UNIVERSITI KEBANGSAAN MALAYSIA
The National University of Malaysia

UKM Parking Program

By Shameer Ali – A173586
Group 13

use the pdf version if the word file seem not organized

Introduction

Since UKM have many parking spaces and also each year new student brings in new vehicle which makes the parking in the whole UKM not organised. So students and staffs don't get to park their vehicle in some places due to this. And the staffs can't keep track of parking's in UKM.

This program is intended to keep track of vehicle and parking in the UKM. Students and staff can use this to select and reserve parking. They can also see if a parking is full or not so that they can quickly reserve a spot in the parking before going there or go to another parking if it's full.

How To Use the Software,

GUI Design,
&

File I/O Format

Mohamed Shameer Ali bin A.M Abdul Rahman

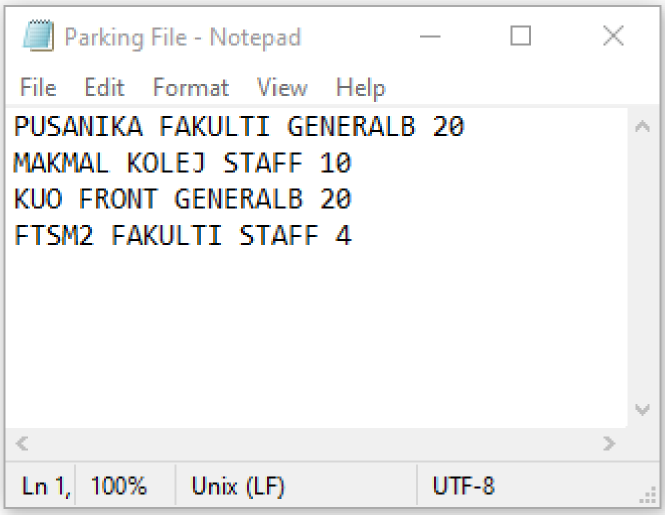
A173586

~run the UKMParking.Java file

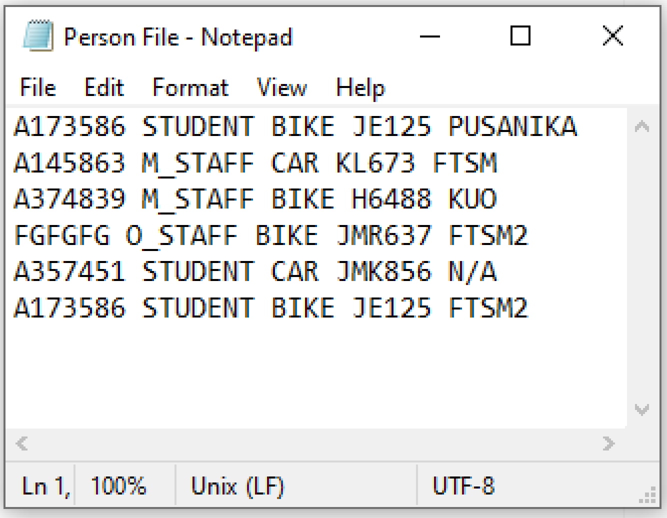
The GUI you see in here won't be 100% the same as the final program. It will have slight colour change and buttons design that's all

Contents

Files to Store Data



```
PUSANIKA FAKULTI GENERALB 20
MAKMAL KOLEJ STAFF 10
KUO FRONT GENERALB 20
FTSM2 FAKULTI STAFF 4
```



```
A173586 STUDENT BIKE JE125 PUSANIKA
A145863 M_STAFF CAR KL673 FTSM
A374839 M_STAFF BIKE H6488 KUO
FGFGFG O_STAFF BIKE JMR637 FTSM2
A357451 STUDENT CAR JMK856 N/A
A173586 STUDENT BIKE JE125 FTSM2
```

1. If you choose to start from a list of data that you have you can follow this step or else you can start a new file from page 4 at "Start New" section.

2. The image above is how the data need to be stored in the files.

   The person file need to be written in :

   Person_ID < > Person_Job < > Vehicle_Type < > Vehicle_Plat_No < > Parked_At
   Person_ID : ID of the person.
   Person_Job: Either "STUDENT", "0_STAFF" (stands for ordinary staff), "M_STAFF" **(**stand for management staff).
   Vehicle_Type: Either "Car", "BIKE", "LORRY", "BUS".
   Vehicle_Plat_No: The Plat Numbur of that vehicle.
   Parked_At: The name of the parking area the vehicle is parked or else write "N/A" if none.

   The parking file need to be written in :

   Parking_name < > Parking_Area < > Parking_Type < > Parking_space
   Parking_name : The name of the parking. Can be what ever you want. Person file will refer this. You don't need to make parkings for all the person in person file. You can do that in the program later if you want.
   Parking_Area : The area at UKM. Eg: KOLEJ, FAKULTI, PUBLIC.
   Parking Type: Either "SATFF", "GENERALB", "GENERALC", "MANAGEMENT".
   Parking_Space: The amount of space in that parking lot.
   Each word is separated by a <space>.

3. Name the file anything you want and just store the file anywhere in you computer.

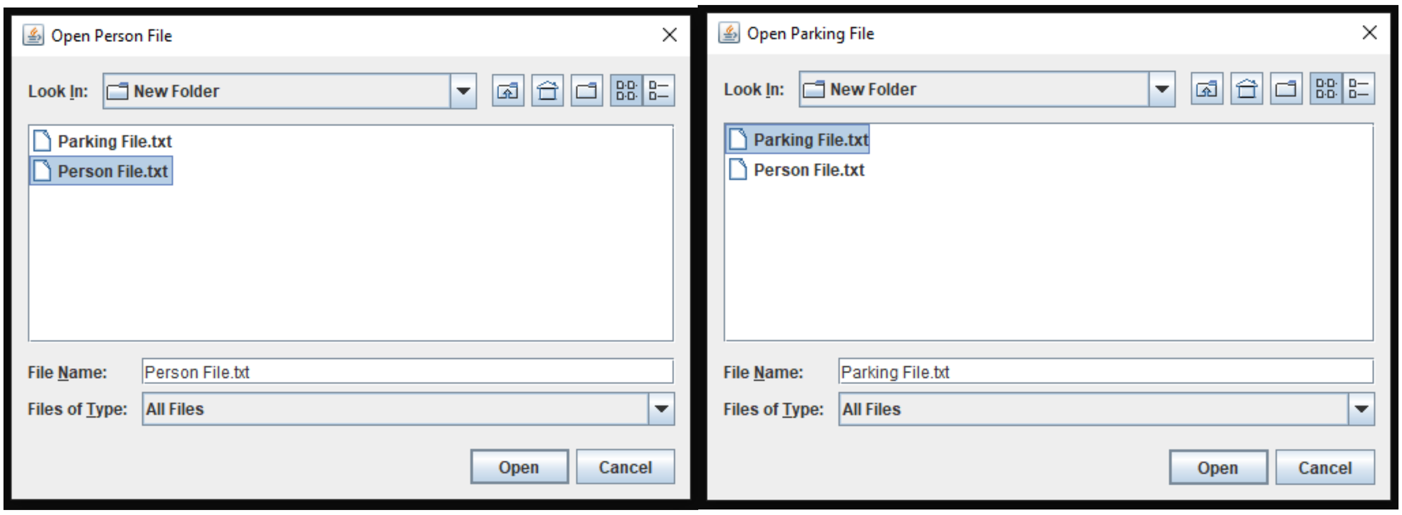
Starting The Program

Select From File

1. Start the program and click “Select From File” Button if you have a file to open.

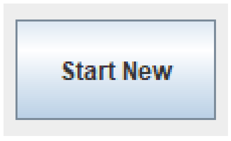
2. See the tittle on the File Chooser pop up.

3. Based on the image below select the file stored the person information on the “Open Person File” File_Chooser and click “Open”. Select the file stored parking information on the “Open Parking File” File_Chooser and click “Open”.



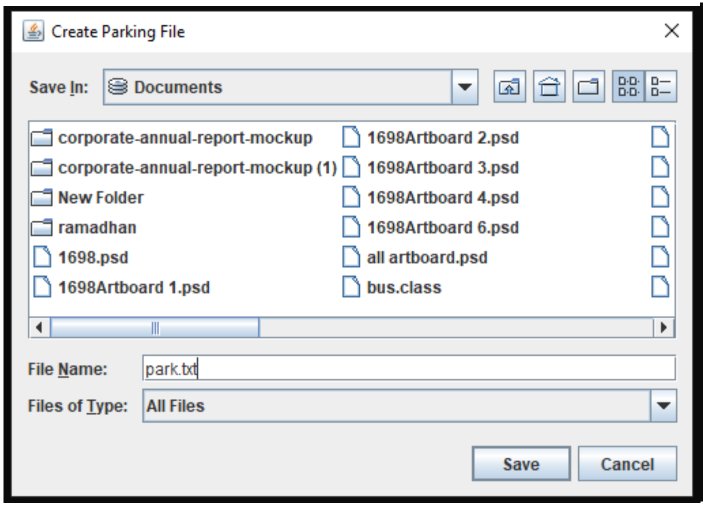
Start New

1. If you don’t have any file to choose from, then juts click the “Start New” Button.

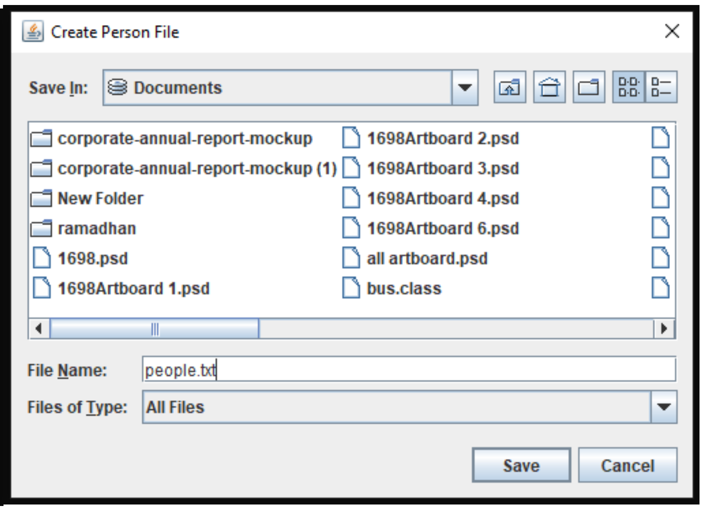


2. See the tittle on the pop up.

3. Based on the image below, type in the name of the person file you want to make at the “Create Person File” pop up and click “Save” to make a new file to store person informations. Then Type in the name of the parking file you want to make at the “Create Parking File pop up and click “Save” to make a new file to store parking informations.

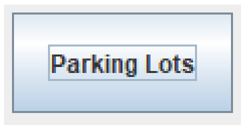


File Name: park.txt
Files of Type: All Files

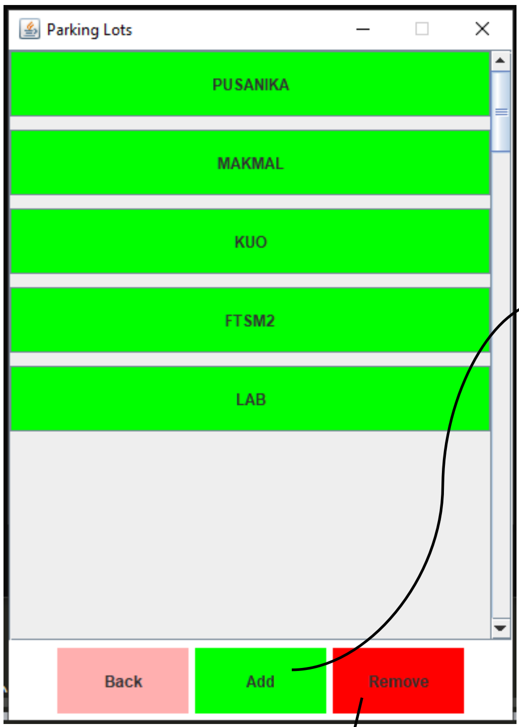


File Name: people.txt
Files of Type: All Files

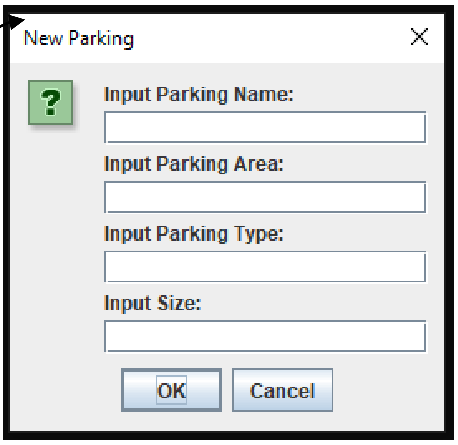
Parking Lots



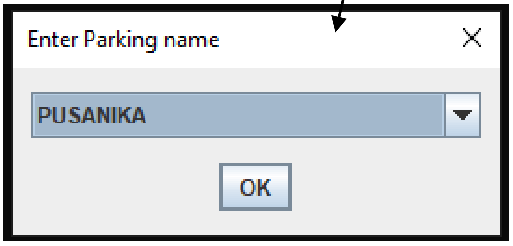
The “Parking Lots” Button will bring you to the list of parking names. The parking names is in button so you can click it and see who’s in that parking lot.



Based on the image on the left, if you use “Select From File” you Gui will look something like that. All the parking you created is a button that you can click. But if you use “Start New” you will still have the same Gui but without the parking names buttons.

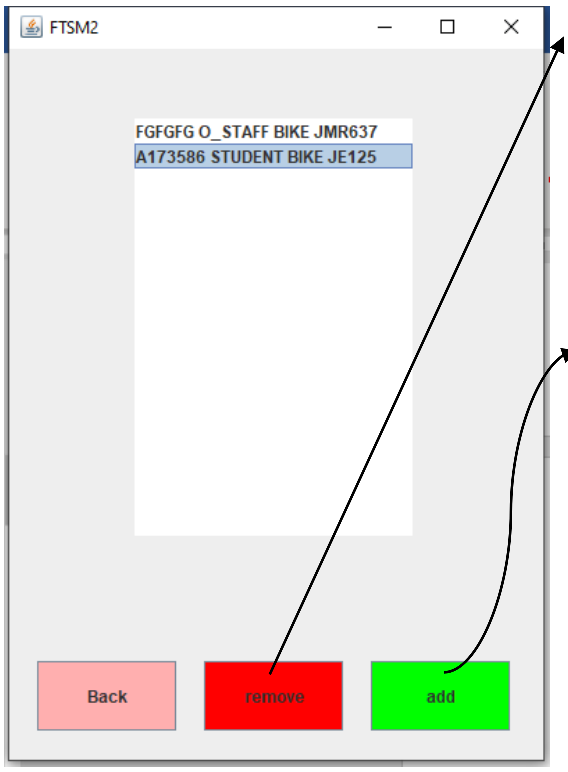


If you click “add” a pop up will appear where you can make a new parking space. Refer to page 2 at the “Parking files need to be written in:“ section for guidance to fill these up. The parking space you created will search the person file to see if there is anyone at the new parking you have created. That’s why you don’t need to make all the parking in the parking file and can just do it in the Gui.



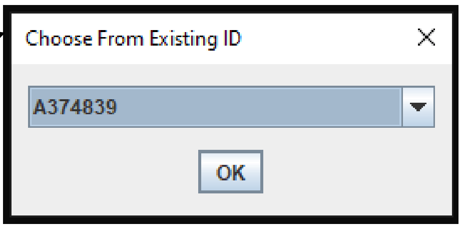
By clicking “remove” you can select the parking space you want to remove from your data from the drop down menu.

Parked List


FTSM2

FGFGFG O_STAFF BIKE JMR637
A173586 STUDENT BIKE JE125

By clicking one of the parking spaces, it will open a frame containing the list of people in that parking lot. To edit the data, you can click on the data in the list like I clicked the A173586 person. Then you can choose "remove" to clear that person from the parking lot. Note that once you click remove the person Parked_At data will be set to "N/A" and be removed from that current list.

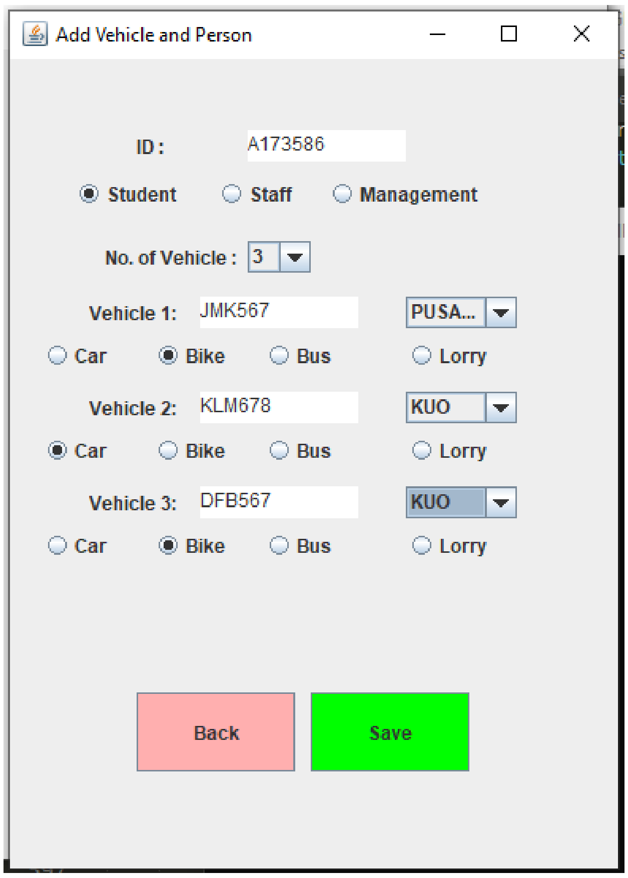

Choose From Existing ID

A374839

If you choose add, a new pop up will appear where you can choose a person to add to that parking lot. The pop up have a drop down menu containing the ID of person that have been created or from the list.

Create New

1. This is where you can create new id, vehicle and assign parking space to that person.


Add Vehicle and Person

ID : A173586

[✓ Student] Staff Management

No. of Vehicle : 3

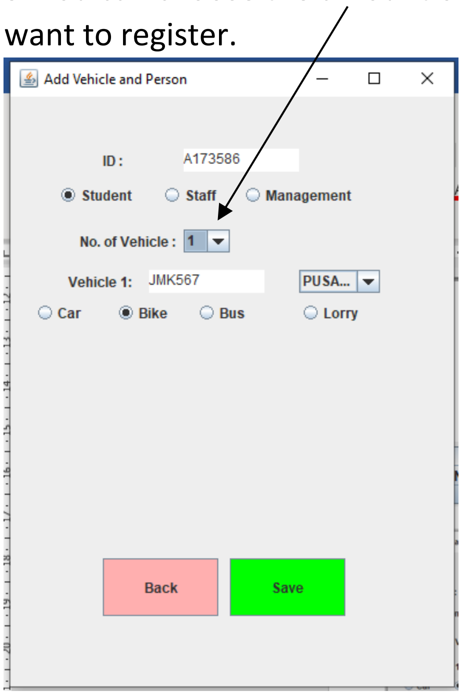
Vehicle 1: JMK567 PUSA...
Car [✓ Bike] Bus Lorry

Vehicle 2: KLM678 KUO
[✓ Car] Bike Bus Lorry

Vehicle 3: DFB567 KUO
Car [✓ Bike] Bus Lorry

2. You need to enter the ID and select the job so that the parking spaces available for that job can be selected.

3. You can choose the amount of vehicle that the ID want to register.


Add Vehicle and Person

ID : A173586

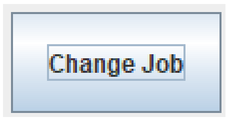
[✓ Student] Staff Management

No. of Vehicle : 1

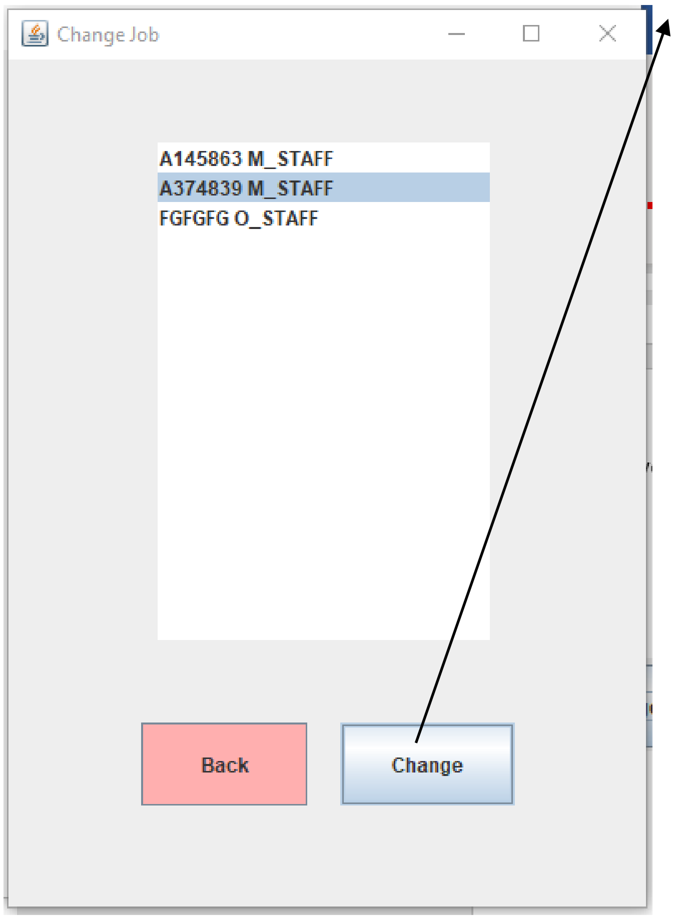
Vehicle 1: JMK567 PUSA...
Car [✓ Bike] Bus Lorry

4. After you have done. Just click save and your new data will be saved.

Change Job



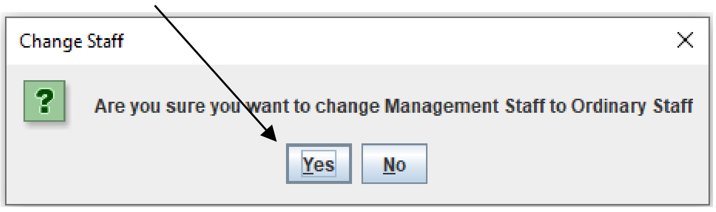
Here you can change appoint management staff or end management staff.



1. Just click the person you want to change their job status and then click change.

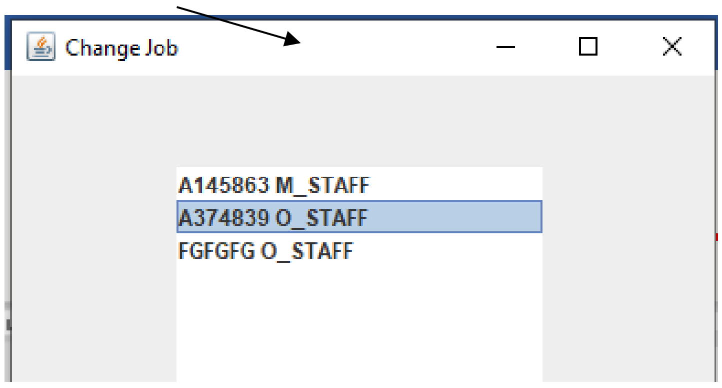
2. A pop up will appear asking you if you wish to change their job from "…" to "…".

3. Click "Yes" if you wish to continue.



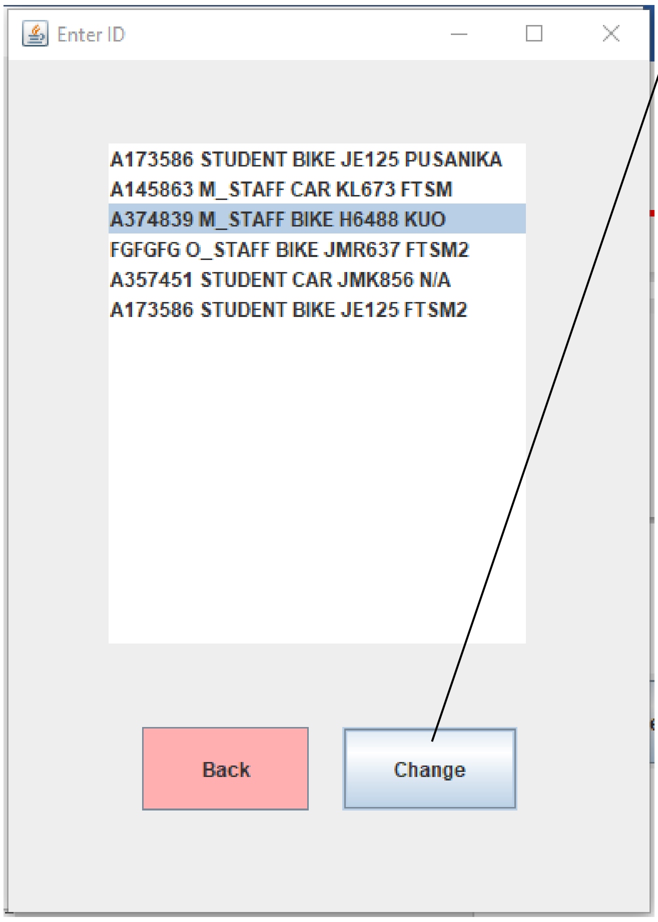
Are you sure you want to change Management Staff to Ordinary Staff

3. It will refresh and show you the change.



Change Vehicle

This is where you can change the vehicle owner that has been assigned before to a new owner.

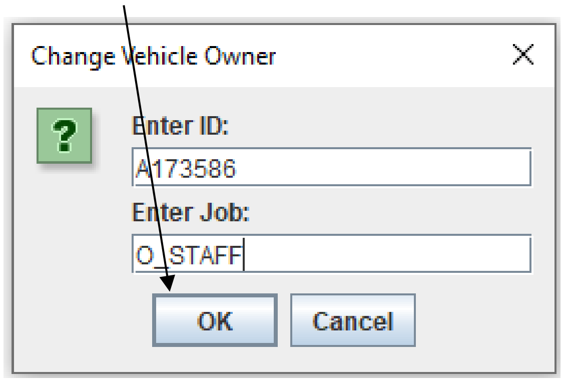


A173586 STUDENT BIKE JE125 PUSANIKA
A145863 M_STAFF CAR KL673 FTSM
A374839 M_STAFF BIKE H6488 KUO
FGFGFG O_STAFF BIKE JMR637 FTSM2
A357451 STUDENT CAR JMK856 N/A
A173586 STUDENT BIKE JE125 FTSM2

1. Just click the vehicle you want to change the owner of and then click change.

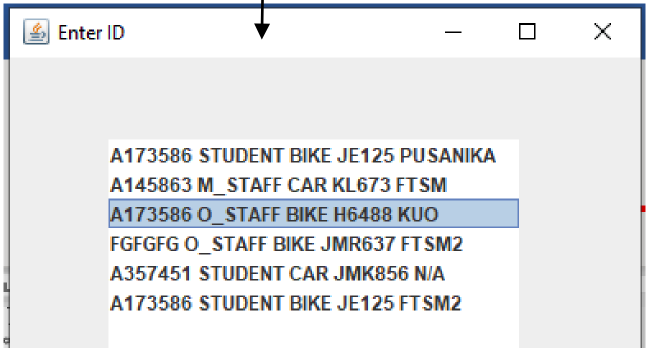
2. A pop up will appear asking you the new owner ID and their job. The job should only be either “STUDENT”, “O_STAFF” or “M_STAFF”.

3. Click ok after you have filled up.



Enter ID:
A173586
Enter Job:
O_STAFF

4. The list will refresh showing the new assigned owner.



A173586 STUDENT BIKE JE125 PUSANIKA
A145863 M_STAFF CAR KL673 FTSM
A173586 O_STAFF BIKE H6488 KUO
FGFGFG O_STAFF BIKE JMR637 FTSM2
A357451 STUDENT CAR JMK856 N/A
A173586 STUDENT BIKE JE125 FTSM2

Features of the program

1. The program will save and write the data in the file anytime the user alter or add any data.
2. The program will automatically turn on the user “Caps Lock” on their keyboard when they run the program. They can see the “Caps lock” light light-up. The user can just close the caps lock if they want and the program will work just fine.

Flaw and Constraints

1. Maybe I could have made the coding shorter by using the full feature of all the classes.

2. The textfield in the optionpane could be turned into radiobutton to make it easier for users to input data.

3. The frame cannot be resized due to the null layout of the frame that will make the component messed up when resized.

4. Forgot to add JScrollPane in the JList to make the JList scroll able. Now the Jlist can only display data inside it's size.

Advantage and Feature

1. Users can create as many parking spaces they want.

2. Any alteration of data will write the file automatically.

~If there's any problem, please contact me at "01140448922" or email me at a173586@siswa.ukm.edu.my . As far as I code & tested, this program is complete and has all the functions needed and also efficient~

UNIVERSITI
KEBANGSAAN
MALAYSIA
The National University of Malaysia

List of Class & Explaination

Mohamed Shameer Ali Bin A.M Abdul Rahman

A173586

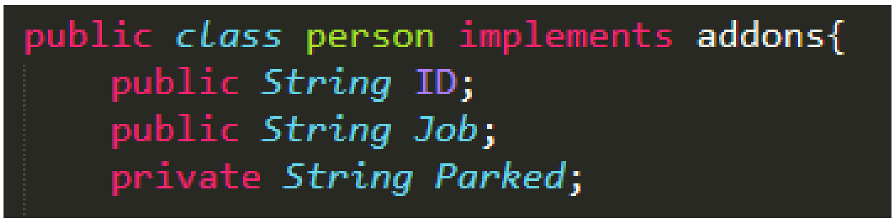
`Run the "UKMParking.java" to use the program`

use the pdf version if the word file seem not organized

Contents

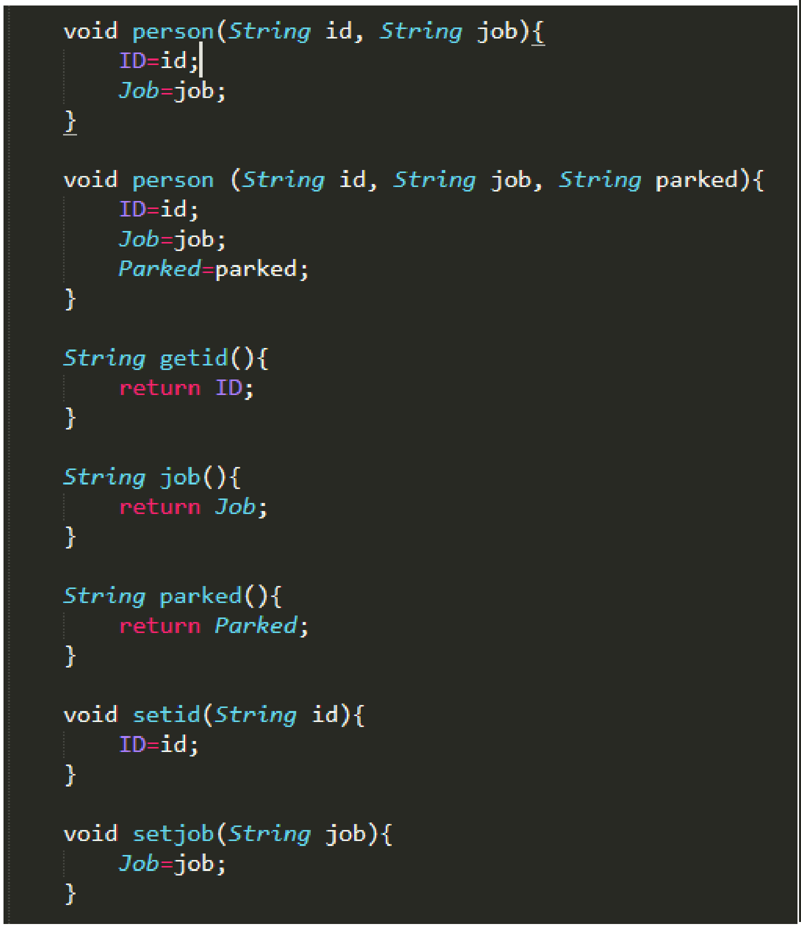
Class: person,

Class: staff, Class: students

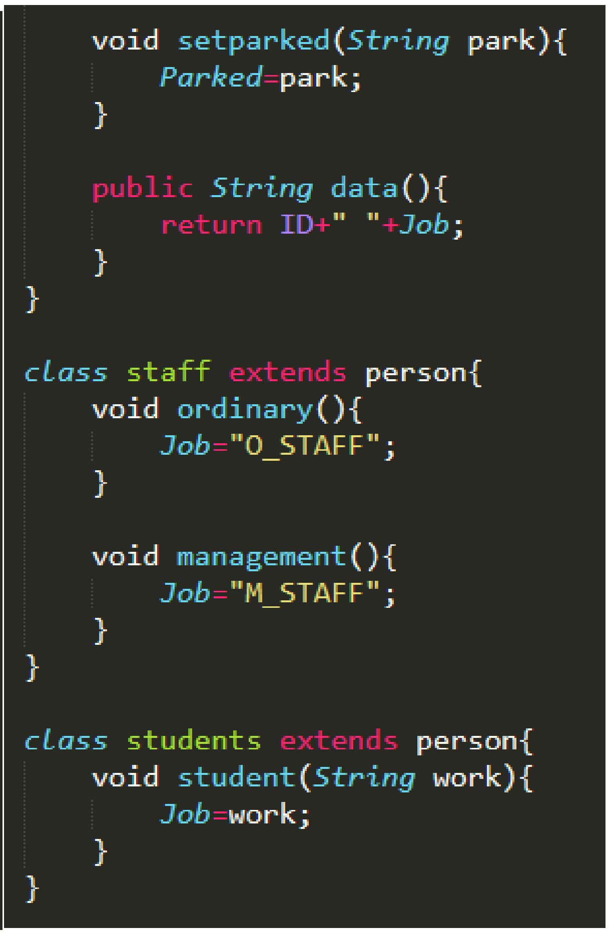


```java
public class person implements addons{
    public String ID;
    public String Job;
    private String Parked;
```

This is the person class where it contains the information of every person who has their data. It implements addons interface.



```java
    void person(String id, String job){
        ID=id;
        Job=job;
    }

    void person (String id, String job, String parked){
        ID=id;
        Job=job;
        Parked=parked;
    }

    String getid(){
        return ID;
    }

    String job(){
        return Job;
    }

    String parked(){
        return Parked;
    }

    void setid(String id){
        ID=id;
    }

    void setjob(String job){
        Job=job;
    }
```



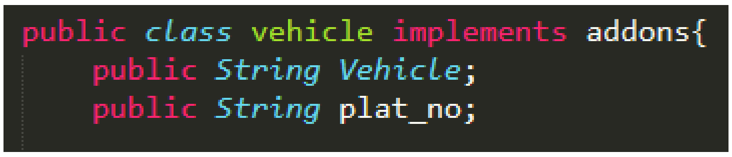
```java
    void setparked(String park){
        Parked=park;
    }

    public String data(){
        return ID+" "+Job;
    }
}

class staff extends person{
    void ordinary(){
        Job="O_STAFF";
    }

    void management(){
        Job="M_STAFF";
    }
}

class students extends person{
    void student(String work){
        Job=work;
    }
}
```

This person has 2 constructors, person(id, job) & person(id, job, parked) with different parameters each. And it also contains set setid(), setjob(), setparked() and get getid(), job(), parked() methods if the data needed to be altered or transferred.

The staff and students class is a child class of person class. It inherits all attributed from person class. Staff class store the information for staffs and students class is the same for student information. The ordinary(), management() & student() methods in the two classes are used to change their job when needed.

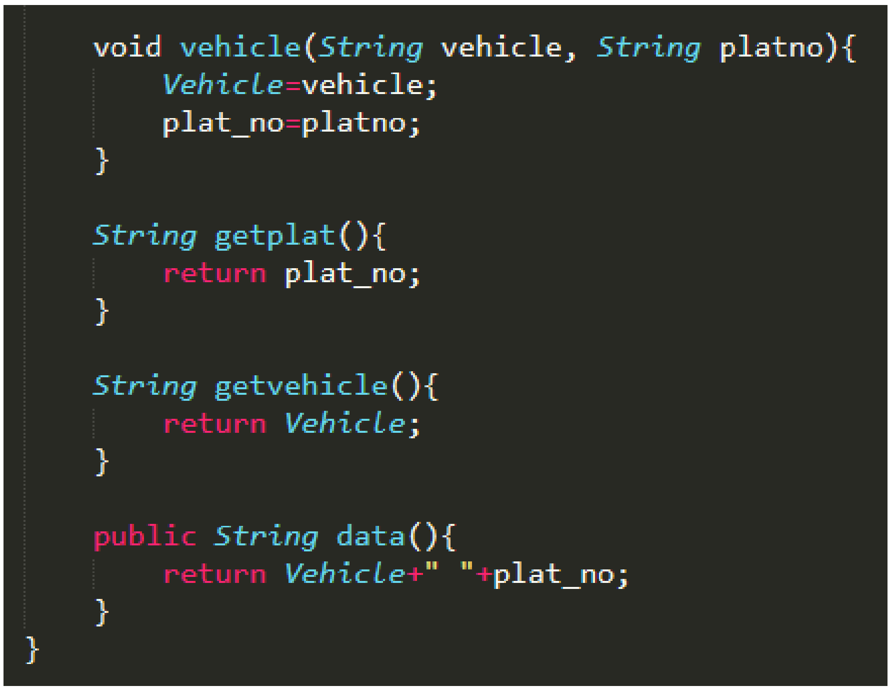
Class: vehicle,

Class: car, Class: motorcycle, Class: bus, Class: lorry

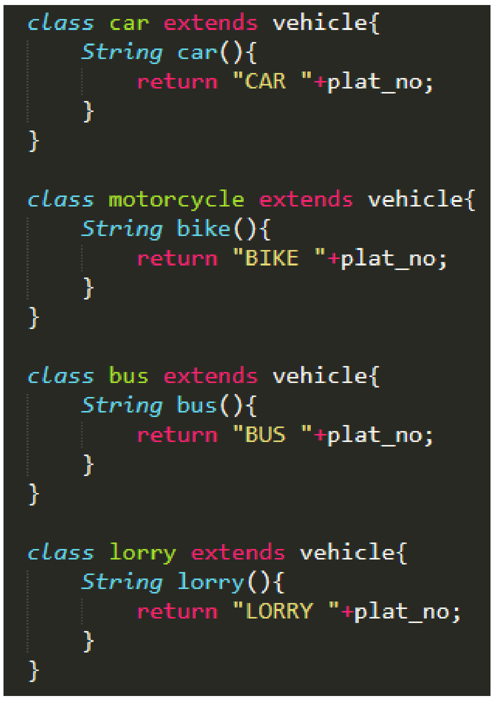


```
public class vehicle implements addons{
    public String Vehicle;
    public String plat_no;
```

This is the vehicle class where it contains the information of every vehicle and it’s data. It implements addons interface.



```
    void vehicle(String vehicle, String platno){
        Vehicle=vehicle;
        plat_no=platno;
    }

    String getplat(){
        return plat_no;
    }

    String getvehicle(){
        return Vehicle;
    }

    public String data(){
        return Vehicle+" "+plat_no;
    }
}
```

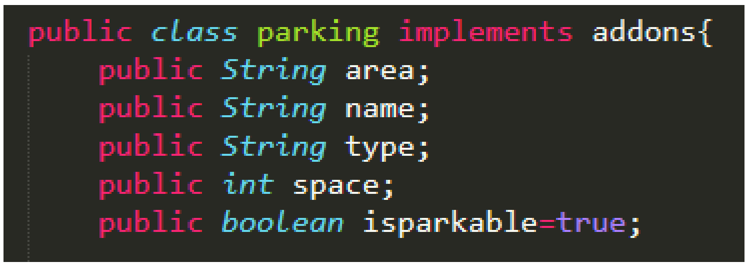


```
class car extends vehicle{
    String car(){
        return "CAR "+plat_no;
    }
}

class motorcycle extends vehicle{
    String bike(){
        return "BIKE "+plat_no;
    }
}

class bus extends vehicle{
    String bus(){
        return "BUS "+plat_no;
    }
}

class lorry extends vehicle{
    String lorry(){
        return "LORRY "+plat_no;
    }
}
```

The vehicle class has 1 constructor vehicle() with 2 parameter. The class has its get getplat(), getvehicle(), data() methods and if any data need to be altered it can be done through the constructor.

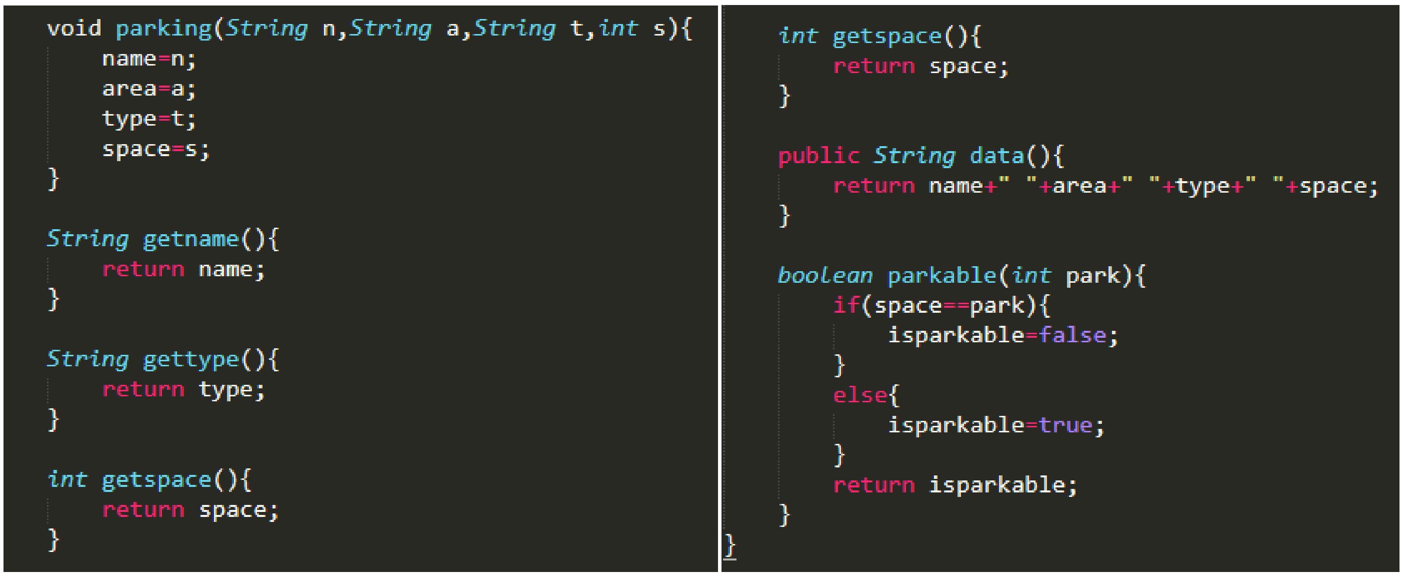
The other four classes car, motorcycle, bus and lorry is a child class of vehicle and is used to stored their own vehicle if any user owns it. Each classes have their own method car(), bike(), bus(), lorry() that is used to get data.

Class: parking



```
public class parking implements addons{
    public String area;
    public String name;
    public String type;
    public int space;
    public boolean isparkable=true;
```

This class handles all the information regarding parking area. It stores the data of parking spaces. It implements addons interface.



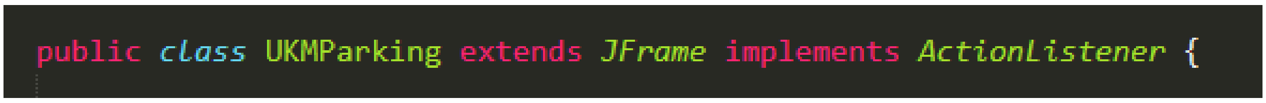
```
    void parking(String n,String a,String t,int s){
        name=n;
        area=a;
        type=t;
        space=s;
    }

    String getname(){
        return name;
    }

    String gettype(){
        return type;
    }

    int getspace(){
        return space;
    }
```

```
    int getspace(){
        return space;
    }

    public String data(){
        return name+" "+area+" "+type+" "+space;
    }

    boolean parkable(int park){
        if(space==park){
            isparkable=false;
        }
        else{
            isparkable=true;
        }
        return isparkable;
    }
}
```

The parking class has one constructor parking() with four parameter. It has its own get getname(), gettype(), getspace(), data() methods to transfer data.

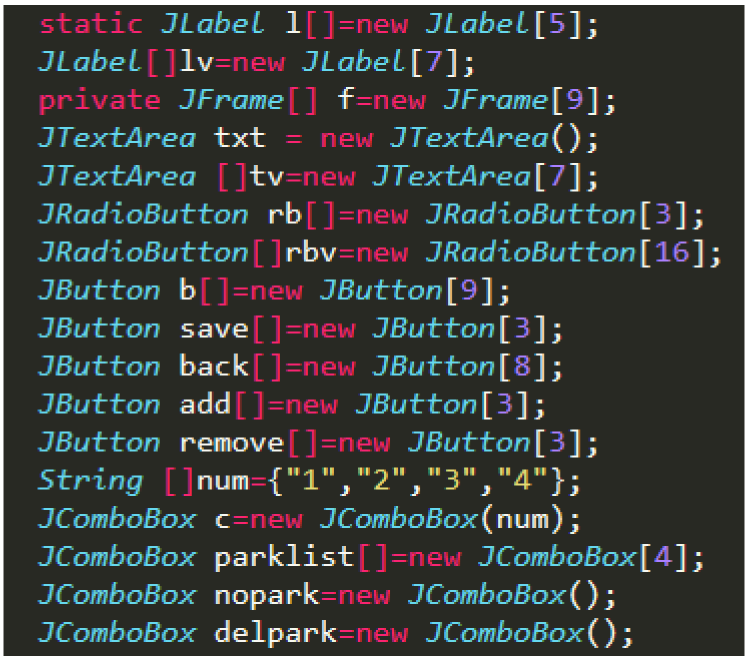
It also has a parkable parkable() methods that keeps record if a specific parking is full or not and if it can be parked.

Class : UKMParking



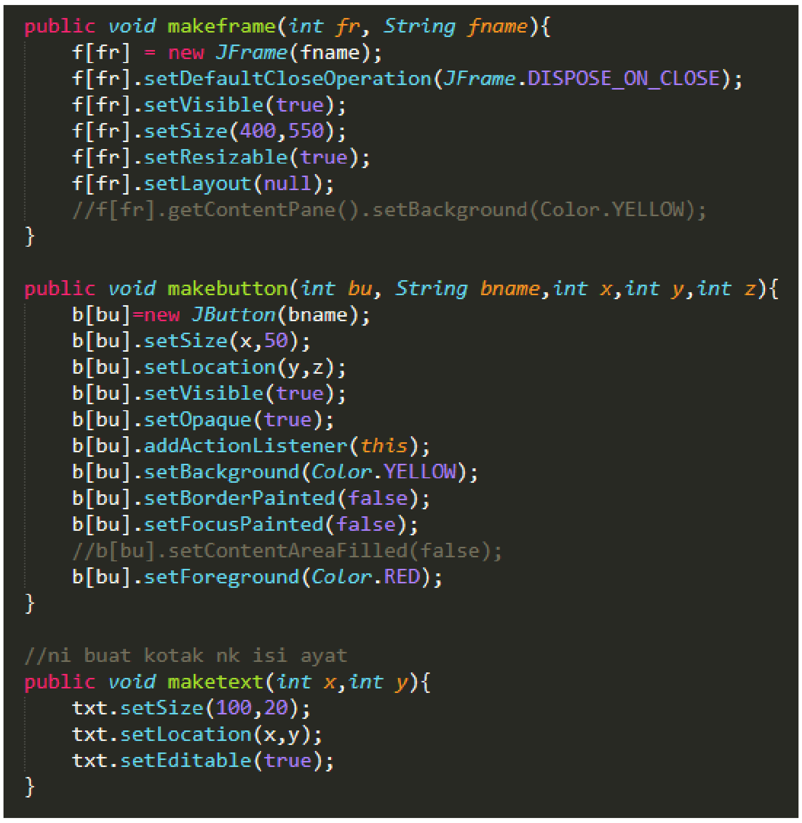
```
public class UKMParking extends JFrame implements ActionListener {
```

This class is the main class that contains the GUI. This is the class the user should run to use the program. It extends JFrame and implements ActionListener



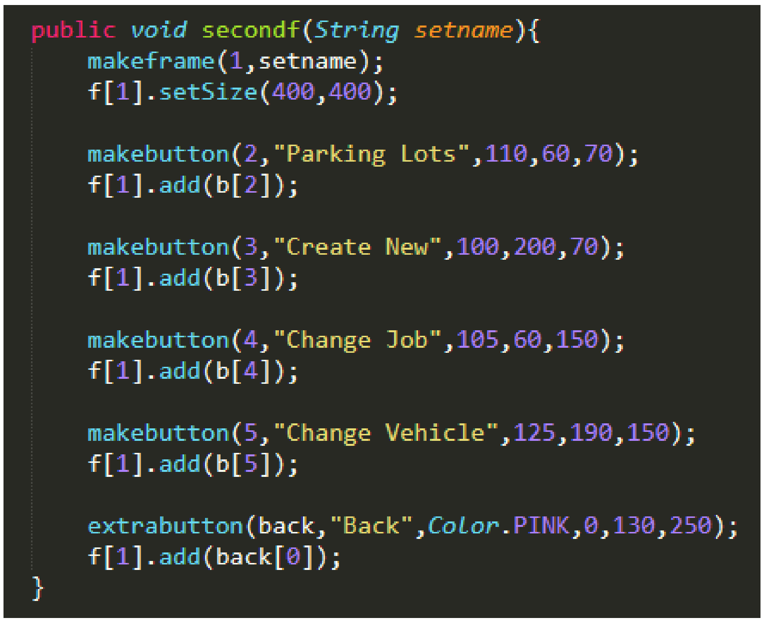
```
static JLabel l[]=new JLabel[5];
JLabel[]lv=new JLabel[7];
private JFrame[] f=new JFrame[9];
JTextArea txt = new JTextArea();
JTextArea []tv=new JTextArea[7];
JRadioButton rb[]=new JRadioButton[3];
JRadioButton[]rbv=new JRadioButton[16];
JButton b[]=new JButton[9];
JButton save[]=new JButton[3];
JButton back[]=new JButton[8];
JButton add[]=new JButton[3];
JButton remove[]=new JButton[3];
String []num={"1","2","3","4"};
JComboBox c=new JComboBox(num);
JComboBox parklist[]=new JComboBox[4];
JComboBox nopark=new JComboBox();
JComboBox delpark=new JComboBox();
```

The buttons, frame, radiobuttons, textarea, lables and other components are created in array to minimize codes.



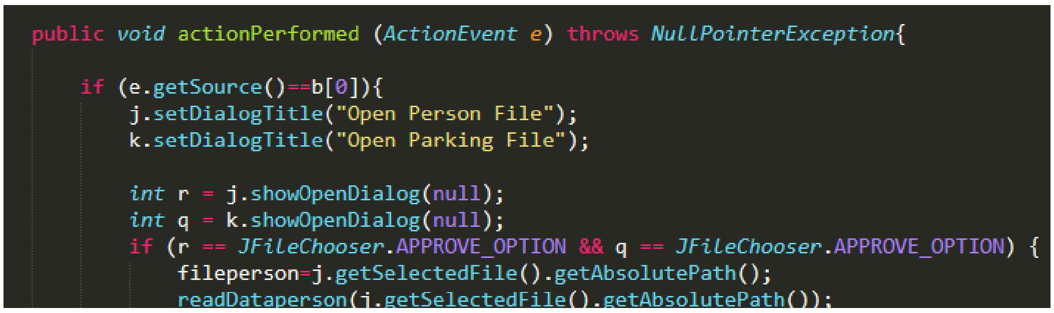
```
public void makeframe(int fr, String fname){
    f[fr] = new JFrame(fname);
    f[fr].setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    f[fr].setVisible(true);
    f[fr].setSize(400,550);
    f[fr].setResizable(true);
    f[fr].setLayout(null);
    //f[fr].getContentPane().setBackground(Color.YELLOW);
}

public void makebutton(int bu, String bname,int x,int y,int z){
    b[bu]=new JButton(bname);
    b[bu].setSize(x,50);
    b[bu].setLocation(y,z);
    b[bu].setVisible(true);
    b[bu].setOpaque(true);
    b[bu].addActionListener(this);
    b[bu].setBackground(Color.YELLOW);
    b[bu].setBorderPainted(false);
    b[bu].setFocusPainted(false);
    //b[bu].setContentAreaFilled(false);
    b[bu].setForeground(Color.RED);
}

//ni buat kotak nk isi ayat
public void maketext(int x,int y){
    txt.setSize(100,20);
    txt.setLocation(x,y);
    txt.setEditable(true);
}
```

The makeframe(), makebutton() maketext() are methods to create any JFrame, JButtons, JTextArea and it can just be called with the parameter needed to make those components.



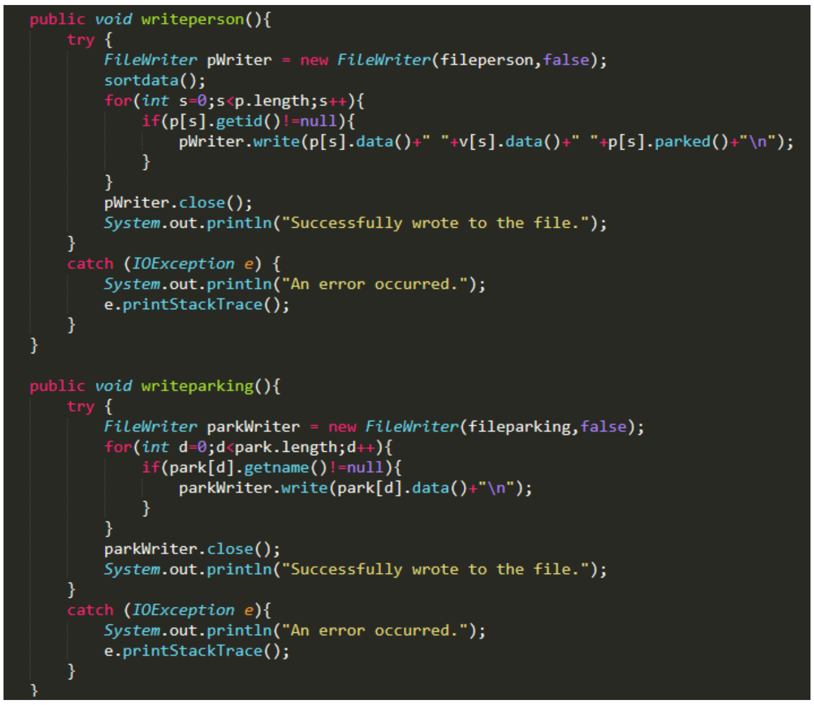
```
public void secondf(String setname){
    makeframe(1,setname);
    f[1].setSize(400,400);

    makebutton(2,"Parking Lots",110,60,70);
    f[1].add(b[2]);

    makebutton(3,"Create New",100,200,70);
    f[1].add(b[3]);

    makebutton(4,"Change Job",105,60,150);
    f[1].add(b[4]);

    makebutton(5,"Change Vehicle",125,190,150);
    f[1].add(b[5]);

    extrabutton(back,"Back",Color.PINK,0,130,250);
    f[1].add(back[0]);
}
```

The firstf(), secondf(), thirdf(), forthf(), fifthf(), sixthf(), seventh(), all the method that end with f are methods that makes the frame. For example, based on the picture o the left, it's a method to make the second frame containing all it's components inside the frame.



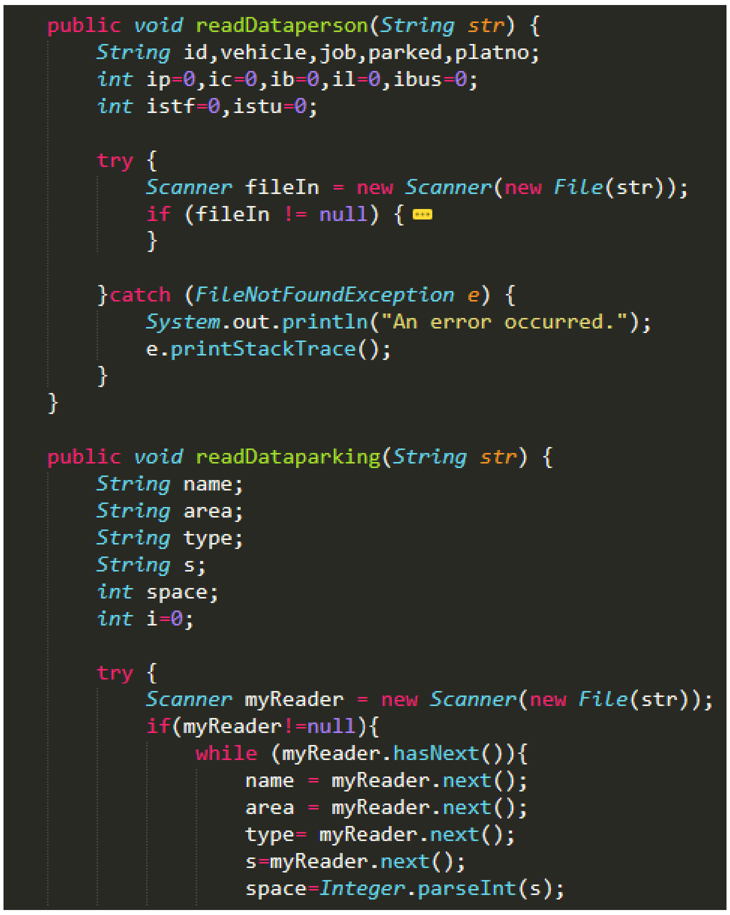
```
public void actionPerformed (ActionEvent e) throws NullPointerException{

    if (e.getSource()==b[0]){
        j.setDialogTitle("Open Person File");
        k.setDialogTitle("Open Parking File");

        int r = j.showOpenDialog(null);
        int q = k.showOpenDialog(null);
        if (r == JFileChooser.APPROVE_OPTION && q == JFileChooser.APPROVE_OPTION) {
            fileperson=j.getSelectedFile().getAbsolutePath();
            readDataperson(j.getSelectedFile().getAbsolutePath());
```

This is the actionPerformed method. This is the method where all the JButtons, JRadiobutton, JList actions take place. It's what makes the GUI work completely.



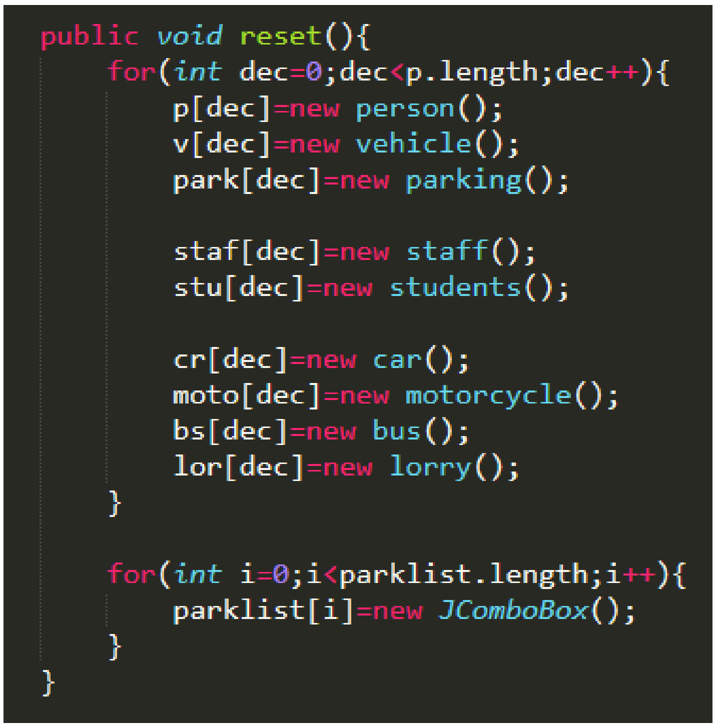
```
public void writeperson(){
    try {
        FileWriter pWriter = new FileWriter(fileperson,false);
        sortdata();
        for(int s=0;s<p.length;s++){
            if(p[s].getid()!=null){
                pWriter.write(p[s].data()+" "+v[s].data()+" "+p[s].parked()+"\n");
            }
        }
        pWriter.close();
        System.out.println("Successfully wrote to the file.");
    }
    catch (IOException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}

public void writeparking(){
    try {
        FileWriter parkWriter = new FileWriter(fileparking,false);
        for(int d=0;d<park.length;d++){
            if(park[d].getname()!=null){
                parkWriter.write(park[d].data()+"\n");
            }
        }
        parkWriter.close();
        System.out.println("Successfully wrote to the file.");
    }
    catch (IOException e){
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}
```

The writeperson() and writeparking() methods are used to write data into the person and parking file. Each time a data is added or altered, this method is called.



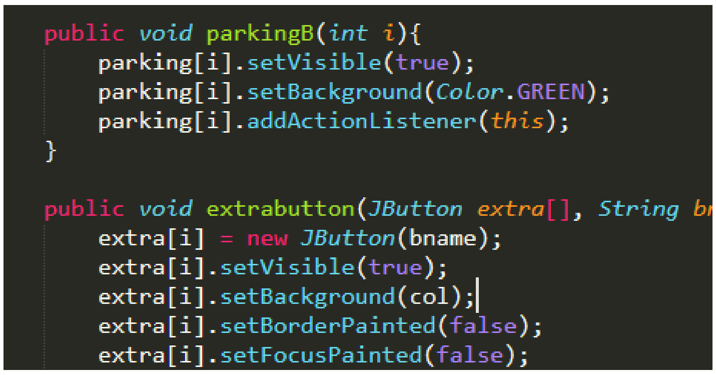
```
public void readDataperson(String str) {
    String id,vehicle,job,parked,platno;
    int ip=0,ic=0,ib=0,il=0,ibus=0;
    int istf=0,istu=0;

    try {
        Scanner fileIn = new Scanner(new File(str));
        if (fileIn != null) {
        }

    }catch (FileNotFoundException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}

public void readDataparking(String str) {
    String name;
    String area;
    String type;
    String s;
    int space;
    int i=0;

    try {
        Scanner myReader = new Scanner(new File(str));
        if(myReader!=null){
            while (myReader.hasNext()){
                name = myReader.next();
                area = myReader.next();
                type= myReader.next();
                s=myReader.next();
                space=Integer.parseInt(s);
```

The readDataperson() and readDataparking() method is called when a file is opened at the starting of the program. It reads the file n the directory the file has been chosen.



```
public void reset(){
    for(int dec=0;dec<p.length;dec++){
        p[dec]=new person();
        v[dec]=new vehicle();
        park[dec]=new parking();

        staf[dec]=new staff();
        stu[dec]=new students();

        cr[dec]=new car();
        moto[dec]=new motorcycle();
        bs[dec]=new bus();
        lor[dec]=new lorry();
    }

    for(int i=0;i<parklist.length;i++){
        parklist[i]=new JComboBox();
    }
}
```

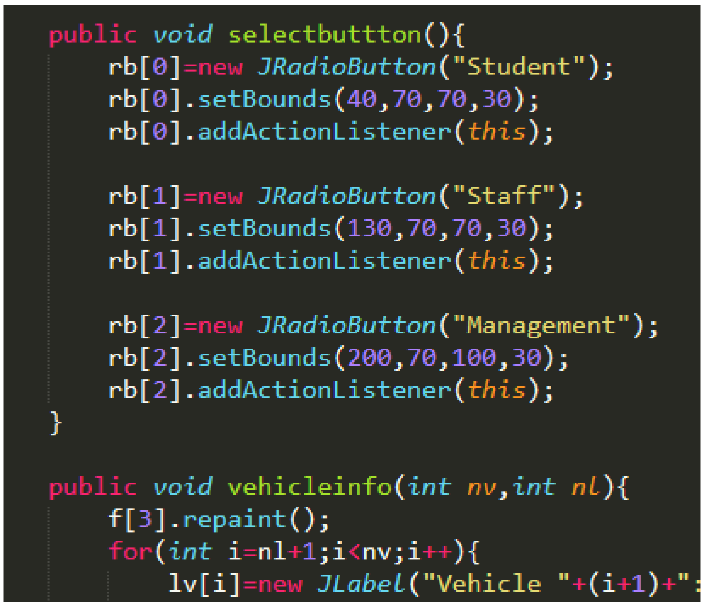
This is the method that creates or make the array 'null' so that if the user wishes to start new all the classes array will be null but it won't effect any files.



```
public void parkingB(int i){
    parking[i].setVisible(true);
    parking[i].setBackground(Color.GREEN);
    parking[i].addActionListener(this);
}

public void extrabutton(JButton extra[], String br
    extra[i] = new JButton(bname);
    extra[i].setVisible(true);
    extra[i].setBackground(col);
    extra[i].setBorderPainted(false);
    extra[i].setFocusPainted(false);
```

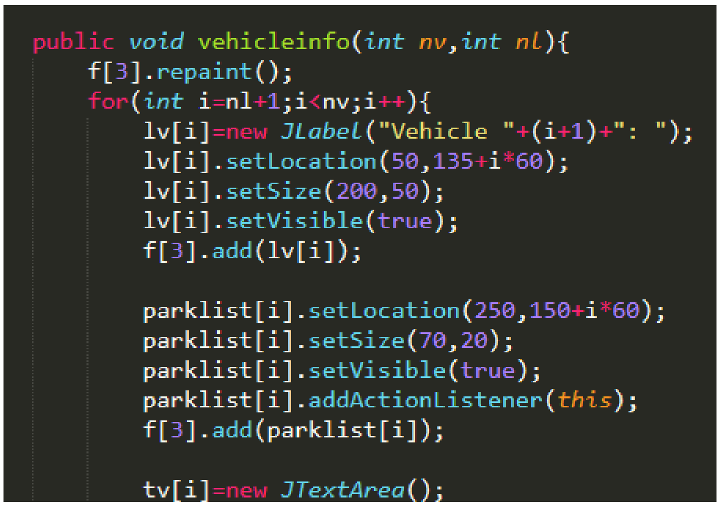
parkingB() method is used to make new parking buttons. Where when user add a new parking the button will be made using the parking.

The extrabutton() method is used to make back, add, remove button. So if one of those button is created and added into frame, it will use this method.



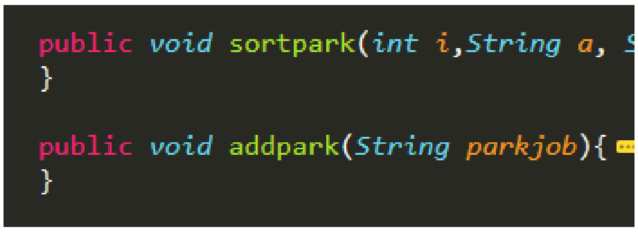
```
public void selectbuttton(){
    rb[0]=new JRadioButton("Student");
    rb[0].setBounds(40,70,70,30);
    rb[0].addActionListener(this);

    rb[1]=new JRadioButton("Staff");
    rb[1].setBounds(130,70,70,30);
    rb[1].addActionListener(this);

    rb[2]=new JRadioButton("Management");
    rb[2].setBounds(200,70,100,30);
    rb[2].addActionListener(this);
}

public void vehicleinfo(int nv,int nl){
    f[3].repaint();
    for(int i=nl+1;i<nv;i++){
        lv[i]=new JLabel("Vehicle "+(i+1)+":
```

selectbuttons() method creates the radiobutton for the create new frame.



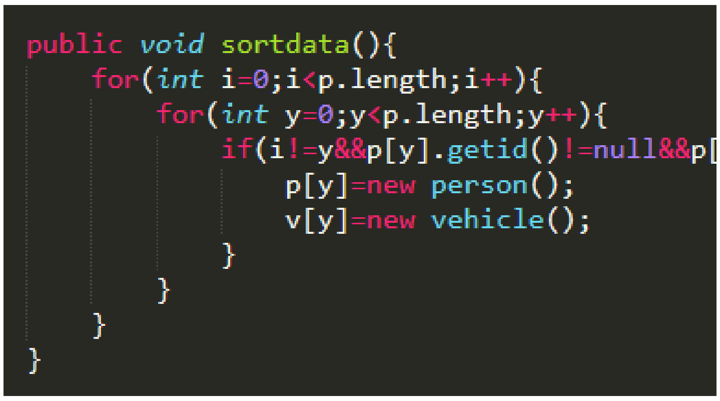
```
public void vehicleinfo(int nv,int nl){
    f[3].repaint();
    for(int i=nl+1;i<nv;i++){
        lv[i]=new JLabel("Vehicle "+(i+1)+": ");
        lv[i].setLocation(50,135+i*60);
        lv[i].setSize(200,50);
        lv[i].setVisible(true);
        f[3].add(lv[i]);

        parklist[i].setLocation(250,150+i*60);
        parklist[i].setSize(70,20);
        parklist[i].setVisible(true);
        parklist[i].addActionListener(this);
        f[3].add(parklist[i]);

        tv[i]=new JTextArea();
```

The vehicleinfo() method handles the addition and removal of textarea, radiobutton and combobox according to the numbers of vehicle selected and the combobox.



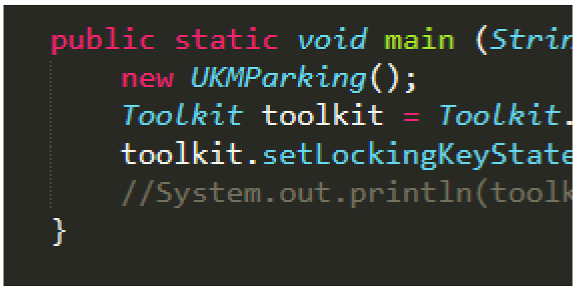
```
public void sortpark(int i,String a, S
}

public void addpark(String parkjob){
}
```

sortpark() and addpark() method handles the addition and removal of strings from comboboxes.



```
public void sortdata(){
    for(int i=0;i<p.length;i++){
        for(int y=0;y<p.length;y++){
            if(i!=y&&p[y].getid()!=null&&p[
                p[y]=new person();
                v[y]=new vehicle();
            }
        }
    }
}
```

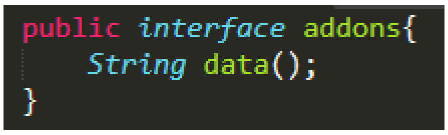
sortdata() method look for any repetitions in data inside the person and vehicle class array and then clears the data to prevent duplication of data when writing.



```
public static void main (Strin
    new UKMParking();
    Toolkit toolkit = Toolkit.
    toolkit.setLockingKeyState
    //System.out.println(toolk
}
```

The last one is the main() method. It starts the program and also turn on the capslock button on the users keyboard.

Interface: Addons



```
public interface addons{
    String data();
}
```

This is the interface file used in person, vehicle, and parking java file.

That's all the classes and methods explained. If you want to run the program, you can run the UKMParking.java file.