Mana
IIT_CarParkManager
java
CarParkManager.java
DateTime.java
EmptyParkingSlot.java
IIT_CarParkManager.java
Run:
IIT_CarParkManager
Parking Charges
How would you like to view the charges?
1. Whole Car Park 2. Specific Vehicle
2
List of Vehicles in the parking right now
CP-KJU-0938
WP-KJH-2939
Please select the vehicle you wish the Charges
CP-KJU-0938
| Vehicle Number Plate | Total Charge (LKR) |
| CP-KJU-0938 | 9600 |
Rates - First 3 hours - LKR 300 per hour , Starting from 4th Hour - LKR 400 p
NOTE - The maximum charge for any 24 hour periods is LKR 3000
Do you want to continue with main menu?
Please select your option
1. Yes
2. No
Run
TODO
Problems
Debug
Terminal
Build
Event Log
ALM Octane
Java Regex T
Build completed successfully in 7 sec, 559 ms (8 minutes ago)
308:1
LF
UTF-8
Project
Structure
Favorites

Parking Log


CM-2601-OOP-Carpark_Management_System – IIT_CarParkManager.java
CM-2601-OOP-Carpark_Mana
IIT_CarParkManager
.java
CarParkManager.java
DateTime.java
EmptyParkingSlot.java
IIT_CarParkManager.java
Run:
IIT_CarParkManager
Please select your option
1. Yes
2. No
1
~ Main Menu ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1. Add a new Vehicle
2. Delete a Vehicle
3. Print a list of vehicle currently parked
4. The percentage of Cars, Vans, MotorBike, MiniBus and the MiniLorry which ar
5. Display the longest Parked Vehicle
6. Display the Last Parked Vehicle
7. Parking Charges
8. Parking Logs
0. Exit Program
Please enter your option:8
:::::::::::::: Parking Log ::::::::::::::::::
Please enter the specific Day - 08
Please enter the specific Month - 09
Please enter the specific Year - 2020
The list of vehicles parked on 2020-9-8File.txt (YYYY/MM/DD)
+--------------+--------------+--------------------+----------------+
| Parked Date | Parked Time | Vehicle Number Plate | Vehicle Brand |
+--------------+--------------+--------------------+----------------+
Project
Structure
Favorites
Run
TODO
Problems
Debug
Terminal
Build
Event Log
ALM Octane
Java Regex T
Build completed successfully in 7 sec, 559 ms (8 minutes ago)
336:1
LF
UTF-8


CM-2601-OOP-Carpark_Management_System
File.txt
IIT_CarParkManager
IIT_CarParkManager.java
2020-7-8.txt
2020-9-2File.txt
2020-9-9File.txt
2020-12-21File.txt
2021-9-3.txt
2021-9-4.txt
2021-9-5.txt
2021-10-4.txt
File.txt
arType='AUTO', TypeOfVehicle='Van', VehicleNumberPlate='WP-KJH-0384', BrandOfVehicle='TOYOTA', dateTime=DateTime{second=32, minute=34, hour=12, year=2020, month=9, day=12}}
Project
Structure
Favorites
Run
TODO
Problems
Debug
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
Build completed successfully in 7 sec, 559 ms (11 minutes ago)
1:1
LF
UTF-8
4 spaces

Exit


1. Add a new Vehicle
2. Delete a Vehicle
3. Print a list of vehicle currently parked
4. The percentage of Cars, Vans, MotorBike, MiniBus and the MiniLorry which are currently parked.
5. Display the longest Parked Vehicle
6. Display the Last Parked Vehicle
7. Parking Charges
8. Parking Logs
0. Exit Program
Please enter your option:0
Process finished with exit code 0
Structure
Favorites
Run
TODO
Problems
Debug
Terminal
Build
Event Log
ALM Octane
Java Regex Tester
Build completed successfully in 7 sec, 559 ms (12 minutes ago)
352:1
LF
UTF-8
4 spaces