BINUS UNIVERSITY INTERNATIONAL

Parking Management System

Troy Poetra Prajoga | ID: 2702291910 | Object Oriented Programming

Table of Contents

Project Specifications

Project Overview

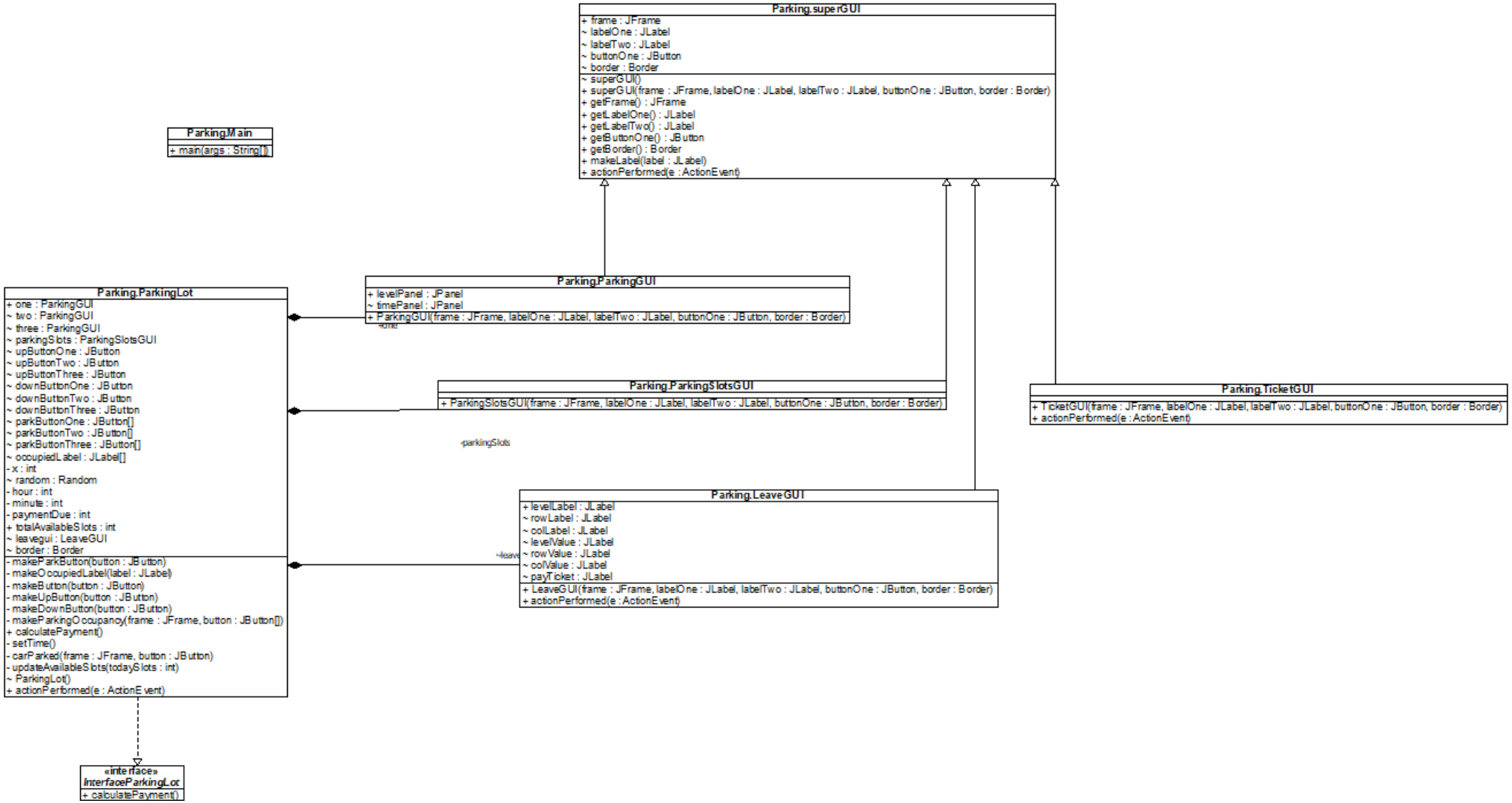
The Program created is a Parking Lot Management System. This program is designed to manage the parking operations of a parking lot, as well as simulate the parking experience. The system provides functionality for tracking and managing parking slots, calculating payment for parked vehicles, and displaying real-time information about parking occupancy.

Objectives

The main objectives of the projects are as follows:

1. Design a user-friendly graphical user interface(GUI) for the parking system.
2. Manage the occupancy and availability of the parking slots within the parking lot.
3. Calculate the payment due for a parked vehicle based on parking duration.
4. Store and retrieve parking data from a text file for data persistence across multiple program runs.
5. Allow users to navigate through different parking levels within the parking lot.
6. Provide real-time updates on availability of parking slots.

Solution Design (Class Diagram)



Solution Design

In this program, the classes that are included are Main.java, superGUI.java, ParkingSlotsGUI.java, TicketGUI.java, ParkingGUI.java, LeaveGUI.java, and ParkingLot.Java. The GUIs were created using Java Swing. The interface that is included is InterfaceParkingLot.java. The txt file that is included is AvailableParkingSlots.txt.

Main.java

The Main.java class is the main class that runs the whole program. It starts by creating a new instance of the TicketGUI class. Since the TicketGUI uses Java Swing, the imports used in this class are javax.swing.* and java.awt*.

```java
package Parking;

import javax.swing.*;
import java.awt.*;


no usages
public class Main {
    no usages
    public static void main(String[] args)
    {
        TicketGUI ticket = new TicketGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
    }
}
```

superGUI.java

The superGUI.java class is the parent class for the GUIs. It contains the main attributes that most of the GUIs have so that the child classes can extend from this class. This class contains one JFrame, two JLabels, one JButton, and one Border.

```java
package Parking;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
4 usages  4 inheritors
public class superGUI implements ActionListener
{
    //initializing elements that the GUIs will use
    2 usages
    JFrame frame = new JFrame();
    2 usages
    JLabel labelOne = new JLabel();
    2 usages
    JLabel labelTwo = new JLabel();
    2 usages
    JButton buttonOne = new JButton();
    2 usages
    Border border = BorderFactory.createLineBorder(Color.BLACK, thickness: 1);
    no usages
    superGUI(){}
```

This class also includes a constructor that initializes the GUI attributes, and also getter methods such as getFrame( ), getLabelOne( ), getLabelTwo( ), getButtonOne( ), and getBorder( ). This class also includes a setter method called makeLabel(JLabel label) which creates the general label attributes. The last thing that this class has is an actionPerformed method for the child classes.

```
public superGUI(JFrame frame,JLabel labelOne,JLabel labelTwo,JButton buttonOne,Border border)
{
    //initialize objects of class
    this.frame = frame;
    this.labelOne = labelOne;
    this.labelTwo = labelTwo;
    this.buttonOne = buttonOne;
    this.border = border;
}

52 usages
public JFrame getFrame()
{
    //to call frame
    return this.frame;
}
15 usages
public JLabel getLabelOne()
{
    //to call label
    return this.labelOne;
}
19 usages
public JLabel getLabelTwo()
{
    //to call second label
    return this.labelTwo;
}
```

```
14 usages
public JButton getButtonOne()
{
    //to call button
    return this.buttonOne;
}
1 usage
public Border getBorder()
{
    //to call border
    return this.border;
}
14 usages
public void makeLabel(JLabel label)
{
    //makes the general label attributes
    label.setFont(new Font( name: "Times New Roman",Font.BOLD, size: 30));
    label.setHorizontalAlignment(JLabel.CENTER);
    label.setVerticalAlignment(JLabel.CENTER);
    label.setBackground(Color.WHITE);
    label.setOpaque(true);
    label.setBorder(getBorder());
    getFrame().add(label);
}
2 overrides
@Override
public void actionPerformed(ActionEvent e) {

}
}
```

TicketGUI.java

The TicketGUI.java class is a child class that extends from the superGUI.java class. Its purpose is to show the initial GUI. The imports used for this class are javax.swing.*, javax.swing.border.Border, java.awt.*, and java.awt.event.ActionEvent. This class creates two labels, one button, and a frame using inheritance to avoid code repetition. The action listener creates a new instance of the ParkingLot.java class and makes the ticketGUI invisible.

```java
package Parking;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.awt.event.ActionEvent;
```

```java
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==getButtonOne())
        {
            //if get ticket button is pressed, make parking lot
            getFrame().setVisible(false);
            new ParkingLot();
        }
    }
}
```

```java
public class TicketGUI extends superGUI {
    1 usage
    public TicketGUI(JFrame frame,JLabel labelOne,JLabel labelTwo,JButton buttonOne,Border border)
    {
        super(frame,labelOne,labelTwo,buttonOne,border);
        //make labels for the gui
        getLabelOne().setText("Enter Parking");
        getLabelOne().setBounds( x: 145, y: 0, width: 200, height: 150);
        makeLabel(getLabelOne());

        getLabelTwo().setText("Every Action is 5 Minutes");
        getLabelTwo().setBounds( x: 45, y: 314, width: 400, height: 150);
        makeLabel(getLabelTwo());

        //make the ticket button to open the ParkingGUI
        getButtonOne().setBounds( x: 100, y: 150, width: 285, height: 165);
        getButtonOne().setText("Get Ticket");
        getButtonOne().setFont(new Font( name: "Times New Roman",Font.BOLD, size: 30));
        getButtonOne().setFocusable(false);
        getButtonOne().addActionListener( l: this);
        getFrame().add(getButtonOne());

        //makes the layout of the frame
        getFrame().setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getFrame().setLayout(null);
        getFrame().setResizable(false);
        getFrame().setSize( width: 500, height: 500);
        getFrame().setVisible(true);
        getFrame().getContentPane().setBackground(new Color( r: 200, g: 200, b: 200));
    }
```

ParkingGUI.java

The ParkingGUI.java class creates the frame of the parking lot that includes a level JPanel and a time JPanel. The imports used in this class are javax.swing.*, javax.swing.border.Border and java.awt.*

```java
package Parking;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;


6 usages
public class ParkingGUI extends superGUI{
    //Initializing ParkingGUI Frame and elements
    4 usages
    JPanel levelPanel = new JPanel();
    4 usages
    JPanel timePanel = new JPanel();
```

```java
3 usages
public ParkingGUI(JFrame frame,JLabel labelOne,JLabel labelTwo,JButton buttonOne,Border border)
{
    super(frame,labelOne,labelTwo,buttonOne,border);
    //Make Label that displays the level of the parking Lot
    makeLabel(getLabelOne());
    getLabelOne().setBackground(new Color( r: 200, g: 200, b: 200));
    levelPanel.setLayout(new BorderLayout());
    levelPanel.setBounds( x: 0, y: 0, width: 600, height: 100);
    levelPanel.add(getLabelOne());
    getFrame().add(levelPanel);

    //Make label that displays the Time
    getLabelTwo().setText("12:00");
    makeLabel(getLabelTwo());
    getLabelTwo().setBackground(new Color( r: 200, g: 200, b: 200));
    timePanel.setLayout(new BorderLayout());
    timePanel.setBounds( x: 600, y: 0, width: 190, height: 100);
    timePanel.add(getLabelTwo());
    getFrame().add(timePanel);

    //Make the frame layout
    getFrame().setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getFrame().setSize( width: 800, height: 800);
    getFrame().setLayout(null);
    getFrame().setVisible(false);
    getFrame().setResizable(false);
    }
}
```

ParkingSlotsGUI.java

The ParkingSlotsGUI.java class creates a new GUI that displays the total available parking slots. It has one label to display the information. The imports used in this class are javax.swing.*, javax.swing.border.Border and java.awt.*

```java
package Parking;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;

2 usages
public class ParkingSlotsGUI extends superGUI{
    1 usage
    public ParkingSlotsGUI(JFrame frame, JLabel labelOne, JLabel labelTwo, JButton buttonOne, Border border)
    {
        super(frame,labelOne,labelTwo,buttonOne,border);

        //make label for the Available Slots
        getLabelOne().setBounds( x: 45, y: 150, width: 400, height: 150);
        makeLabel(getLabelOne());

        //create a basic frame
        getFrame().setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getFrame().setLayout(null);
        getFrame().setResizable(false);
        getFrame().setSize( width: 500, height: 500);
        getFrame().setVisible(true);
        getFrame().getContentPane().setBackground(new Color( r: 200, g: 200, b: 200));

    }
}
```

LeaveGUI.java

The LeaveGUI.java class creates the last GUI that allows the user to exit the program. This class makes 7 labels. The first 6 labels in the picture below, are labels to display what level, row and column the user's car is parked on. The last label called payTicket will display "Payment due:".The imports used are javax.swing.*, javax.swing.border.Border, java.awt.* and java.awt.event.ActionEvent.

```
package Parking;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.awt.event.ActionEvent;
2 usages
public class LeaveGUI extends superGUI{
    //Initializes elements and frame of leaveGUI
    3 usages
    JLabel levelLabel = new JLabel();
    3 usages
    JLabel rowLabel = new JLabel();
    3 usages
    JLabel colLabel = new JLabel();
    5 usages
    JLabel levelValue = new JLabel();
    8 usages
    JLabel rowValue = new JLabel();
    8 usages
    JLabel colValue = new JLabel();
    3 usages
    JLabel payTicket = new JLabel();
```

The LeaveGUI uses inheritance for the frame, two labels, and the button. The first label inherited is used to display “Your Car is Parked at”, and the second label is used to display the ticket price. The button inherited is used to exit the program.

```
public LeaveGUI(JFrame frame,JLabel labelOne,JLabel labelTwo,JButton buttonOne,Border border)
{
    super(frame,labelOne,labelTwo,buttonOne,border);
    //Make the labels for where the car is parked
    //and also how much payment is due
    getLabelOne().setText("Your Car is Parked at ");
    makeLabel(getLabelOne());
    getLabelOne().setBounds( x: 0, y: 0, width: 300, height: 200);

    levelLabel.setText("Level");
    makeLabel(levelLabel);
    levelLabel.setBounds( x: 300, y: 0, width: 100, height: 100);

    rowLabel.setText("Row");
    makeLabel(rowLabel);
    rowLabel.setBounds( x: 400, y: 0, width: 100, height: 100);

    colLabel.setText("Col");
    makeLabel(colLabel);
    colLabel.setBounds( x: 500, y: 0, width: 100, height: 100);

    makeLabel(levelValue);
    levelValue.setBounds( x: 300, y: 100, width: 100, height: 100);

    makeLabel(rowValue);
    rowValue.setBounds( x: 400, y: 100, width: 100, height: 100);

    makeLabel(colValue);
    colValue.setBounds( x: 500, y: 100, width: 100, height: 100);
```

```
    payTicket.setText("Payment Due: ");
    makeLabel(payTicket);
    payTicket.setBounds( x: 0, y: 400, width: 300, height: 165);

    makeLabel(getLabelTwo());
    getLabelTwo().setBounds( x: 300, y: 400, width: 300, height: 165);

    //This button will close the program
    getButtonOne().setText("Leave Parking");
    getButtonOne().setFocusable(false);
    getButtonOne().setBounds( x: 100, y: 200, width: 400, height: 200);
    getButtonOne().setFont(new Font( name: "Times New Roman",Font.BOLD, size: 40));
    getButtonOne().addActionListener( l: this);
    getFrame().add(getButtonOne());

    //Frame for leaveGUI
    getFrame().setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getFrame().setLayout(null);
    getFrame().setVisible(false);
    getFrame().setSize( width: 615, height: 600);
    getFrame().getContentPane().setBackground(new Color( r: 200, g: 200, b: 200));
    getFrame().setResizable(false);
}

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==getButtonOne())
    {
        //if button is pressed, close program
        System.exit( status: 0);
    }
```

InterfaceParkingLot.java

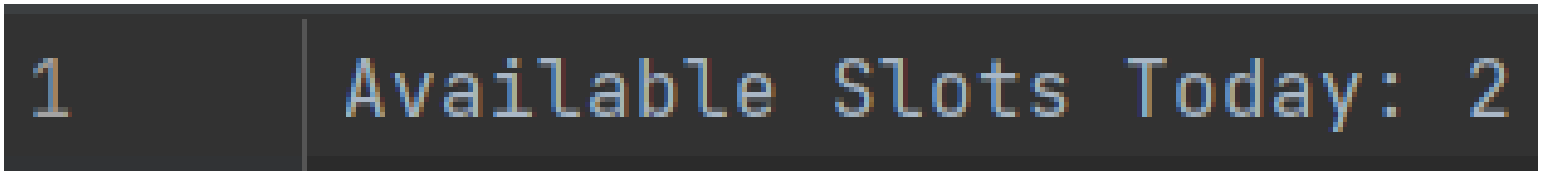
The InterfaceParkingLot.java class is an interface that has a calculatePayment( ) method.

```
package Parking;


1 usage   1 implementation
public interface InterfaceParkingLot {
    //method for calculating payment
    1 usage   1 implementation
    void calculatePayment();
}
```

AvailableParkingSlots.txt

The AvailableParkingSlots.txt is a text file that keeps the information on how many available slots there are in the parking lot today. This information is updated every time the program runs.



ParkingLot.java

The ParkingLot.java class is where all the classes come together to create the Parking Management System. The imports used in this class are javax.swing.*, javax.swing.border.Border, java.io.*, java.util.Random, java.awt.*, java.awt.event.ActionEvent, java.awt.event.ActionListener, and java.io.FileWriter.

```
package Parking;


import javax.swing.*;
import javax.swing.border.Border;
import java.io.*;
import java.util.Random;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
```

In this class, three instances of the ParkingGUI class are created in order to create three levels of the parking system. One instance of the ParkingSlotsGUI class is created to display the total available parking slots. Six buttons were also made for the user to be able to navigate through different parking levels. The Array data structure was used to make the program more compact. In this class there are three JButton arrays that will display the ability to park in a parking slot and one JLabel array that will display the occupied slots.

```
public class ParkingLot implements ActionListener,InterfaceParkingLot {
    //Initializes ParkingLot elements
    11 usages
    ParkingGUI one = new ParkingGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
    12 usages
    ParkingGUI two = new ParkingGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
    10 usages
    ParkingGUI three = new ParkingGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
    1 usage
    ParkingSlotsGUI parkingSlots = new ParkingSlotsGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
    3 usages
    JButton upButtonOne = new JButton();
    3 usages
    JButton upButtonTwo = new JButton();
    2 usages
    JButton upButtonThree = new JButton();
    2 usages
    JButton downButtonOne = new JButton();
    3 usages
    JButton downButtonTwo = new JButton();
    3 usages
    JButton downButtonThree = new JButton();
    3 usages
    JButton[] parkButtonOne = new JButton[8];
    3 usages
    JButton[] parkButtonTwo = new JButton[8];
    3 usages
    JButton[] parkButtonThree = new JButton[8];
    8 usages
    JLabel[] occupiedLabel = new JLabel[8];
```

In this class, the private int x that is declared and the random variable is used to randomize the occupancy of the parking slots. The hour and minute integers are used to display the time. The paymentDue integer is used to calculate the payment ticket price depending on how long the user stays in the parking system. The totalAvailableSlots integer will display the number of available slots throughout the three levels of the parking system. An instance of the LeaveGUI class is created to prepare for the user to exit the system.

```
private int x;
//random class
2 usages
Random random = new Random();
//hour and minute to calculate the time
11 usages
private int hour = 12;
11 usages
private int minute = 0;
//paymentDue to calculate payment
2 usages
private int paymentDue = 0;
//to count today's available slots
4 usages
public int totalAvailableSlots = 0;
18 usages
LeaveGUI leavegui = new LeaveGUI(new JFrame(),new JLabel(),new JLabel(),new JButton(),BorderFactory.createLineBorder(Color.BLACK, thickness: 1));
1 usage
Border border = BorderFactory.createLineBorder(Color.BLACK, thickness: 1);
```

The first few methods of this class are makeParkButton(JButton button), makeOccupiedLabel(JLabel label) and makeButton(JButton button). The first method creates the attributes of all the parking buttons. The second method creates the attributes of all the occupied labels. The third method creates the attributes for the up and down buttons.

```
private void makeParkButton(JButton button)
{
    //This method makes the Parking Slot Button
    button.setFocusable(false);
    button.setFont(new Font( name: "Arial", Font.BOLD,  size: 45));
    button.addActionListener( l: this);
    button.setText("[P]");
}
2 usages
private void makeOccupiedLabel(JLabel label)
{
    //This makes the occupied slot
    label.setFont(new Font( name: "Arial", Font.BOLD,  size: 30));
    label.setText("Occupied");
    label.setBackground(Color.RED);
    label.setBorder(border);
    label.setOpaque(true);
}
2 usages
private void makeButton(JButton button)
{
    //This method makes the common attributes of the button
    button.setFont(new Font( name: "Times New Roman",Font.BOLD, size: 30));
    button.addActionListener( l: this);
    button.setFocusable(false);
}
```

These two methods are makeUpButton(Jbutton button) and makeDownButton(JButton button). These methods are separate so that one button created will say “UP” and the other button will say “DOWN”. In these methods, the makeButton method from previously is used so that both the up and down buttons can obtain the common attributes.

```
private void makeUpButton(JButton button)
{
    //this method makes the "UP" button to change levels
    button.setBounds( x: 600, y: 100, width: 190, height: 332);
    button.setText("UP");
    makeButton(button);
}
3 usages
private void makeDownButton(JButton button)
{
    //this method makes the "DOWN" button to change levels
    button.setBounds( x: 600, y: 432, width: 190, height: 332);
    button.setText("DOWN");
    makeButton(button);
}
```

This method is called makeParkingOccupancy(JFrame frame, JButton[] button). It takes two inputs because this method needs to make the parking occupancy of three floors (frame input) and 8 parking slots (button input). This method creates a 2 x 4 matrix of parking slots. This method is split into two for loops, the first for loop creates the top row of methods and creates an 1 in 6 chance of creating an available parking slot. The second for loop does the same thing, but for the bottom row. Everytime an available slot is created, it is added to the total available slots counter.

```
private void makeParkingOccupancy(JFrame frame,JButton[] button)
{
    // this method makes the parking lot and randomly changes
    // between an available parking slot and an occupied one
    for(int i = 0; i < 4; i++)
    {
        //This makes a 1 in 6 chance for a parking slot to be available
        //This makes the top 4 parking slots
        x = random.nextInt( bound: 6);
        if(x == 1)
        {
            button[i] = new JButton();
            button[i].setBounds( x: i * 150,  y: 100,  width: 150,  height: 332);
            makeParkButton(button[i]);
            frame.add(button[i]);
            totalAvailableSlots++;
        }
        else {
            occupiedLabel[i] = new JLabel();
            occupiedLabel[i].setBounds( x: i*150, y: 100, width: 150, height: 332);
            makeOccupiedLabel(occupiedLabel[i]);
            frame.add(occupiedLabel[i]);
        }
    }
```

```
    for(int i = 4; i < 8; i++) {
        //This makes a 1 in 6 chance for a parking slot to be available
        //This makes the bottom 4 parking slots
        x = random.nextInt( bound: 6);
        if (x == 1) {
            button[i] = new JButton();
            button[i].setBounds( x: (i - 4) * 150,  y: 432,  width: 150,  height: 332);
            makeParkButton(button[i]);
            frame.add(button[i]);
            totalAvailableSlots++;
        } else {
            occupiedLabel[i] = new JLabel();
            occupiedLabel[i].setBounds( x: (i - 4) * 150,  y: 432,  width: 150,  height: 332);
            makeOccupiedLabel(occupiedLabel[i]);
            frame.add(occupiedLabel[i]);
        }
    }
}
```

These are two methods called calculatePayment( ) and setTime( ). The calculatePayment( ) simply adds 15 to the paymentDue counter for every 15 minutes that passes by. The payment due will be displayed in thousand of rupiah.

```
public void calculatePayment()
{
    if((minute == 15) || (minute == 30 || (minute == 45) || (minute == 0)))
    {
        //This adds 15000 to the payment due for every 3 actions performed.
        paymentDue += 15;
    }
}
```

The setTime( ) adds five minutes to the minute counter and everytime the minute count gets to 60, the minute count resets to zero and the hour count gets added by one. This method also displays the time and if the hour gets to 24, the program will immediately close the screen and open the LeaveGUI. At the end of this method, the calculatePayment ( ) is called to calculate the payment.

```
private void setTime()
{
    //This changes the time that is displayed and simultaneously keeps a count
    //of how many minutes has passed.
    minute += 5;
    if(minute == 60)
    {
        //This makes the minute not fo to 61 but instead, it resets the minute
        //and adds 1 to the hour
        hour++;
        minute = 0;
        one.getLabelTwo().setText(hour + ":00");
        two.getLabelTwo().setText(hour + ":00");
        three.getLabelTwo().setText(hour + ":00");
    }
    else if(minute == 5)
    {
        //So that the time will not display 12:5 but 12:05
        one.getLabelTwo().setText(hour + ":05");
        two.getLabelTwo().setText(hour + ":05");
        three.getLabelTwo().setText(hour + ":05");
    }
    else
    {
        //Displays time
        one.getLabelTwo().setText(hour + ":" + minute);
        two.getLabelTwo().setText(hour + ":" + minute);
        three.getLabelTwo().setText(hour + ":" + minute);
    }
    if(hour == 24)
    {
        //IF the hour goes to 24, the parking lot will automatically open the leaveGUI
        leavegui.getFrame().setVisible(true);
    }
    calculatePayment();
}
```

These two methods are called carParked(JFrame frame, JButton button) and updateAvailableSlots(int todaySlots). The first method is used to display the LeaveGUI, the payment due on the LeaveGUI, and close the current GUI. The second method is to update the txt file to show how many available slots there are.

```
private void carParked(JFrame frame,JButton button)
{
    //will close the current parkingGUI and open the leaveGUI
    //also showing the ticket payment price
    frame.setVisible(false);
    leavegui.getLabelTwo().setText(paymentDue + ".000 RP");
    leavegui.getFrame().setVisible(true);
}
1 usage
private void updateAvailableSlots(int todaySlots)
{
    try {
        // Update the file with today's available slots
        FileWriter writer = new FileWriter( fileName: "AvailableParkingSlots.txt");
        writer.write( str: "Available Slots Today: " + todaySlots);
        writer.close();
    } catch (IOException e){
        e.printStackTrace();
    }
}
```

The ParkingLot() combines all the previous elements and methods into one. First, it creates the up and down buttons using the makeUpButton( ) and makeDownButton( ) methods. Then, it adds these buttons to their respective parking levels (frames). Afterwards, each parking level gets different parking labels to differentiate the levels. Then, the makeParkingOccupancy( ) method is applied to each parking level and the first level of the parking system is displayed as the first GUI.

Once all the available parking slots are created, the totalAvailableSlots and both the txt files and ParkingSlotsGUI are updated.

```
ParkingLot(){
    //make up and down buttons
    makeUpButton(upButtonOne);
    makeUpButton(upButtonTwo);
    makeUpButton(upButtonThree);

    makeDownButton(downButtonOne);
    makeDownButton(downButtonTwo);
    makeDownButton(downButtonThree);

    //add up and down buttons to parkingGUIs
    one.getFrame().add(upButtonOne);
    one.getFrame().add(downButtonOne);
    two.getFrame().add(upButtonTwo);
    two.getFrame().add(downButtonTwo);
    three.getFrame().add(upButtonThree);
    three.getFrame().add(downButtonThree);

    //set level labels
    one.getLabelOne().setText("Level: 1");
    two.getLabelOne().setText("Level: 2");
    three.getLabelOne().setText("Level: 3");
```

```
    //create parking occupancies
    makeParkingOccupancy(one.getFrame(),parkButtonOne);
    makeParkingOccupancy(two.getFrame(),parkButtonTwo);
    makeParkingOccupancy(three.getFrame(),parkButtonThree);

    //the parking lot starts at level 1
    one.getFrame().setVisible(true);

    //Updates txt file and also adds the total slots today to the Parking Slots GUI
    updateAvailableSlots(totalAvailableSlots);
    parkingSlots.getLabelOne().setText("Available Slots Today: " + totalAvailableSlots);
}
```

This is where all the button actions occur. These first four create actions for the up and down buttons in the different parking levels, so that users are able to navigate through them easily.

```
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==upButtonOne)
    {
        //This allows the user to go up a level from level 1
        setTime();
        one.getFrame().setVisible(false);
        two.getFrame().setVisible(true);
    }
    if(e.getSource()==upButtonTwo)
    {
        //This allows the user to go up a level from level 2
        setTime();
        two.getFrame().setVisible(false);
        three.getFrame().setVisible(true);
    }
    if(e.getSource()==downButtonTwo)
    {
        //This allows the user to go down a level from level 2
        setTime();
        two.getFrame().setVisible(false);
        one.getFrame().setVisible(true);
    }
    if(e.getSource()==downButtonThree)
    {
        //This allows the user to go down a level from level 3
        setTime();
        three.getFrame().setVisible(false);
        two.getFrame().setVisible(true);
    }
```

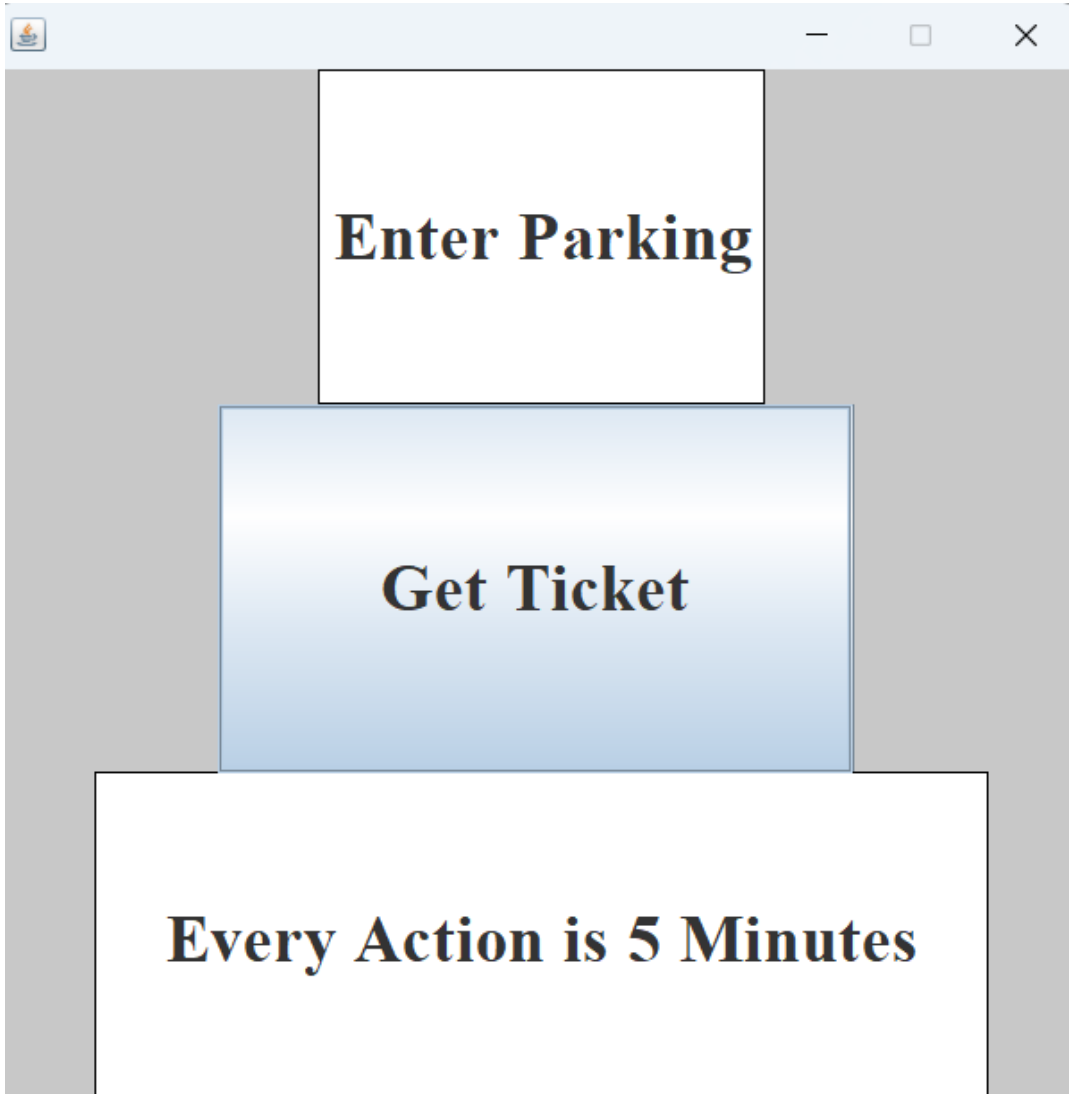
There are three for loops that are used to determine and display what level, row, and column that the user parked on. The first for loop is for when the user parks on the first level, the second for loop is for when the user parks on the second level, and the third for loop is for when the user parks on the third level.

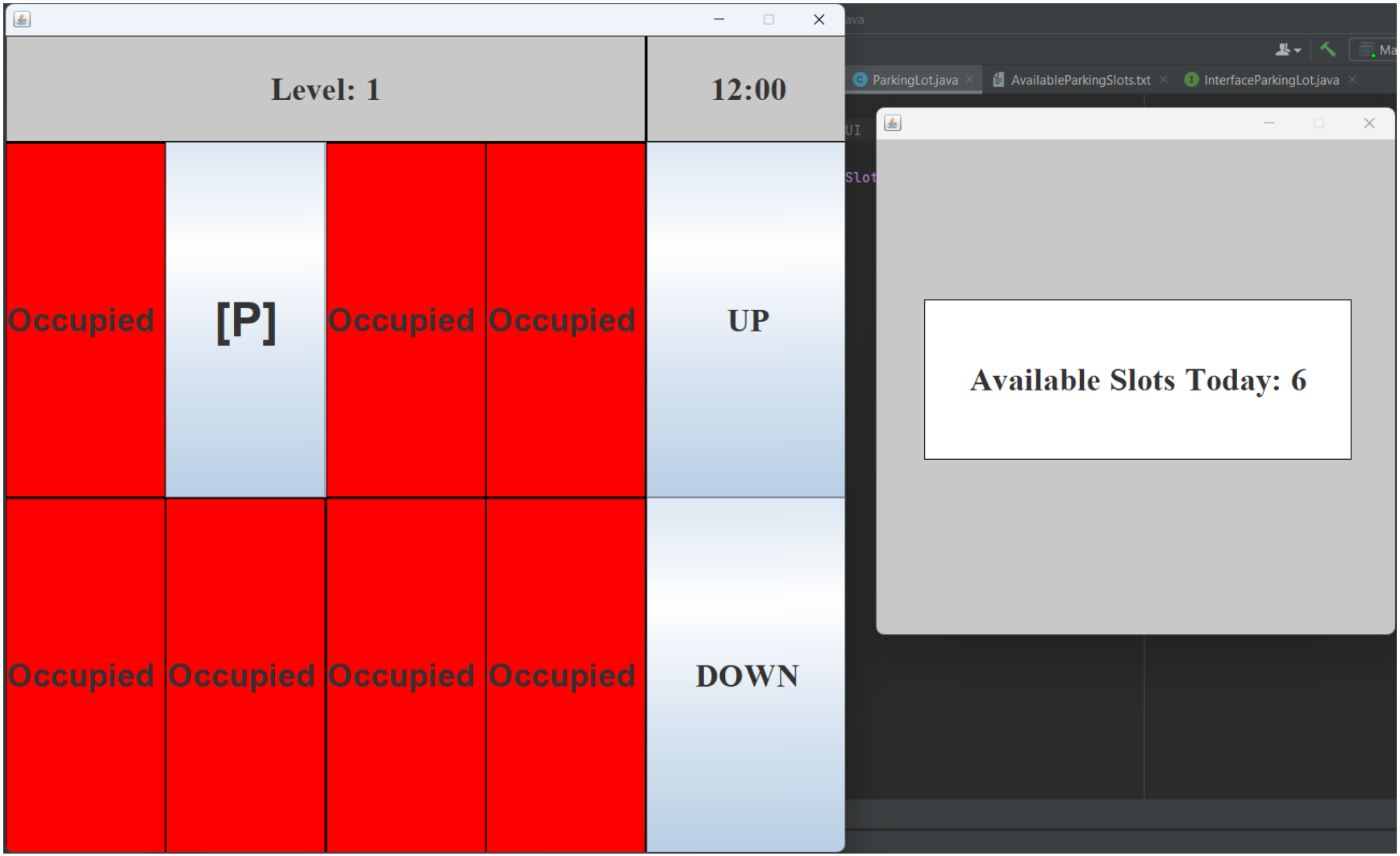
```
for(int i = 0; i < 8; i++)
{
    //checks which parking button is pressed
    //and displays on the leaveGUI what level, row, and column
    //that the car is parked at
    if(e.getSource() == parkButtonOne[i])
    {
        carParked(one.getFrame(),parkButtonOne[i]);
        if(i < 4)
        {
            leavegui.rowValue.setText("1");
            leavegui.colValue.setText(Integer.toString( i: i+1));
        }
        else {
            leavegui.rowValue.setText("2");
            leavegui.colValue.setText(Integer.toString( i: i-3));
        }
        leavegui.levelValue.setText("1");
    }
}
```

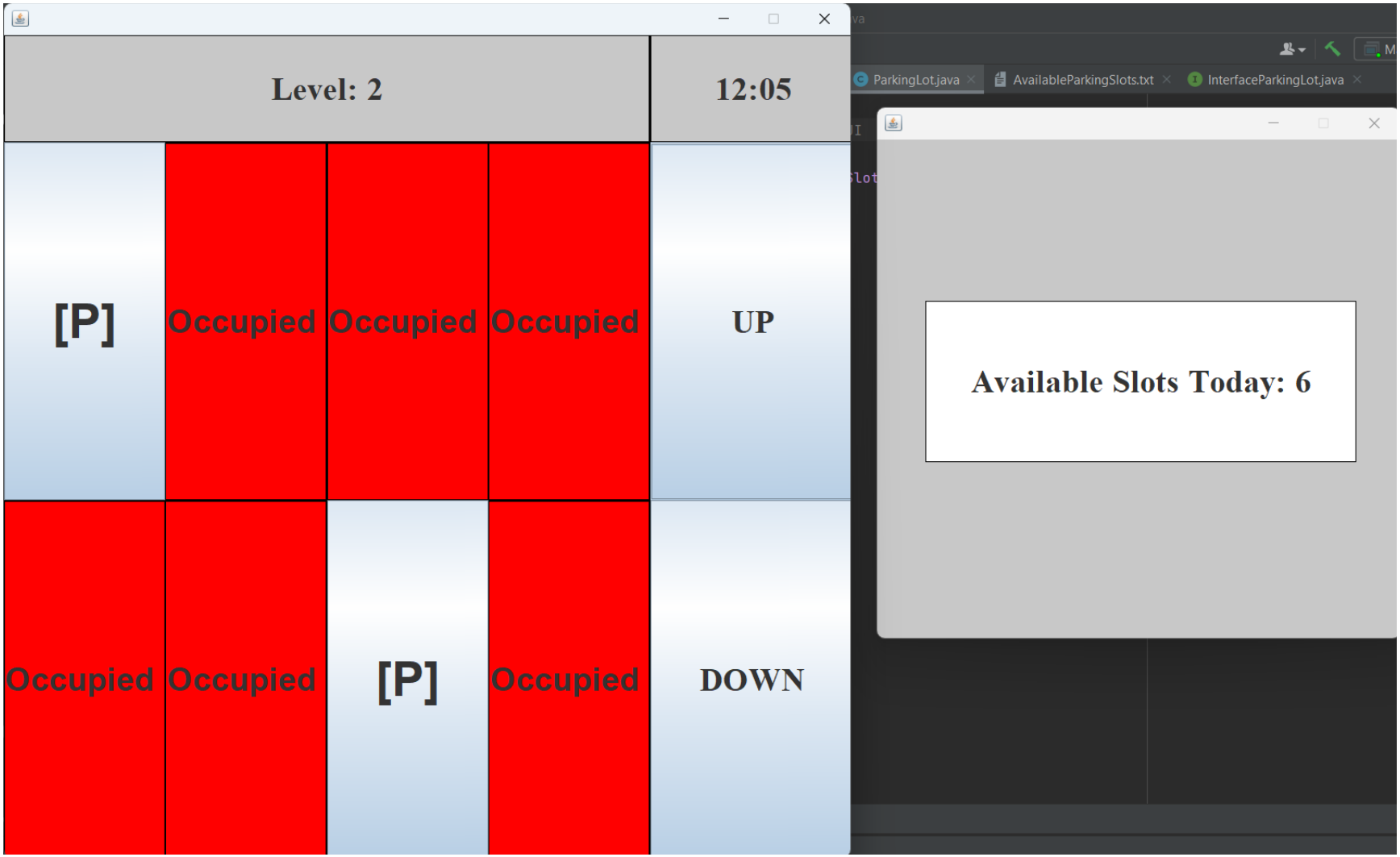
```
for(int i = 0; i < 8; i++)
{
    //checks which parking button is pressed
    //and displays on the leaveGUI what level, row, and column
    //that the car is parked at
    if(e.getSource()== parkButtonTwo[i])
    {
        carParked(two.getFrame(),parkButtonTwo[i]);
        if(i < 4)
        {
            leavegui.rowValue.setText("1");
            leavegui.colValue.setText(Integer.toString( i: i+1));
        }
        else {
            leavegui.rowValue.setText("2");
            leavegui.colValue.setText(Integer.toString( i: i-3));
        }
        leavegui.levelValue.setText("2");
    }
}
```

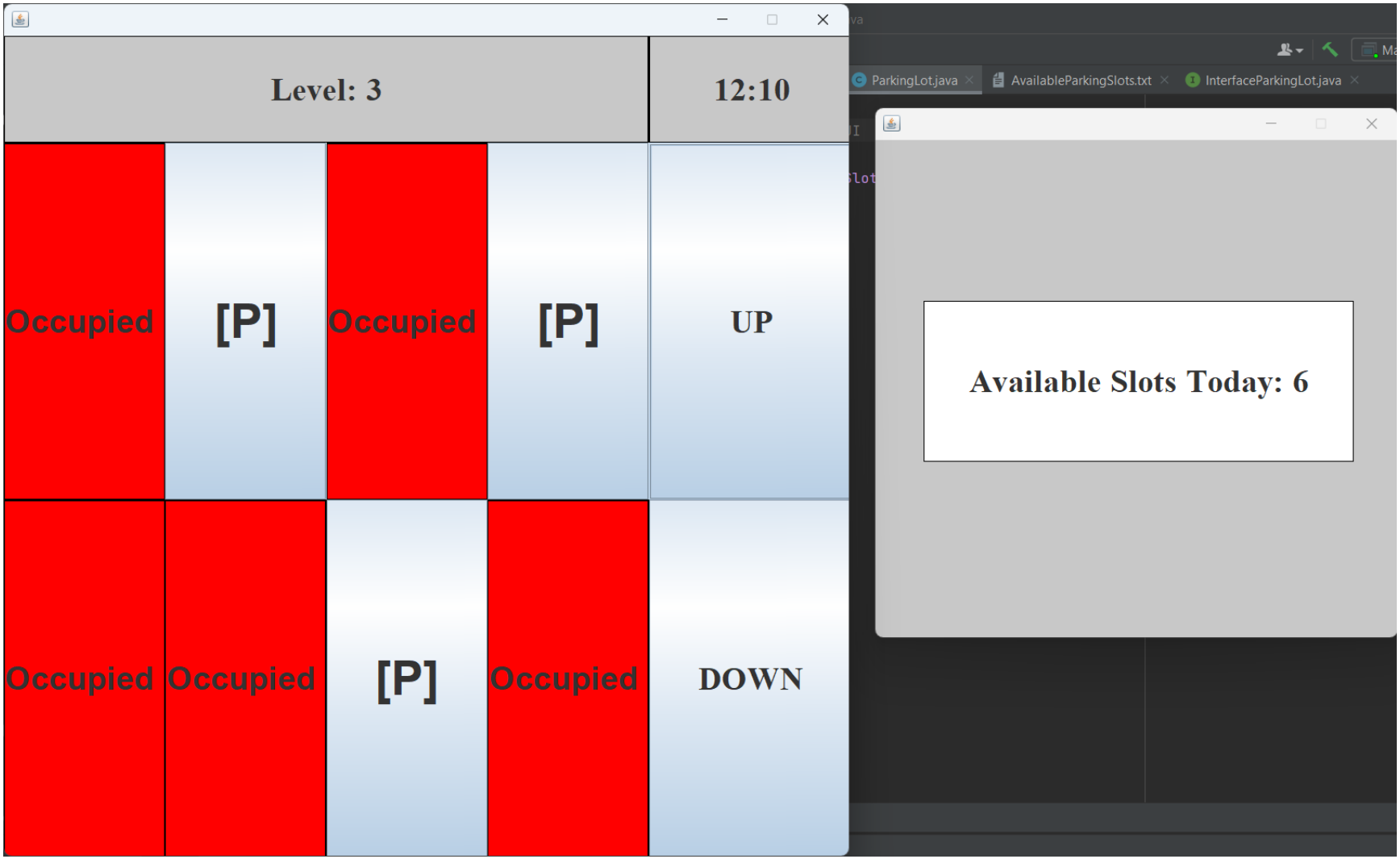
```
        for(int i = 0; i < 8; i++)
        {
            //checks which parking button is pressed
            //and displays on the leaveGUI what level, row, and column
            //that the car is parked at
            if(e.getSource()== parkButtonThree[i])
            {
                carParked(three.getFrame(),parkButtonThree[i]);
                if(i <= 4)
                {
                    leavegui.rowValue.setText("1");
                    leavegui.colValue.setText(Integer.toString( i: i+1));
                }
                else {
                    leavegui.rowValue.setText("2");
                    leavegui.colValue.setText(Integer.toString( i: i-3));
                }
                leavegui.levelValue.setText("3");
            }
        }
    }
}
```

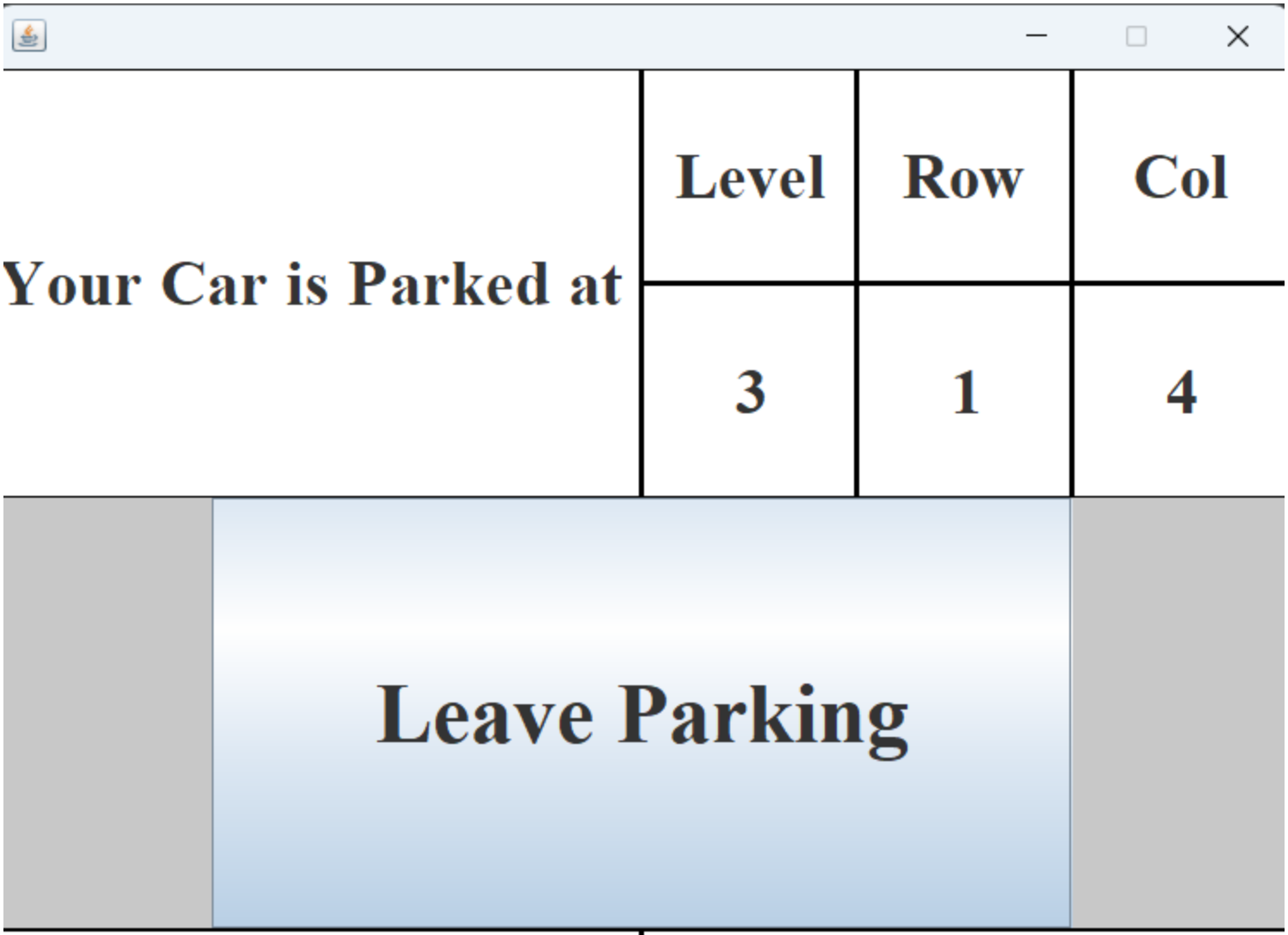
Evidence of Working Program




Level: 1
12:00
Occupied
[P]
Occupied
Occupied
UP
Occupied
Occupied
Occupied
Occupied
DOWN
ParkingLot.java
AvailableParkingSlots.txt
InterfaceParkingLot.java
Available Slots Today: 6


Level: 2
12:05
[P]
Occupied
Occupied
Occupied
UP
Occupied
Occupied
[P]
Occupied
DOWN
ParkingLot.java
AvailableParkingSlots.txt
InterfaceParkingLot.java
Available Slots Today: 6


Level: 3
12:10
Occupied
[P]
Occupied
[P]
UP
Occupied
Occupied
[P]
Occupied
DOWN
ParkingLot.java
AvailableParkingSlots.txt
InterfaceParkingLot.java
Available Slots Today: 6


Level
Row
Col
Your Car is Parked at
3
1
4
Leave Parking
Payment Due:
0.000 RP