Mathematics and Algorithms for Computing Mini-Project #1

Topic

PC to Home

Project Owner

6131501052 Wachirachai Nitsomboon
6131501054 Wirakan Kaewkanya

BECHELOR OF ENGINEERING
IN COMPUTER ENGINEERING

This document is submitted in partial fulfillment of the requirements for 1302206 Mathematics and Algorithms for Computing
Academic Year 2019, Second semester

Mathematics and Algorithms for Computing
Mini-Project #1

Topic

PC to Home

Project Owner

6131501052 Wachirachai Nitsomboon
6131501054 Wirakan Kaewkanya

[illegible]
Signature
(Wachirachai Nitsomboon)

Signature
(Wirakan Kaewkanya)

Project Title

PC to Home

Background and Rationale

Going out can be inconvenient especially now with the pandemic like COVID-19 many people would be more likely to stay inside of their comfort home. This application aims to ease the pain of having to go out and pick up a PC by themselves.

Obejective of the study

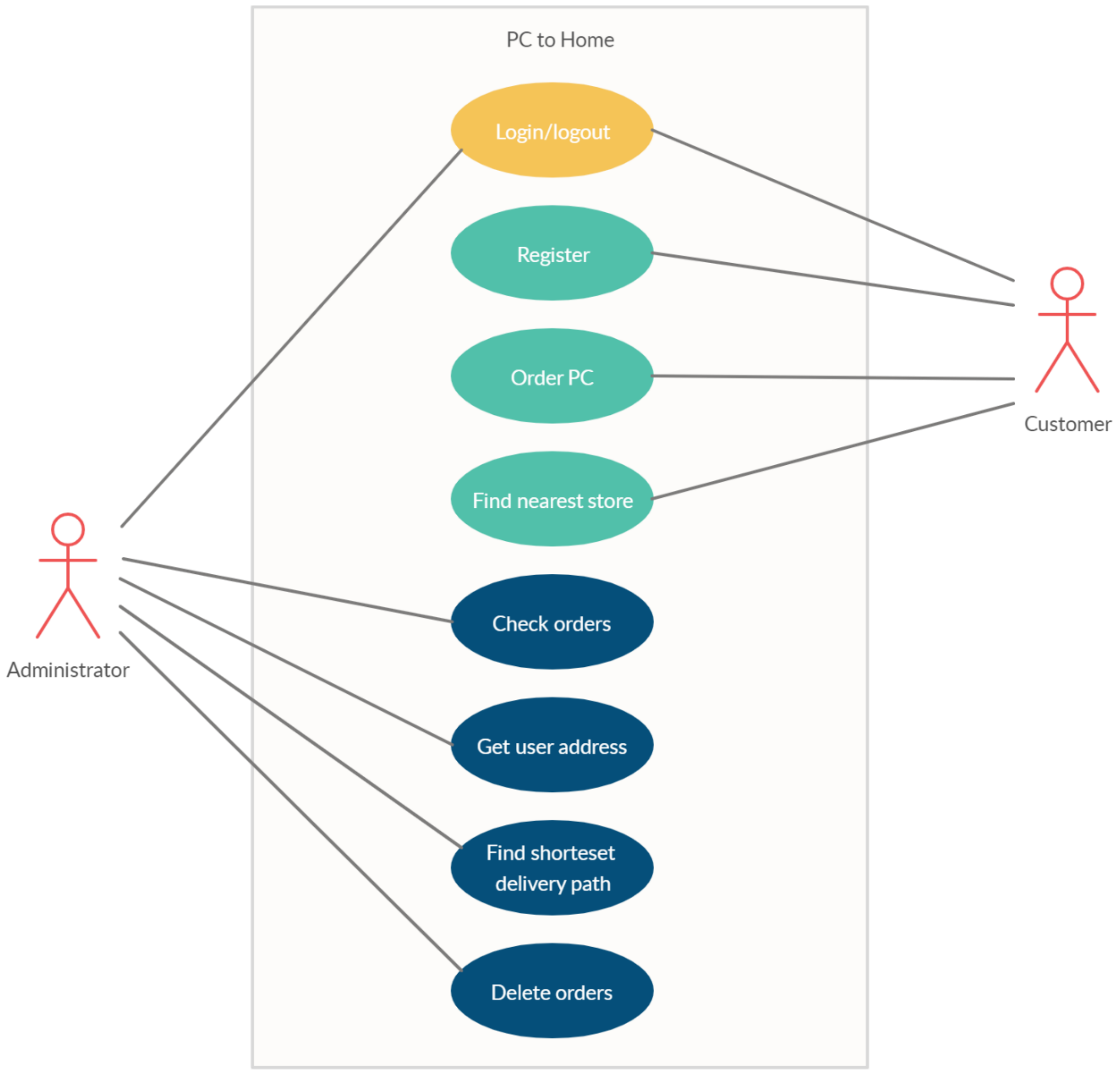
-To make shipping logistics a little easier by finding the shortest path.
-Helps customer find the closest store

Scope of study

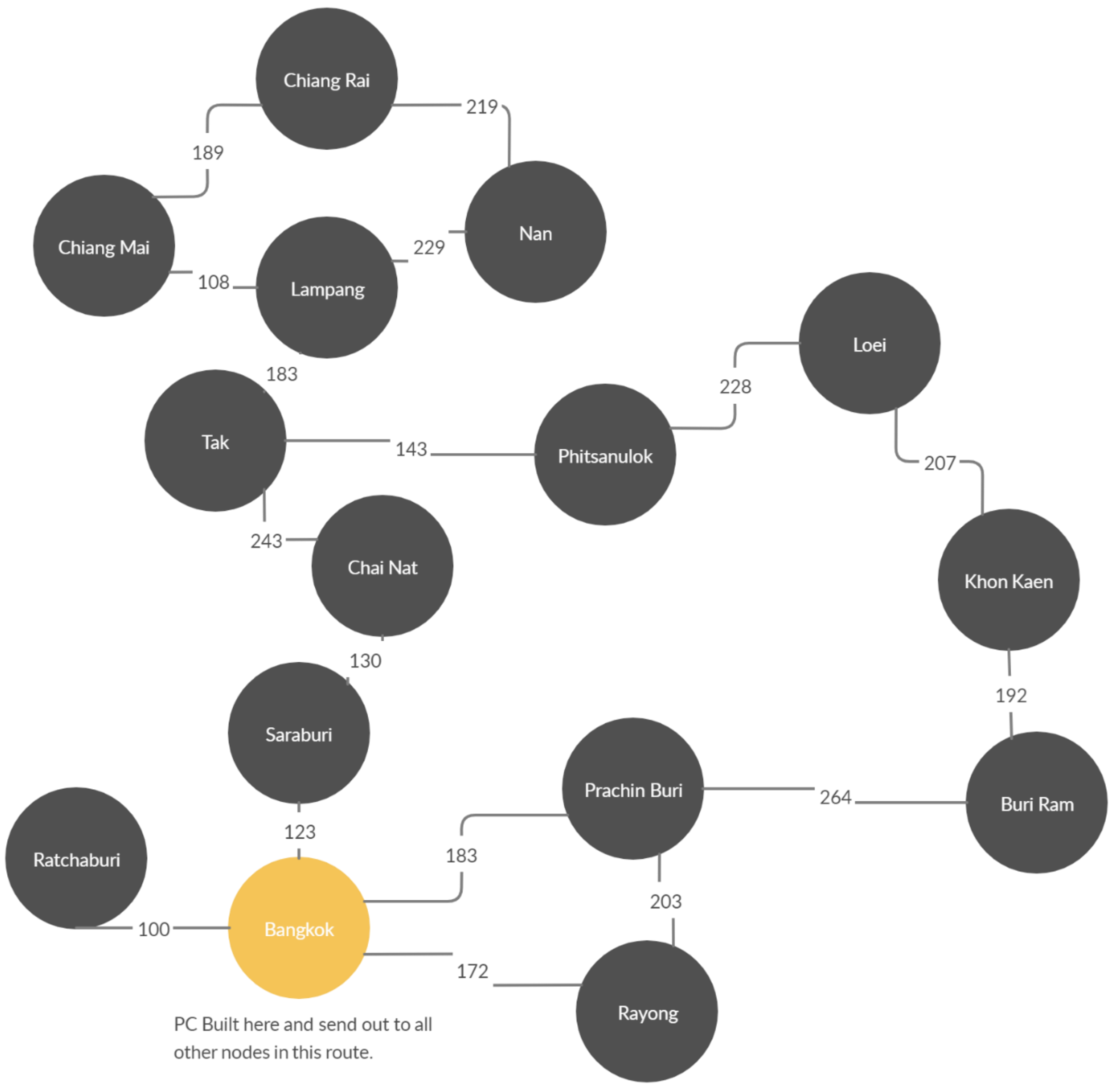
There are two user roles

1. Administrator
   - Login/logout
   - Check orders
   - Delete orders
   - Get user's address from their order
   - Find the shortest delivery path from the delivery route
2. Customers
   - Login/logout
   - Register
   - Order PC
   - Look up the nearest store

Use case diagram

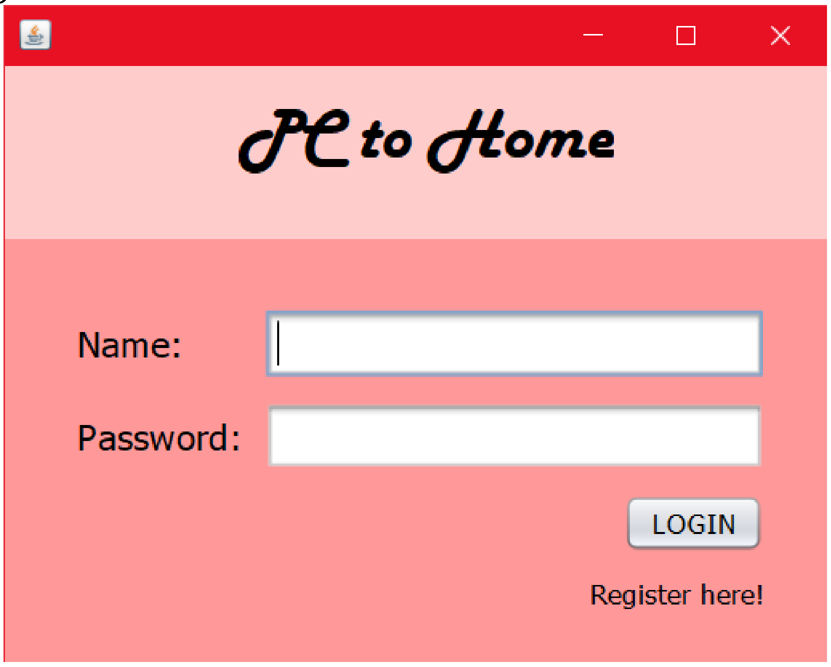


Graph for reference:

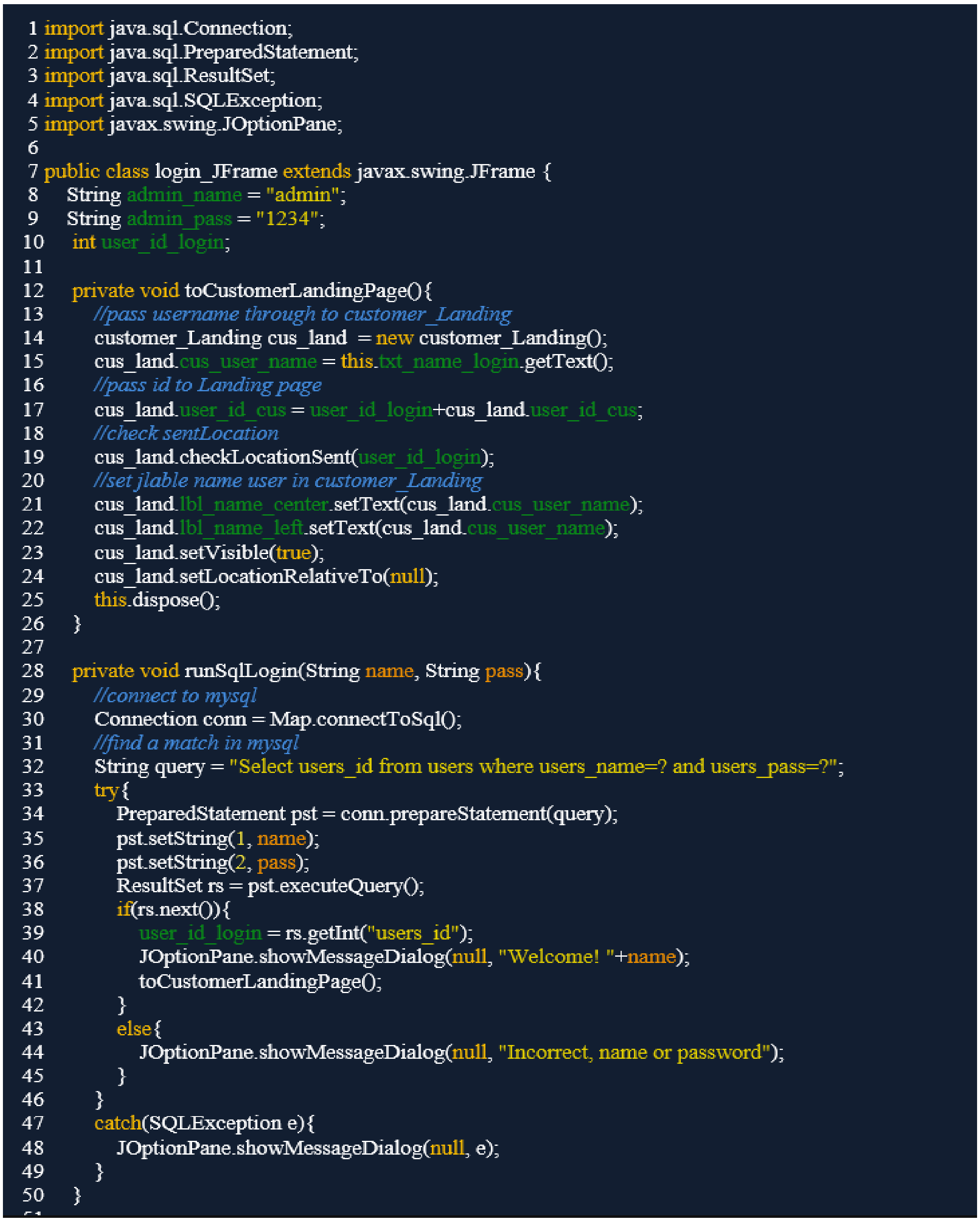


User interface

1.Login

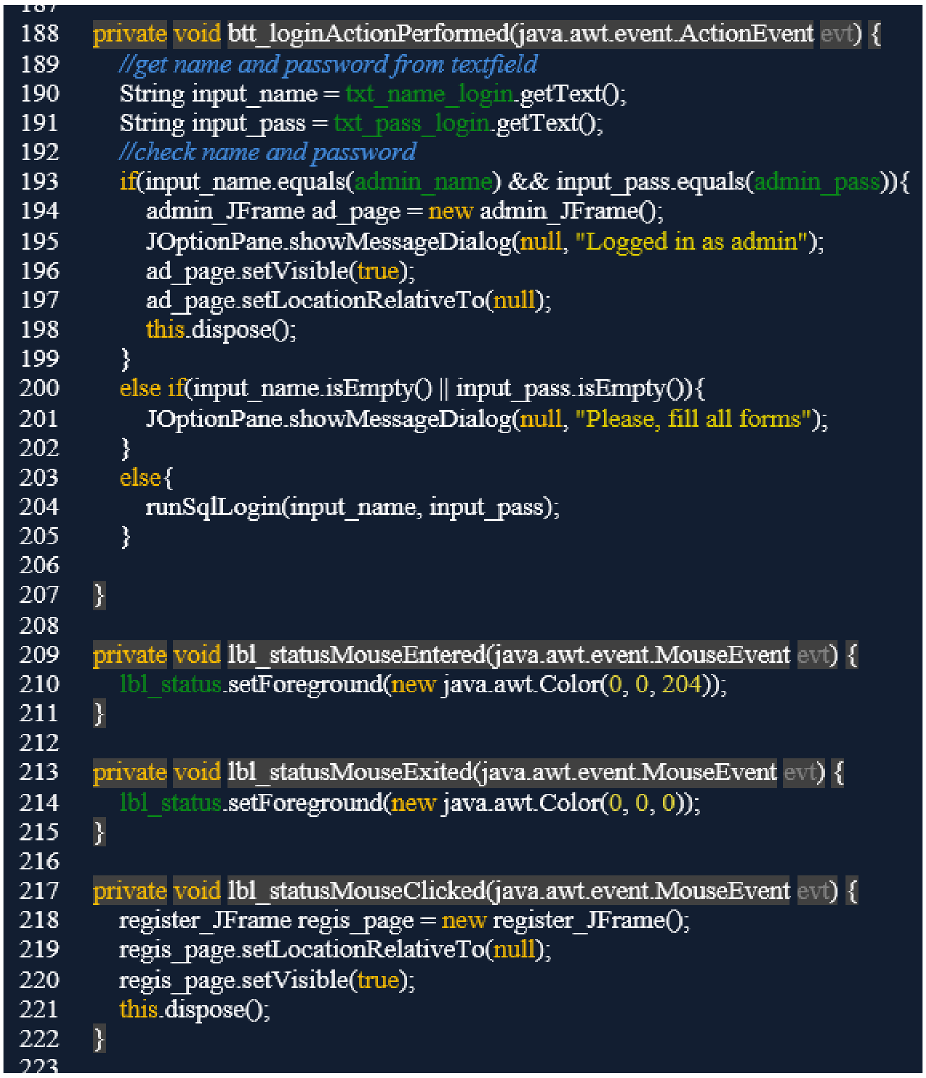


Customers and admin login via this page. Customer needs to register before using the application so that their address can be saved. admin needs to enter the correct name and password (which is admin, 1234).



```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class login_JFrame extends javax.swing.JFrame {
  String admin_name = "admin";
  String admin_pass = "1234";
   int user_id_login;

   private void toCustomerLandingPage(){
     //pass username through to customer_Landing
     customer_Landing cus_land  = new customer_Landing();
     cus_land.cus_user_name = this.txt_name_login.getText();
     //pass id to Landing page
     cus_land.user_id_cus = user_id_login+cus_land.user_id_cus;
     //check sentLocation
     cus_land.checkLocationSent(user_id_login);
     //set jlable name user in customer_Landing
     cus_land.lbl_name_center.setText(cus_land.cus_user_name);
     cus_land.lbl_name_left.setText(cus_land.cus_user_name);
     cus_land.setVisible(true);
     cus_land.setLocationRelativeTo(null);
     this.dispose();
   }

   private void runSqlLogin(String name, String pass){
     //connect to mysql
     Connection conn = Map.connectToSql();
     //find a match in mysql
     String query = "Select users_id from users where users_name=? and users_pass=?";
     try{
        PreparedStatement pst = conn.prepareStatement(query);
        pst.setString(1, name);
        pst.setString(2, pass);
        ResultSet rs = pst.executeQuery();
        if(rs.next()){
           user_id_login = rs.getInt("users_id");
           JOptionPane.showMessageDialog(null, "Welcome! "+name);
           toCustomerLandingPage();
        }
        else{
           JOptionPane.showMessageDialog(null, "Incorrect, name or password");
        }
     }
     catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
     }
   }
```

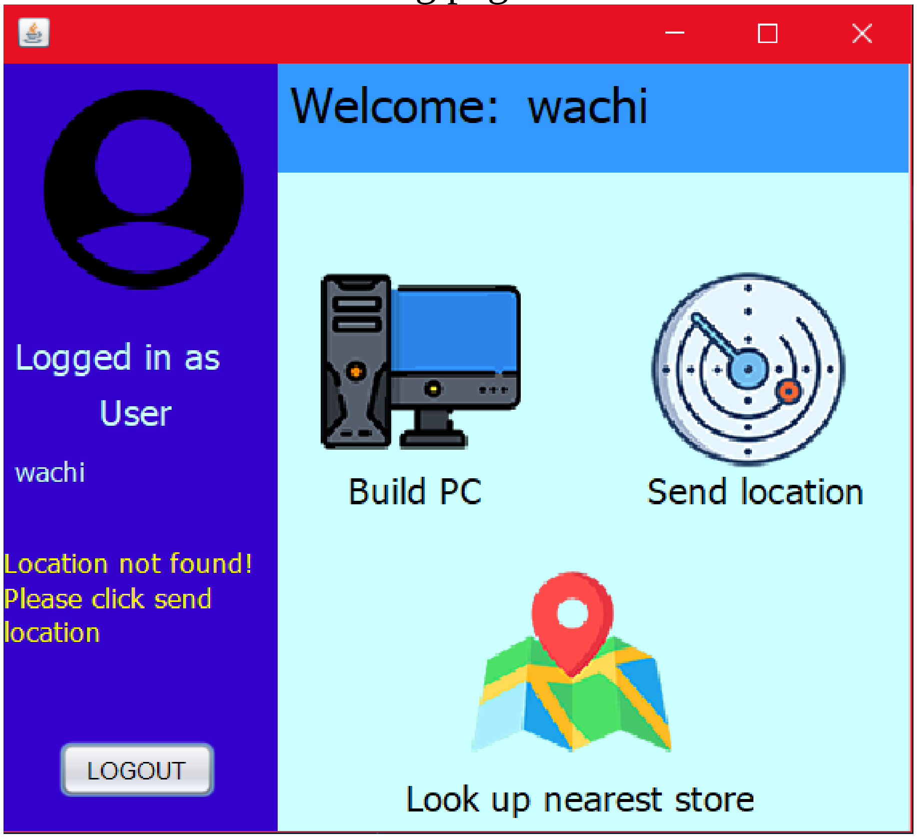
At the top of the login page, there’re 2 methods. ToCustomerLandingPage method used for transition to another jFrame and runSqlLogin used to check the name and password to the MySQL database.



```java
private void btt_loginActionPerformed(java.awt.event.ActionEvent evt) {
    //get name and password from textfield
    String input_name = txt_name_login.getText();
    String input_pass = txt_pass_login.getText();
    //check name and password
    if(input_name.equals(admin_name) && input_pass.equals(admin_pass)){
        admin_JFrame ad_page = new admin_JFrame();
        JOptionPane.showMessageDialog(null, "Logged in as admin");
        ad_page.setVisible(true);
        ad_page.setLocationRelativeTo(null);
        this.dispose();
    }
    else if(input_name.isEmpty() || input_pass.isEmpty()){
        JOptionPane.showMessageDialog(null, "Please, fill all forms");
    }
    else{
        runSqlLogin(input_name, input_pass);
    }

}

private void lbl_statusMouseEntered(java.awt.event.MouseEvent evt) {
    lbl_status.setForeground(new java.awt.Color(0, 0, 204));
}

private void lbl_statusMouseExited(java.awt.event.MouseEvent evt) {
    lbl_status.setForeground(new java.awt.Color(0, 0, 0));
}

private void lbl_statusMouseClicked(java.awt.event.MouseEvent evt) {
    register_JFrame regis_page = new register_JFrame();
    regis_page.setLocationRelativeTo(null);
    regis_page.setVisible(true);
    this.dispose();
}
```

Login button will get name and password, check with the database and if it’s a registered user they will be transfer to the next page. Lbl_status will display register which can be click to go to the register page.

2.Customer landing page


Welcome: wachi

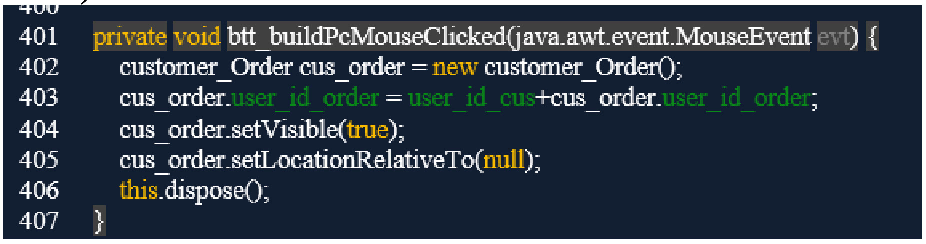
Logged in as User

wachi

Location not found! Please click send location

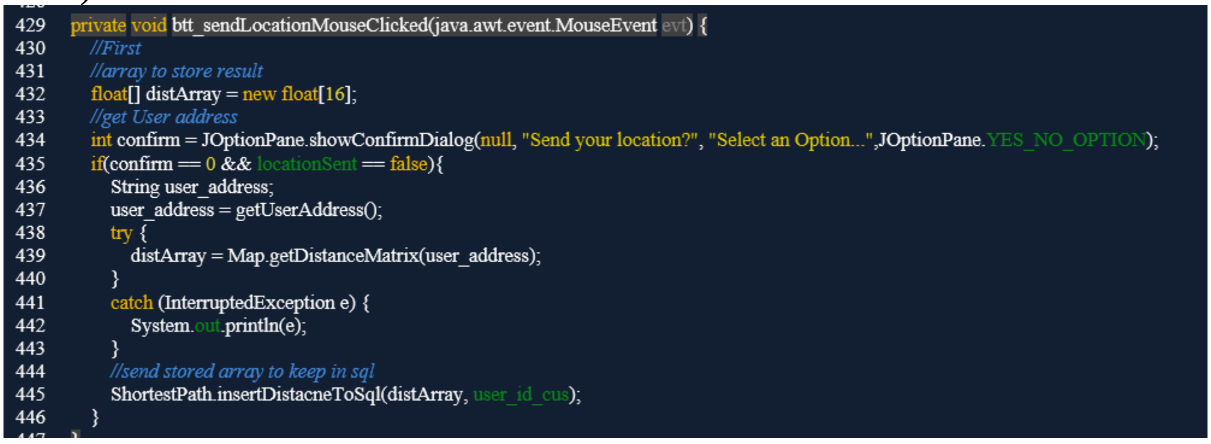
User can go to build PC, send location and look up the nearest store. To look up any store user needs to send their location first (like how GPS would work).

1.) First lets look at Build PC


```java
private void btt_buildPcMouseClicked(java.awt.event.MouseEvent evt) {
    customer_Order cus_order = new customer_Order();
    cus_order.user_id_order = user_id_cus+cus_order.user_id_order;
    cus_order.setVisible(true);
    cus_order.setLocationRelativeTo(null);
    this.dispose();
}
```

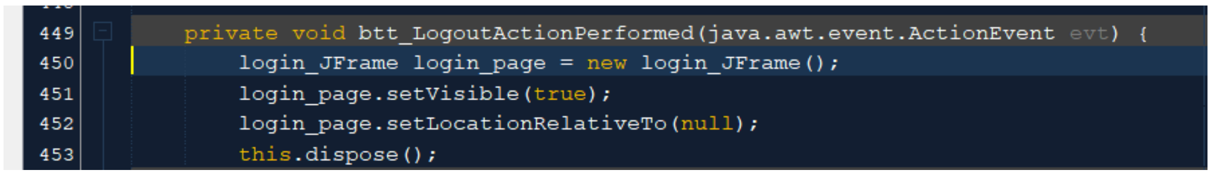
When user came in to this page their user id also came along with them. And when they are going to Build PC their id will be transfer there too.

2.) Send location



```
private void btt_sendLocationMouseClicked(java.awt.event.MouseEvent evt) {
    //First
    //array to store result
    float[] distArray = new float[16];
    //get User address
    int confirm = JOptionPane.showConfirmDialog(null, "Send your location?", "Select an Option...",JOptionPane.YES_NO_OPTION);
    if(confirm == 0 && locationSent == false){
        String user_address;
        user_address = getUserAddress();
        try {
            distArray = Map.getDistanceMatrix(user_address);
        }
        catch (InterruptedException e) {
            System.out.println(e);
        }
        //send stored array to keep in sql
        ShortestPath.insertDistacneToSql(distArray, user_id_cus);
    }
```

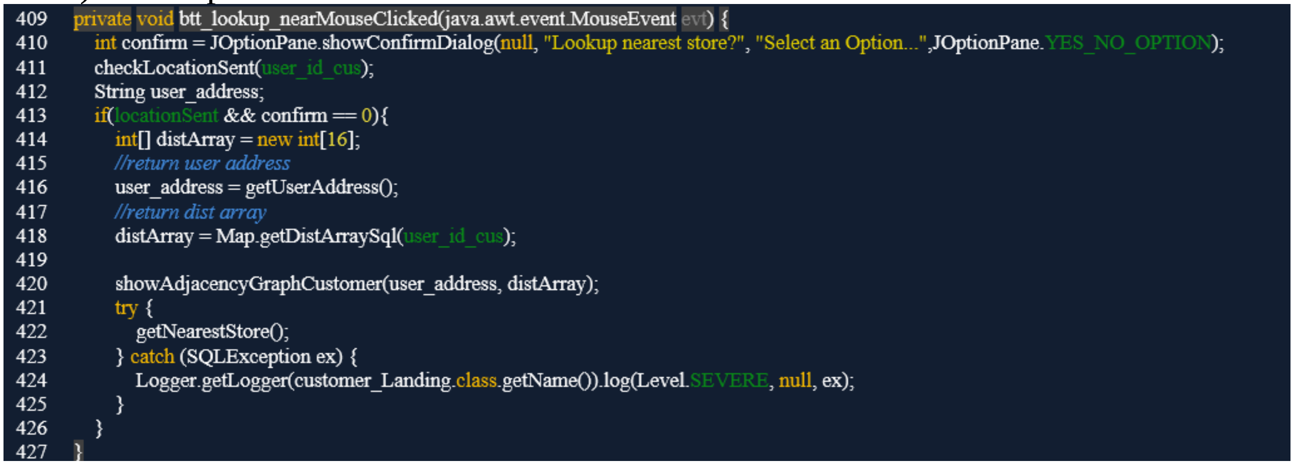
It'll start by getting the user's address then pass into the getDistanceMatrix() method in Map class. This method will run Google's distance matrix API this API will compare distance between two places see the code here.



```
private void btt_LogoutActionPerformed(java.awt.event.ActionEvent evt) {
    login_JFrame login_page = new login_JFrame();
    login_page.setVisible(true);
    login_page.setLocationRelativeTo(null);
    this.dispose();
```

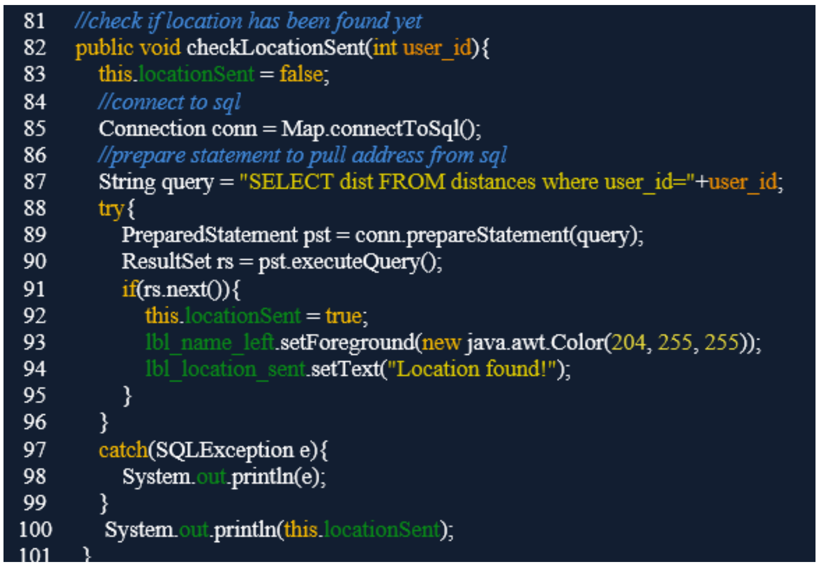
A simple code for logout button

3.) Look up nearest store



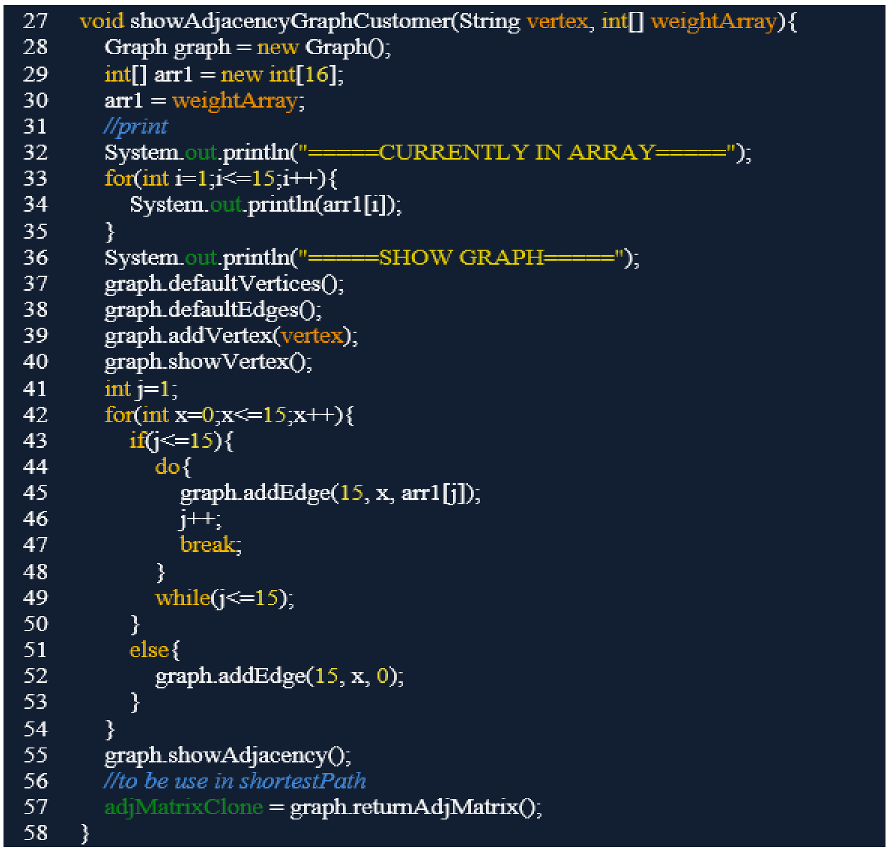
```
private void btt_lookup_nearMouseClicked(java.awt.event.MouseEvent evt) {
    int confirm = JOptionPane.showConfirmDialog(null, "Lookup nearest store?", "Select an Option...",JOptionPane.YES_NO_OPTION);
    checkLocationSent(user_id_cus);
    String user_address;
    if(locationSent && confirm == 0){
        int[] distArray = new int[16];
        //return user address
        user_address = getUserAddress();
        //return dist array
        distArray = Map.getDistArraySql(user_id_cus);

        showAdjacencyGraphCustomer(user_address, distArray);
        try {
            getNearestStore();
        } catch (SQLException ex) {
            Logger.getLogger(customer_Landing.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Start by using checkLocationSent() to see if user has already sent their location or not.



```
//check if location has been found yet
public void checkLocationSent(int user_id){
    this.locationSent = false;
    //connect to sql
    Connection conn = Map.connectToSql();
    //prepare statement to pull address from sql
    String query = "SELECT dist FROM distances where user_id="+user_id;
    try{
        PreparedStatement pst = conn.prepareStatement(query);
        ResultSet rs = pst.executeQuery();
        if(rs.next()){
            this.locationSent = true;
            lbl_name_left.setForeground(new java.awt.Color(204, 255, 255));
            lbl_location_sent.setText("Location found!");
        }
    }
    catch(SQLException e){
        System.out.println(e);
    }
     System.out.println(this.locationSent);
}
```

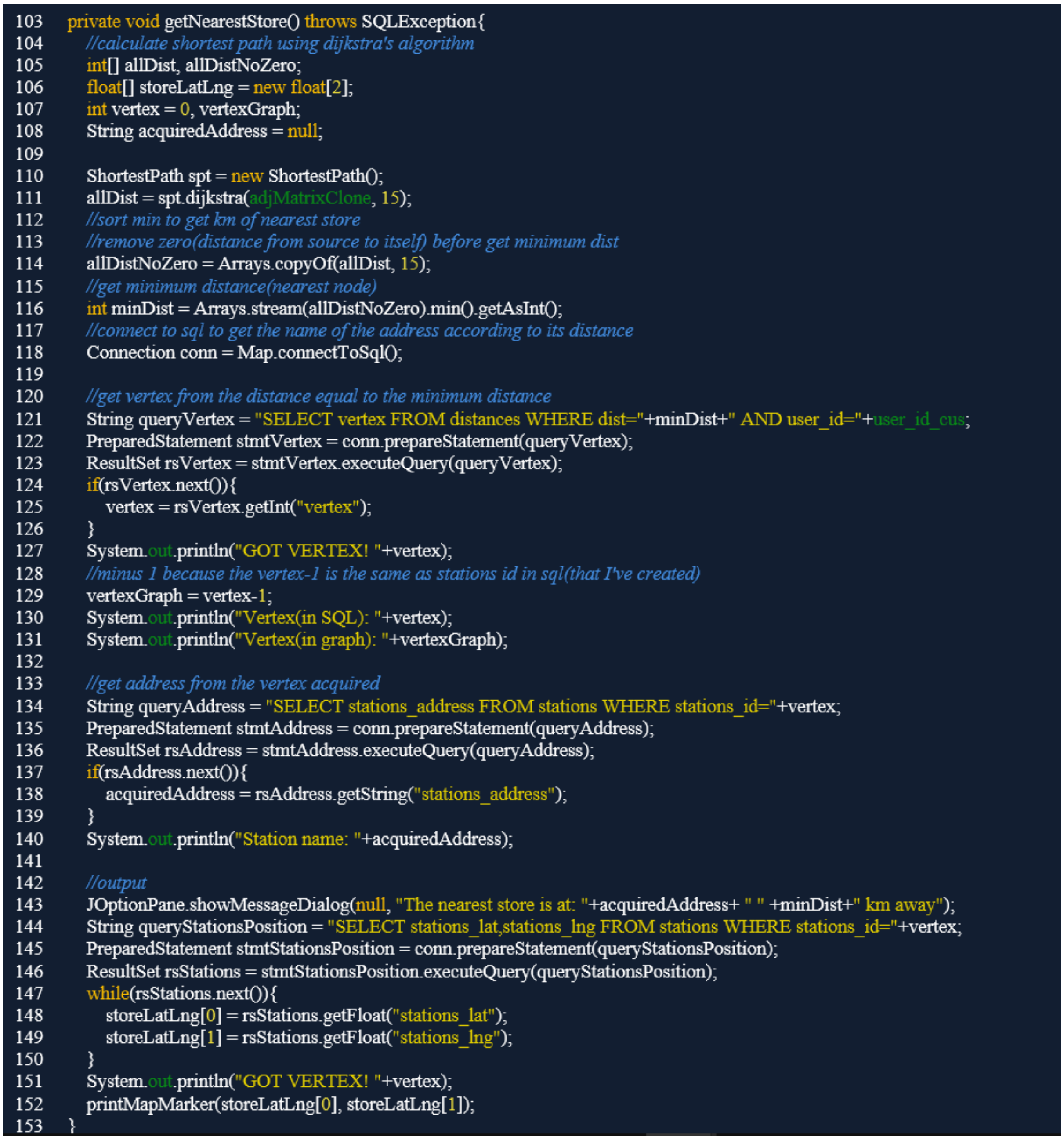
If location has been sent getDistArraySql() will get the distances between user and all other nodes(15 of them). All 15 nodes represent a store in 15 provinces That are connected like a delivery route. We'll use them to find the nearest available store to the user's address. Then showAdjacencyGraphCustomer().



```
void showAdjacencyGraphCustomer(String vertex, int[] weightArray){
    Graph graph = new Graph();
    int[] arr1 = new int[16];
    arr1 = weightArray;
    //print
    System.out.println("=====CURRENTLY IN ARRAY=====");
    for(int i=1;i<=15;i++){
        System.out.println(arr1[i]);
    }
    System.out.println("=====SHOW GRAPH=====");
    graph.defaultVertices();
    graph.defaultEdges();
    graph.addVertex(vertex);
    graph.showVertex();
    int j=1;
    for(int x=0;x<=15;x++){
        if(j<=15){
            do{
                graph.addEdge(15, x, arr1[j]);
                j++;
                break;
            }
            while(j<=15);
        }
        else{
            graph.addEdge(15, x, 0);
        }
    }
    graph.showAdjacency();
    //to be use in shortestPath
    adjMatrixClone = graph.returnAdjMatrix();
}
```

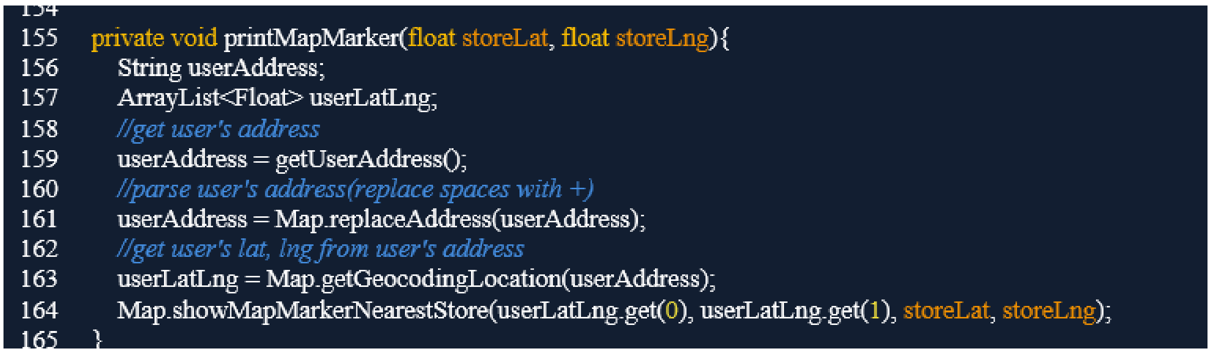
This is a modified version of showAdjacencyGraph in Graph class.

It'll set adjMatrixClone to be used in dijkstra's alogorithm to find the distances from source. More about dijkstar's alogorithm used here.



```
private void getNearestStore() throws SQLException{
    //calculate shortest path using dijkstra's algorithm
    int[] allDist, allDistNoZero;
    float[] storeLatLng = new float[2];
    int vertex = 0, vertexGraph;
    String acquiredAddress = null;

    ShortestPath spt = new ShortestPath();
    allDist = spt.dijkstra(adjMatrixClone, 15);
    //sort min to get km of nearest store
    //remove zero(distance from source to itself) before get minimum dist
    allDistNoZero = Arrays.copyOf(allDist, 15);
    //get minimum distance(nearest node)
    int minDist = Arrays.stream(allDistNoZero).min().getAsInt();
    //connect to sql to get the name of the address according to its distance
    Connection conn = Map.connectToSql();

    //get vertex from the distance equal to the minimum distance
    String queryVertex = "SELECT vertex FROM distances WHERE dist="+minDist+" AND user_id="+user_id_cus;
    PreparedStatement stmtVertex = conn.prepareStatement(queryVertex);
    ResultSet rsVertex = stmtVertex.executeQuery(queryVertex);
    if(rsVertex.next()){
        vertex = rsVertex.getInt("vertex");
    }
    System.out.println("GOT VERTEX! "+vertex);
    //minus 1 because the vertex-1 is the same as stations id in sql(that I've created)
    vertexGraph = vertex-1;
    System.out.println("Vertex(in SQL): "+vertex);
    System.out.println("Vertex(in graph): "+vertexGraph);

    //get address from the vertex acquired
    String queryAddress = "SELECT stations_address FROM stations WHERE stations_id="+vertex;
    PreparedStatement stmtAddress = conn.prepareStatement(queryAddress);
    ResultSet rsAddress = stmtAddress.executeQuery(queryAddress);
    if(rsAddress.next()){
        acquiredAddress = rsAddress.getString("stations_address");
    }
    System.out.println("Station name: "+acquiredAddress);

    //output
    JOptionPane.showMessageDialog(null, "The nearest store is at: "+acquiredAddress+ " " +minDist+" km away");
    String queryStationsPosition = "SELECT stations_lat,stations_lng FROM stations WHERE stations_id="+vertex;
    PreparedStatement stmtStationsPosition = conn.prepareStatement(queryStationsPosition);
    ResultSet rsStations = stmtStationsPosition.executeQuery(queryStationsPosition);
    while(rsStations.next()){
        storeLatLng[0] = rsStations.getFloat("stations_lat");
        storeLatLng[1] = rsStations.getFloat("stations_lng");
    }
    System.out.println("GOT VERTEX! "+vertex);
    printMapMarker(storeLatLng[0], storeLatLng[1]);
}
```

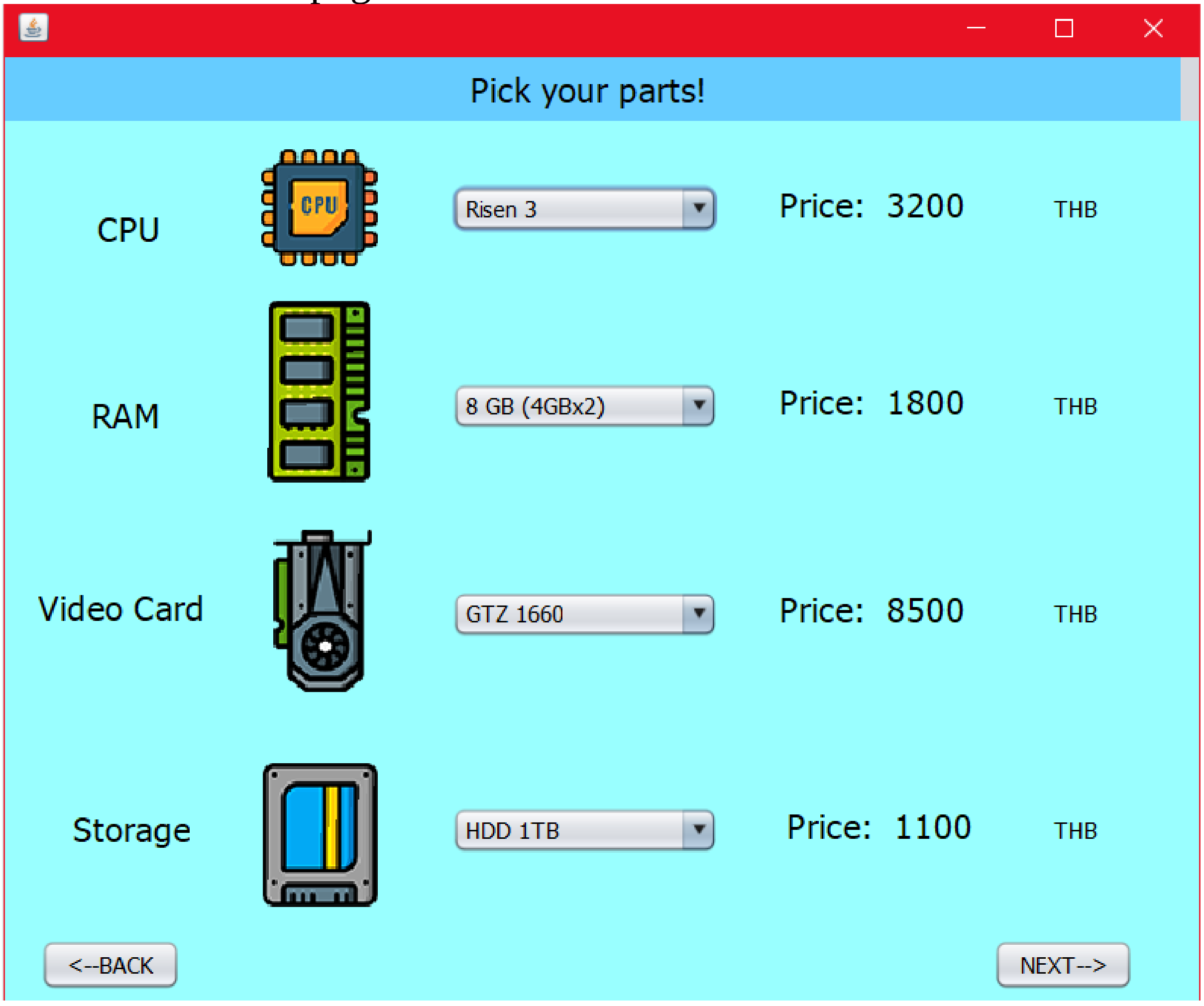
Lastly, getNearestStore() will run dijkstar's alogorithm in ShortestPath class. After that JOptionPane will show message to tell which store is the nearest and how far from the user's location. And the printMapMarker() will run showing a marked map between user and the store.



```
private void printMapMarker(float storeLat, float storeLng){
    String userAddress;
    ArrayList<Float> userLatLng;
    //get user's address
    userAddress = getUserAddress();
    //parse user's address(replace spaces with +)
    userAddress = Map.replaceAddress(userAddress);
    //get user's lat, lng from user's address
    userLatLng = Map.getGeocodingLocation(userAddress);
    Map.showMapMarkerNearestStore(userLatLng.get(0), userLatLng.get(1), storeLat, storeLng);
}
```

printMapMarker() code.

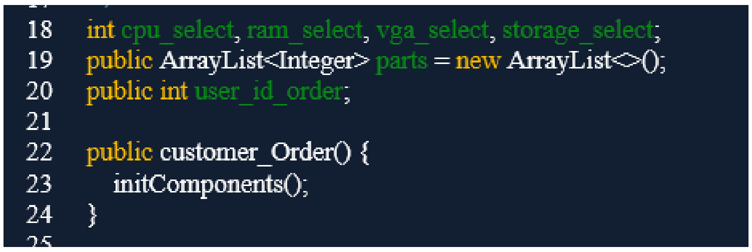
3.Customer order page



Pick your parts!

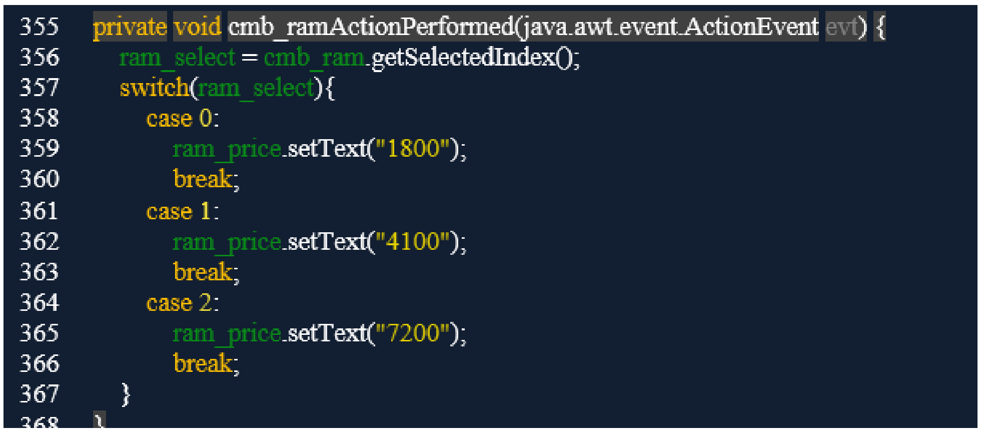
CPU	Risen 3	Price: 3200	THB
RAM	8 GB (4GBx2)	Price: 1800	THB
Video Card	GTZ 1660	Price: 8500	THB
Storage	HDD 1TB	Price: 1100	THB

Customer can pick PC parts here before place the order.



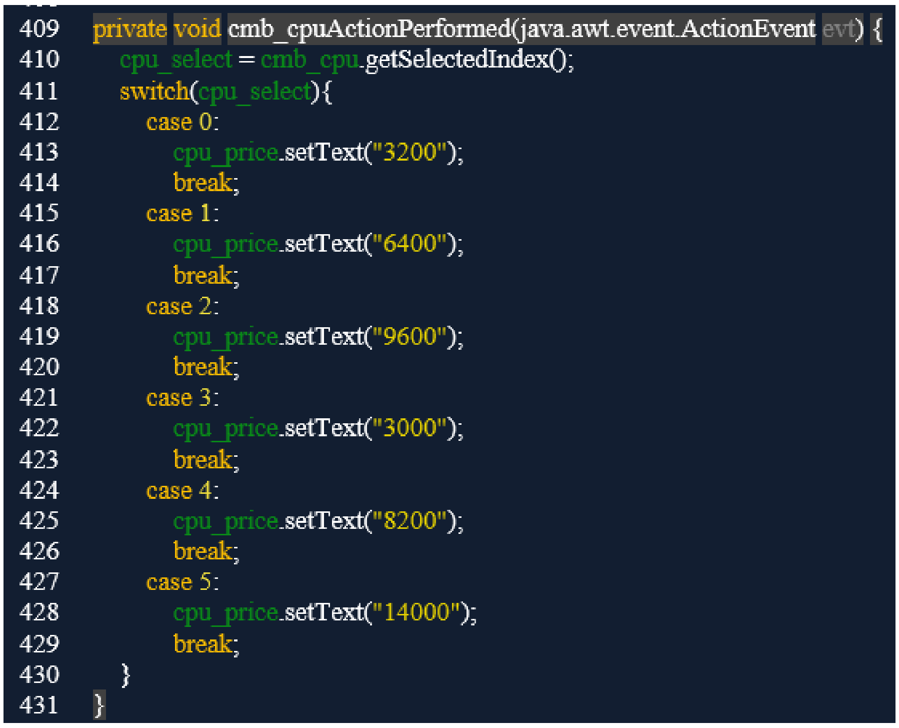
```java
int cpu_select, ram_select, vga_select, storage_select;
public ArrayList<Integer> parts = new ArrayList<>();
public int user_id_order;

public customer_Order() {
    initComponents();
}
```

Declared variables at the top.



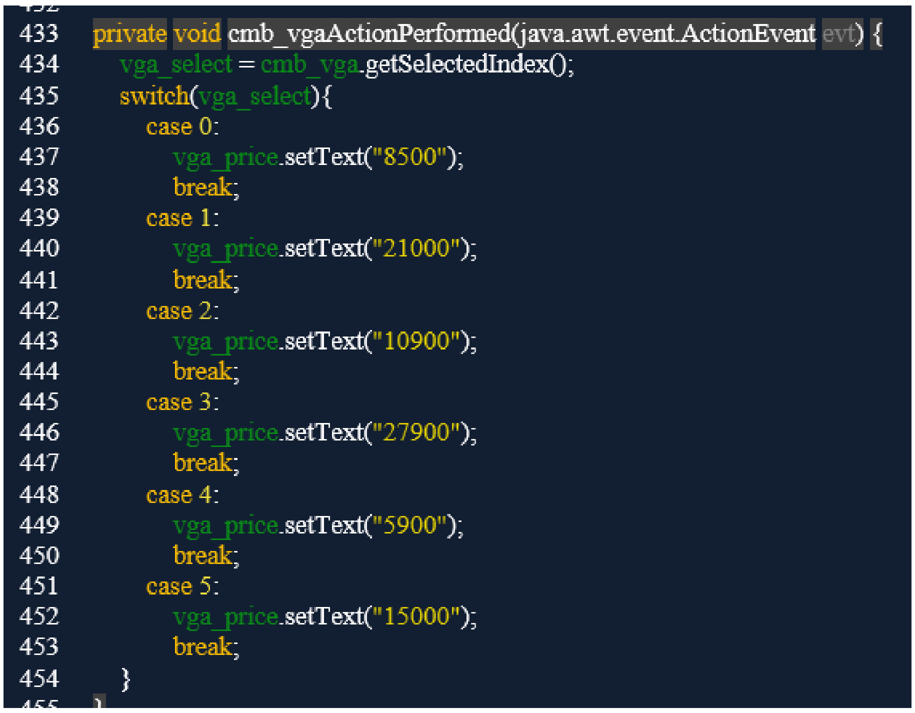
```java
private void cmb_ramActionPerformed(java.awt.event.ActionEvent evt) {
    ram_select = cmb_ram.getSelectedIndex();
    switch(ram_select){
        case 0:
            ram_price.setText("1800");
            break;
        case 1:
            ram_price.setText("4100");
            break;
        case 2:
            ram_price.setText("7200");
            break;
    }
}
```

Combo box for choosing RAM.



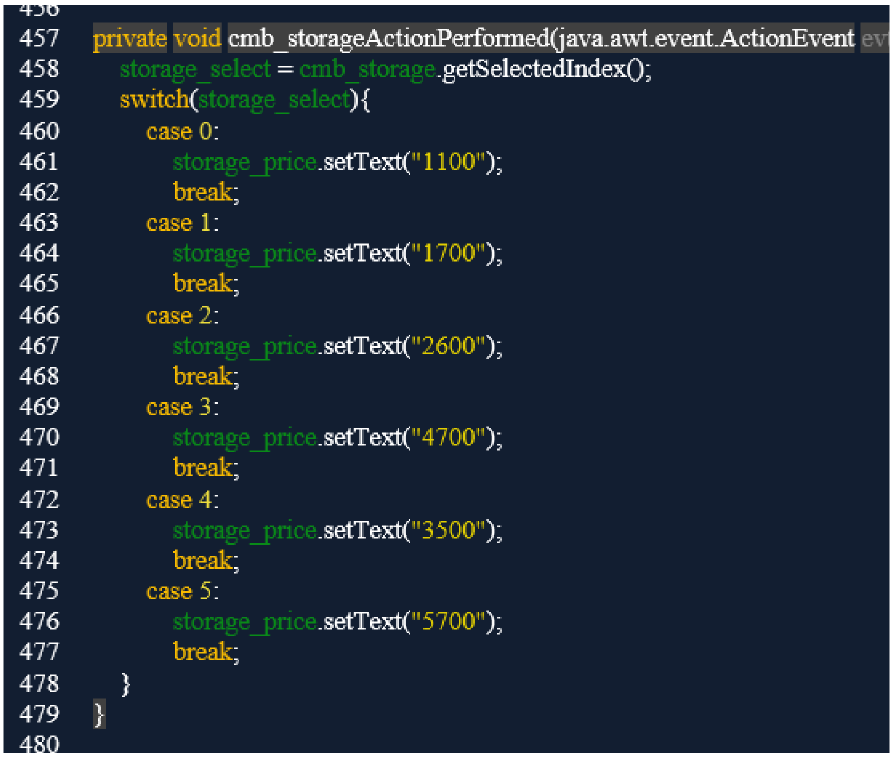
```java
private void cmb_cpuActionPerformed(java.awt.event.ActionEvent evt) {
    cpu_select = cmb_cpu.getSelectedIndex();
    switch(cpu_select){
        case 0:
            cpu_price.setText("3200");
            break;
        case 1:
            cpu_price.setText("6400");
            break;
        case 2:
            cpu_price.setText("9600");
            break;
        case 3:
            cpu_price.setText("3000");
            break;
        case 4:
            cpu_price.setText("8200");
            break;
        case 5:
            cpu_price.setText("14000");
            break;
    }
}
```

Combo box for choosing CPU.



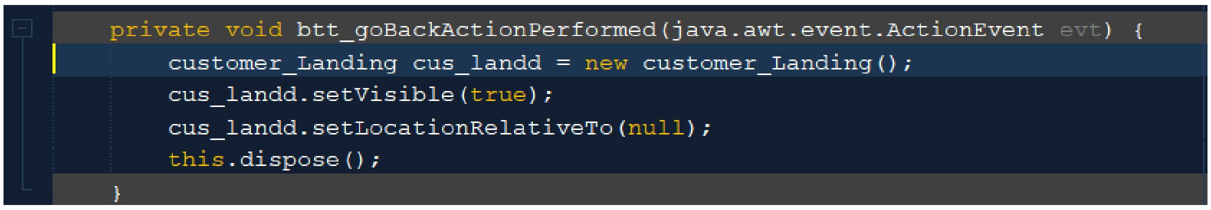
```java
private void cmb_vgaActionPerformed(java.awt.event.ActionEvent evt) {
    vga_select = cmb_vga.getSelectedIndex();
    switch(vga_select){
        case 0:
            vga_price.setText("8500");
            break;
        case 1:
            vga_price.setText("21000");
            break;
        case 2:
            vga_price.setText("10900");
            break;
        case 3:
            vga_price.setText("27900");
            break;
        case 4:
            vga_price.setText("5900");
            break;
        case 5:
            vga_price.setText("15000");
            break;
    }
}
```

Combo box for choosing VGA.



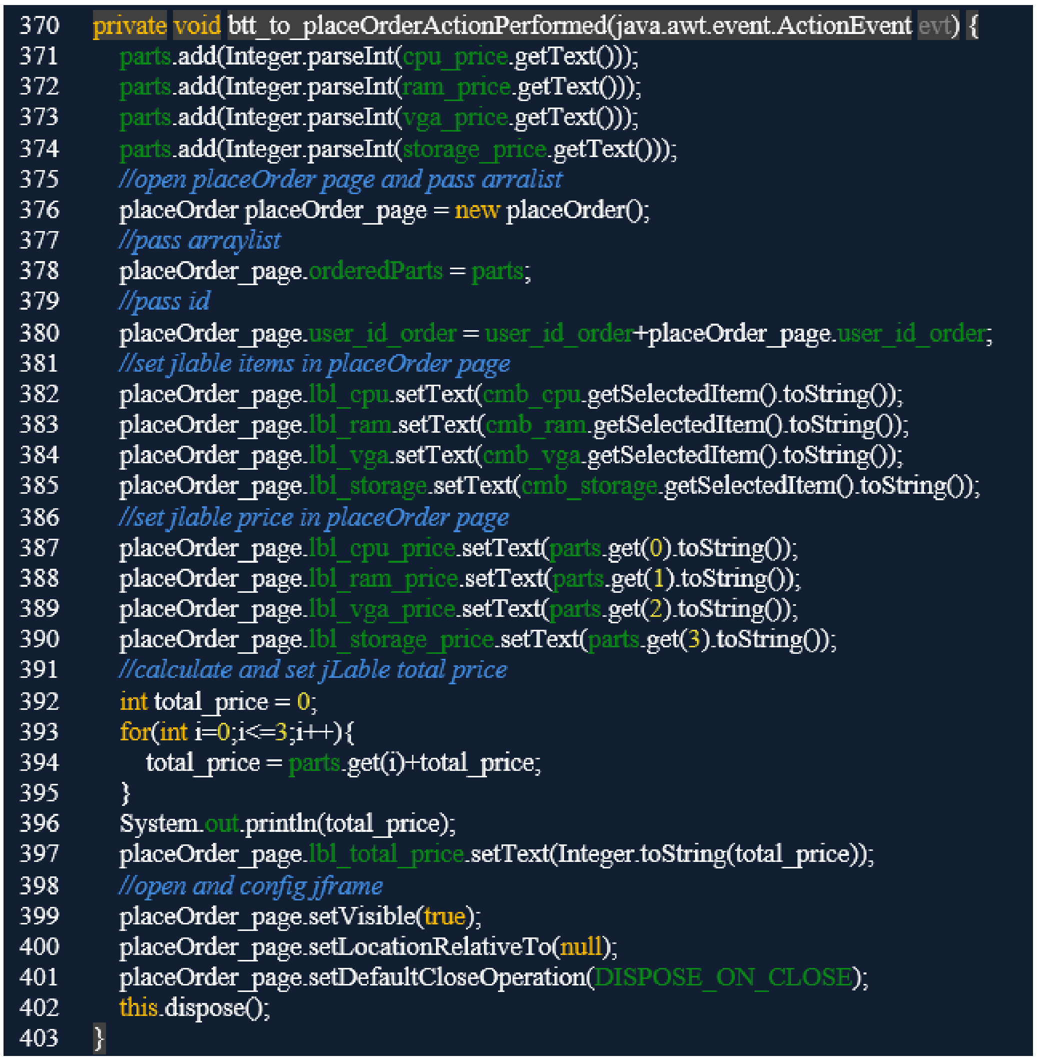
```java
private void cmb_storageActionPerformed(java.awt.event.ActionEvent ev
    storage_select = cmb_storage.getSelectedIndex();
    switch(storage_select){
        case 0:
            storage_price.setText("1100");
            break;
        case 1:
            storage_price.setText("1700");
            break;
        case 2:
            storage_price.setText("2600");
            break;
        case 3:
            storage_price.setText("4700");
            break;
        case 4:
            storage_price.setText("3500");
            break;
        case 5:
            storage_price.setText("5700");
            break;
    }
}
```

Combo box for choosing Storage.



```java
private void btt_goBackActionPerformed(java.awt.event.ActionEvent evt) {
    customer_Landing cus_landd = new customer_Landing();
    cus_landd.setVisible(true);
    cus_landd.setLocationRelativeTo(null);
    this.dispose();
}
```

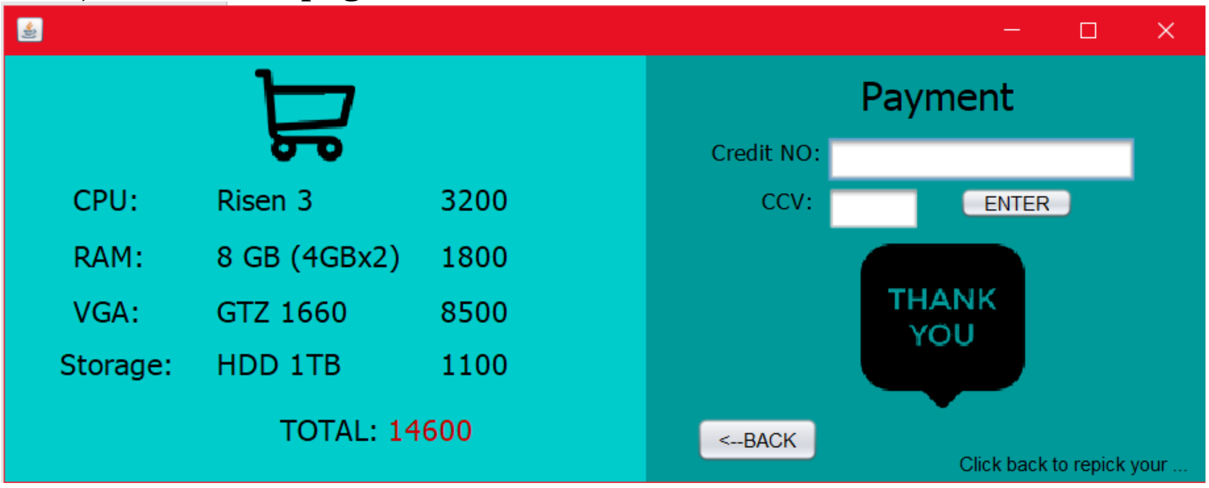
Go back button



```java
private void btt_to_placeOrderActionPerformed(java.awt.event.ActionEvent evt) {
    parts.add(Integer.parseInt(cpu_price.getText()));
    parts.add(Integer.parseInt(ram_price.getText()));
    parts.add(Integer.parseInt(vga_price.getText()));
    parts.add(Integer.parseInt(storage_price.getText()));
    //open placeOrder page and pass arralist
    placeOrder placeOrder_page = new placeOrder();
    //pass arraylist
    placeOrder_page.orderedParts = parts;
    //pass id
    placeOrder_page.user_id_order = user_id_order+placeOrder_page.user_id_order;
    //set jlable items in placeOrder page
    placeOrder_page.lbl_cpu.setText(cmb_cpu.getSelectedItem().toString());
    placeOrder_page.lbl_ram.setText(cmb_ram.getSelectedItem().toString());
    placeOrder_page.lbl_vga.setText(cmb_vga.getSelectedItem().toString());
    placeOrder_page.lbl_storage.setText(cmb_storage.getSelectedItem().toString());
    //set jlable price in placeOrder page
    placeOrder_page.lbl_cpu_price.setText(parts.get(0).toString());
    placeOrder_page.lbl_ram_price.setText(parts.get(1).toString());
    placeOrder_page.lbl_vga_price.setText(parts.get(2).toString());
    placeOrder_page.lbl_storage_price.setText(parts.get(3).toString());
    //calculate and set jLable total price
    int total_price = 0;
    for(int i=0;i<=3;i++){
        total_price = parts.get(i)+total_price;
    }
    System.out.println(total_price);
    placeOrder_page.lbl_total_price.setText(Integer.toString(total_price));
    //open and config jframe
    placeOrder_page.setVisible(true);
    placeOrder_page.setLocationRelativeTo(null);
    placeOrder_page.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    this.dispose();
}
```

Next(to_placeOrder) button will add all of the parts to ArrayList, then set the jLable on the next page to the same items selected.

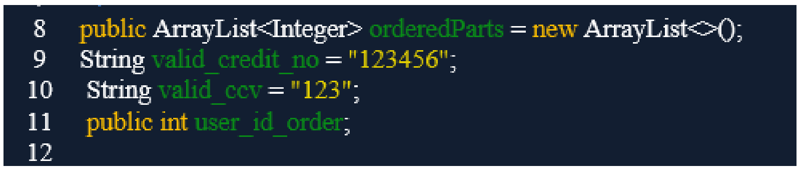
4.)Place order page



CPU: Risen 3 3200
RAM: 8 GB (4GBx2) 1800
VGA: GTZ 1660 8500
Storage: HDD 1TB 1100
TOTAL: 14600

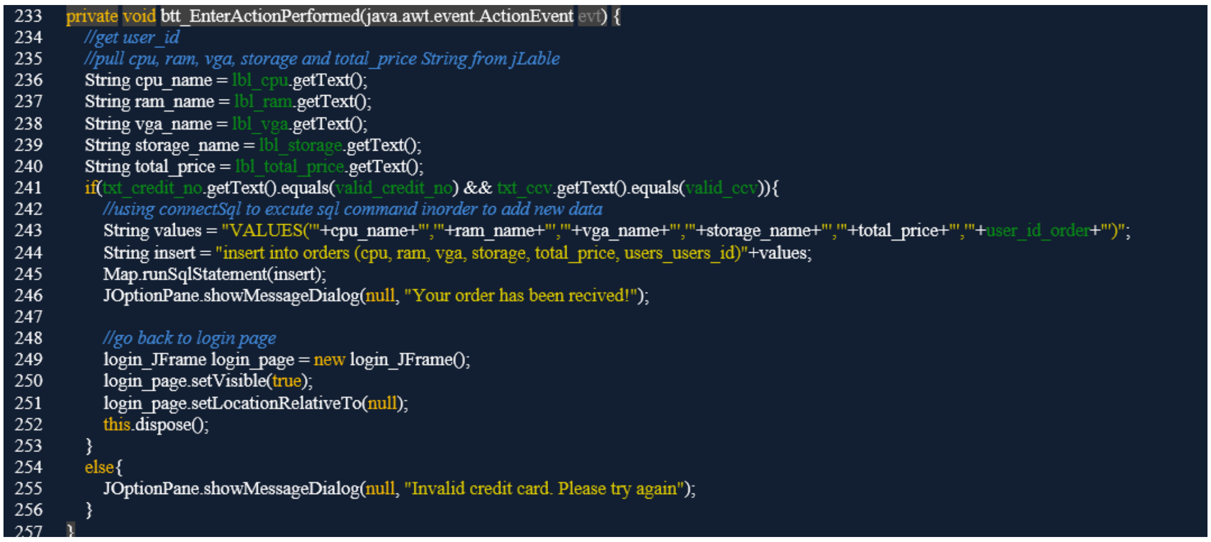
Payment
Credit NO:
CCV:
Click back to repick your ...

User can check their ordered parts(on the right) and fill in the payment information (default is 123456, 123).



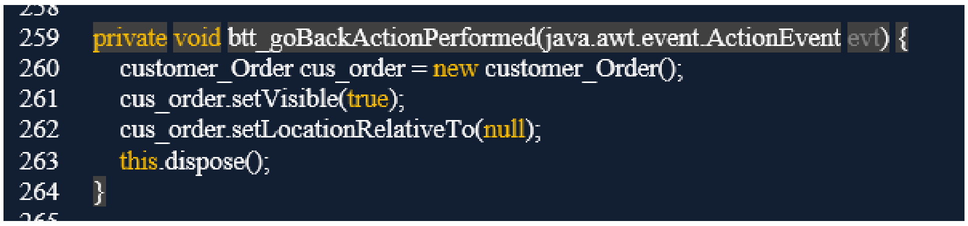
```java
public ArrayList<Integer> orderedParts = new ArrayList<>();
String valid_credit_no = "123456";
 String valid_ccv = "123";
 public int user_id_order;
```

Declared variables at the top



```java
private void btt_EnterActionPerformed(java.awt.event.ActionEvent evt) {
  //get user_id
  //pull cpu, ram, vga, storage and total_price String from jLable
  String cpu_name = lbl_cpu.getText();
  String ram_name = lbl_ram.getText();
  String vga_name = lbl_vga.getText();
  String storage_name = lbl_storage.getText();
  String total_price = lbl_total_price.getText();
  if(txt_credit_no.getText().equals(valid_credit_no) && txt_ccv.getText().equals(valid_ccv)){
    //using connectSql to excute sql command inorder to add new data
    String values = "VALUES('"+cpu_name+"','"+ram_name+"','"+vga_name+"','"+storage_name+"','"+total_price+"','"+user_id_order+"')";
    String insert = "insert into orders (cpu, ram, vga, storage, total_price, users_users_id)"+values;
    Map.runSqlStatement(insert);
    JOptionPane.showMessageDialog(null, "Your order has been recived!");

    //go back to login page
    login_JFrame login_page = new login_JFrame();
    login_page.setVisible(true);
    login_page.setLocationRelativeTo(null);
    this.dispose();
  }
  else{
    JOptionPane.showMessageDialog(null, "Invalid credit card. Please try again");
  }
}
```

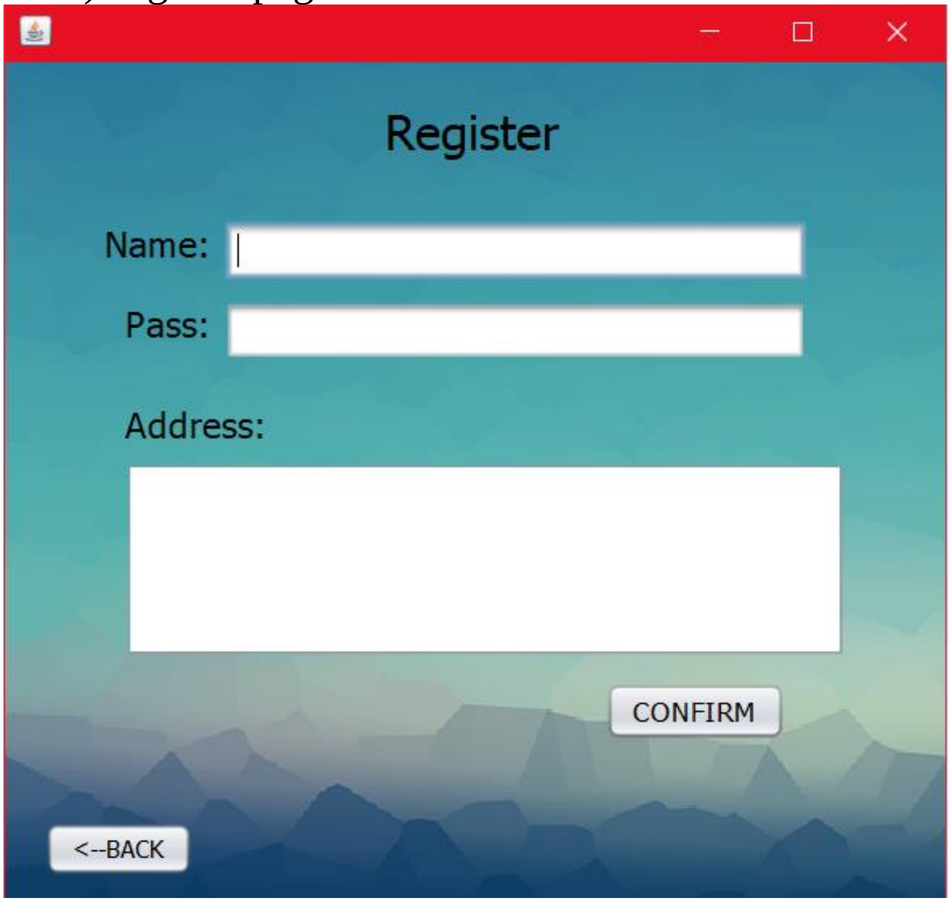
Enter button will first validate the credit number, then the ordered parts will be inserted into MySQL database.



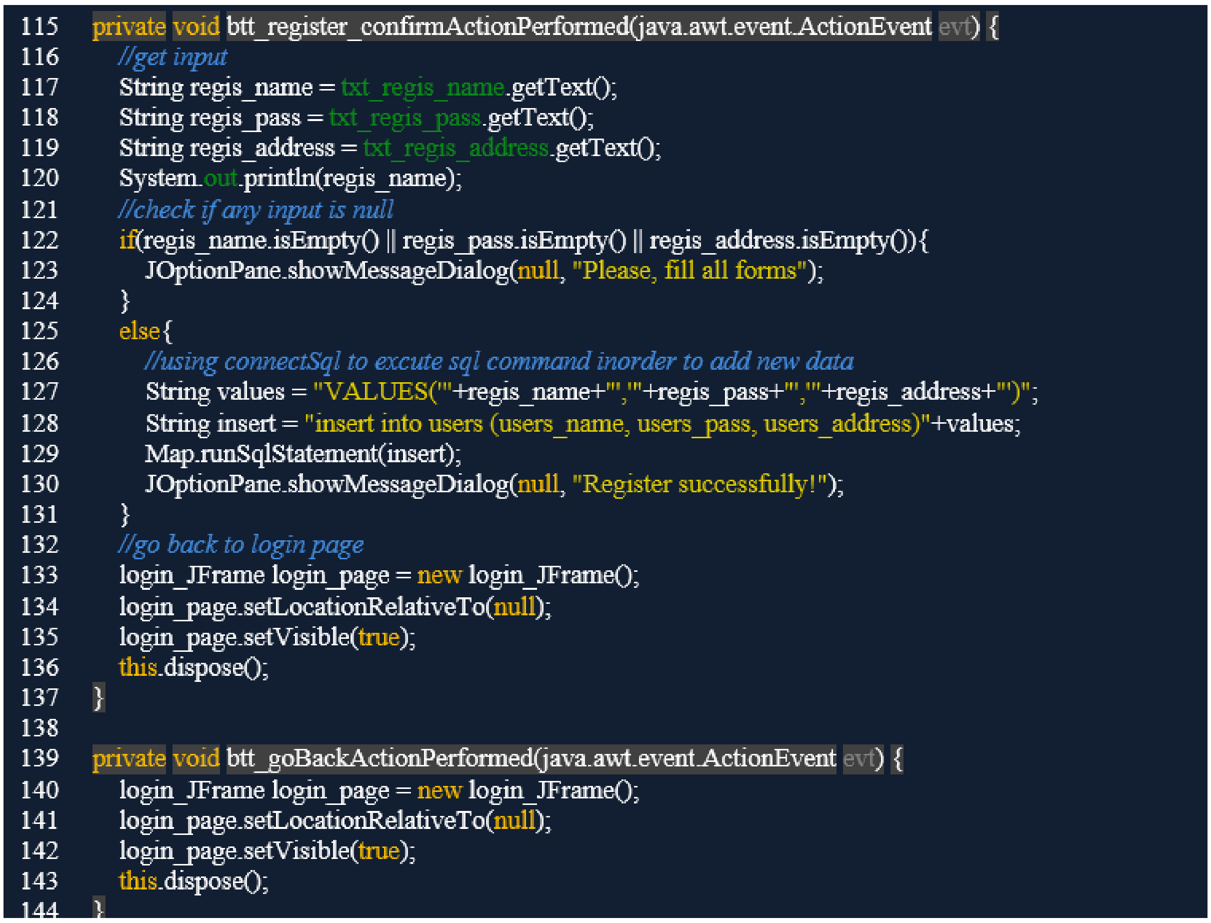
```
private void btt_goBackActionPerformed(java.awt.event.ActionEvent evt) {
    customer_Order cus_order = new customer_Order();
    cus_order.setVisible(true);
    cus_order.setLocationRelativeTo(null);
    this.dispose();
}
```

Go back button.

5.)Register page



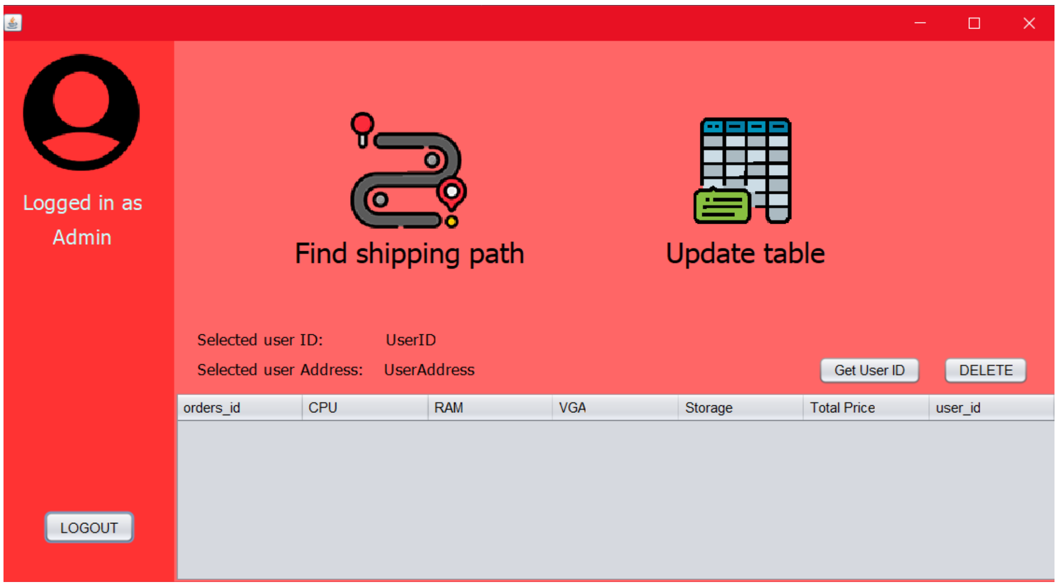
First time user will be registering their ID here. All of these info will be stored in MySQL database.



```java
private void btt_register_confirmActionPerformed(java.awt.event.ActionEvent evt) {
    //get input
    String regis_name = txt_regis_name.getText();
    String regis_pass = txt_regis_pass.getText();
    String regis_address = txt_regis_address.getText();
    System.out.println(regis_name);
    //check if any input is null
    if(regis_name.isEmpty() || regis_pass.isEmpty() || regis_address.isEmpty()){
        JOptionPane.showMessageDialog(null, "Please, fill all forms");
    }
    else{
        //using connectSql to excute sql command inorder to add new data
        String values = "VALUES('"+regis_name+"','"+regis_pass+"','"+regis_address+"')";
        String insert = "insert into users (users_name, users_pass, users_address)"+values;
        Map.runSqlStatement(insert);
        JOptionPane.showMessageDialog(null, "Register successfully!");
    }
    //go back to login page
    login_JFrame login_page = new login_JFrame();
    login_page.setLocationRelativeTo(null);
    login_page.setVisible(true);
    this.dispose();
}

private void btt_goBackActionPerformed(java.awt.event.ActionEvent evt) {
    login_JFrame login_page = new login_JFrame();
    login_page.setLocationRelativeTo(null);
    login_page.setVisible(true);
    this.dispose();
}
```

Confirm button will take all of the input and store them in the MySQL database. And go back button will simply take you back to the login page.

6.) Admin page

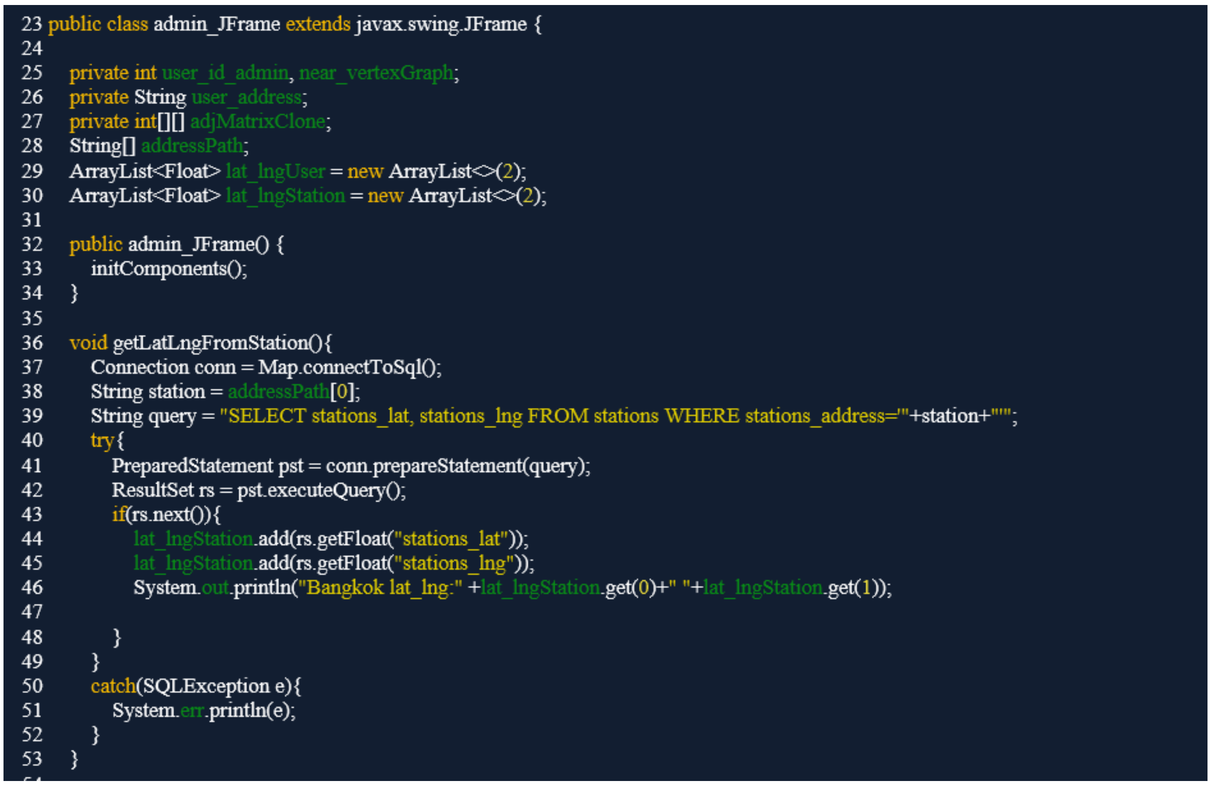


Selected user ID: UserID
Selected user Address: UserAddress

orders_id	CPU	RAM	VGA	Storage	Total Price	user_id

In admin page, admin can update orders table, get user ID and Address, delete order and find the shortest delivery path using the existing delivering route that pass through 15 existing stores.

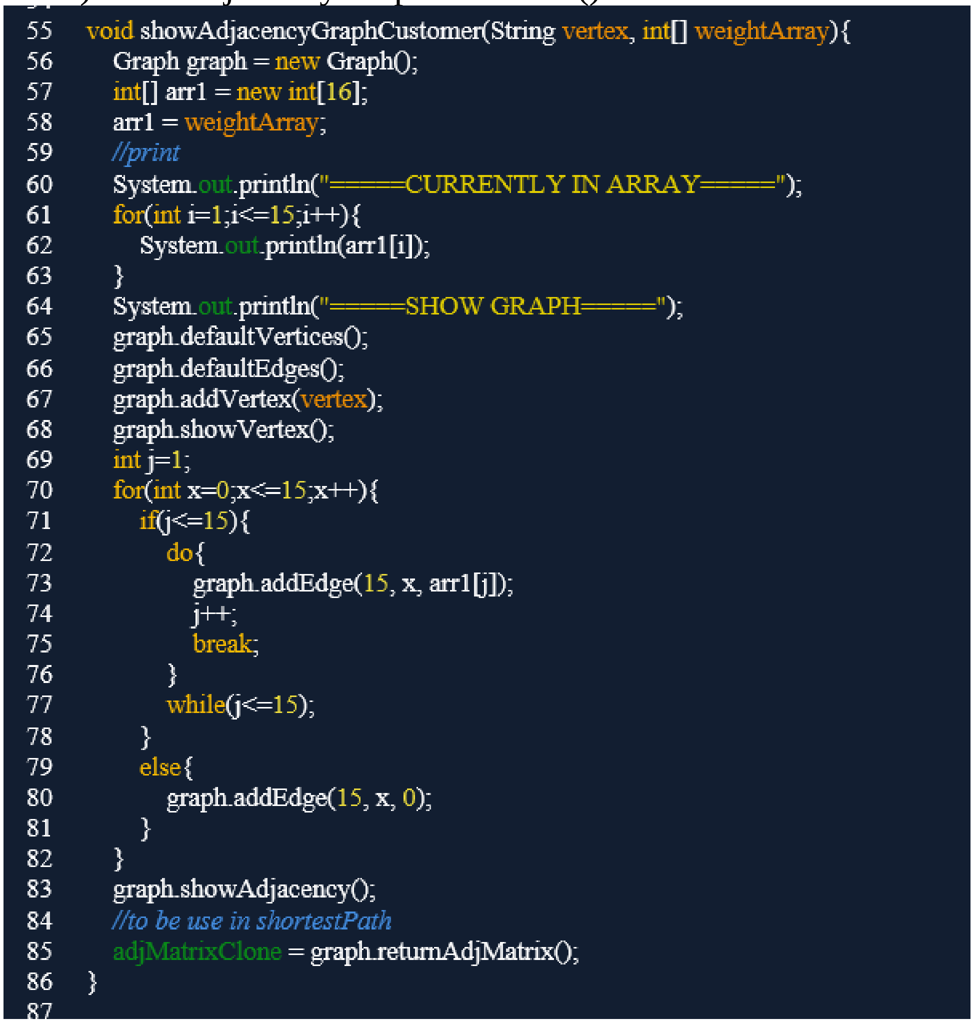
1.) getLatLngFromStation()



```java
public class admin_JFrame extends javax.swing.JFrame {

  private int user_id_admin, near_vertexGraph;
  private String user_address;
  private int[][] adjMatrixClone;
  String[] addressPath;
  ArrayList<Float> lat_lngUser = new ArrayList<>(2);
  ArrayList<Float> lat_lngStation = new ArrayList<>(2);

  public admin_JFrame() {
    initComponents();
  }

  void getLatLngFromStation(){
    Connection conn = Map.connectToSql();
    String station = addressPath[0];
    String query = "SELECT stations_lat, stations_lng FROM stations WHERE stations_address='"+station+"'";
    try{
      PreparedStatement pst = conn.prepareStatement(query);
      ResultSet rs = pst.executeQuery();
      if(rs.next()){
        lat_lngStation.add(rs.getFloat("stations_lat"));
        lat_lngStation.add(rs.getFloat("stations_lng"));
        System.out.println("Bangkok lat_lng:" +lat_lngStation.get(0)+" "+lat_lngStation.get(1));

      }
    }
    catch(SQLException e){
      System.err.println(e);
    }
  }
```

At the top of the code there're some variables and getLatLngFromStation() method that will get latitude and longitude of "Bangkok" (use Bangkok as the origin point because we imagine the PC build in Bangkok the send out to all other stores(nodes) on the delivery route) from MySQL database.

2.) showAdjacencyGraphCustomer()



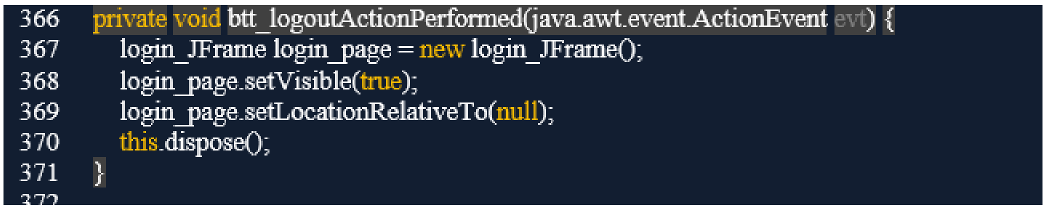
```java
void showAdjacencyGraphCustomer(String vertex, int[] weightArray){
    Graph graph = new Graph();
    int[] arr1 = new int[16];
    arr1 = weightArray;
    //print
    System.out.println("=====CURRENTLY IN ARRAY=====");
    for(int i=1;i<=15;i++){
        System.out.println(arr1[i]);
    }
    System.out.println("=====SHOW GRAPH=====");
    graph.defaultVertices();
    graph.defaultEdges();
    graph.addVertex(vertex);
    graph.showVertex();
    int j=1;
    for(int x=0;x<=15;x++){
        if(j<=15){
            do{
                graph.addEdge(15, x, arr1[j]);
                j++;
                break;
            }
            while(j<=15);
        }
        else{
            graph.addEdge(15, x, 0);
        }
    }
    graph.showAdjacency();
    //to be use in shortestPath
    adjMatrixClone = graph.returnAdjMatrix();
}
```

Similar to showAdjacency() in Graph class.

3.) getNearestStoreAdmin()

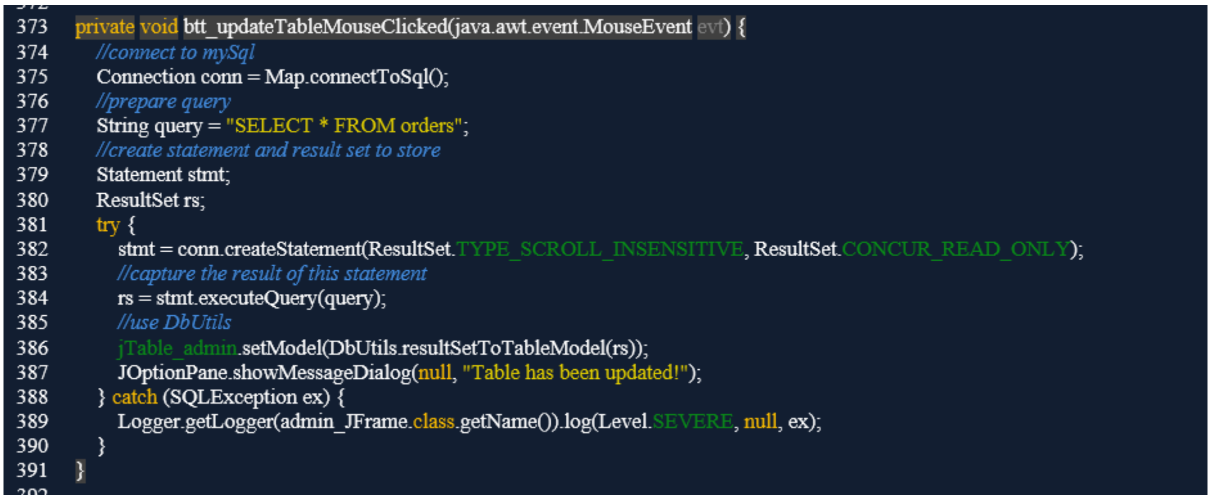
```
private void getNearestStoreAdmin(int user_id) throws SQLException{
    //calculate shortest path using dijkstra's algorithm
    int[] allDist, allDistNoZero, distArray;
    int vertex = 0, vertexGraph;
    String acquiredAddress = null;

    //return dist array
    distArray = Map.getDistArraySql(user_id_admin);
    //show adjacency representation and set adjmatrixclone
    showAdjacencyGraphCustomer(user_address, distArray);

    //find nearest node to user node
    ShortestPath spt = new ShortestPath();
    allDist = spt.dijkstra(adjMatrixClone, 15);

    //sort min to get km of nearest store
    //remove zero(distance from source to itself) before get minimum dist
    allDistNoZero = Arrays.copyOf(allDist, 15);
    //get minimum distance(nearest node)
    int minDist = Arrays.stream(allDistNoZero).min().getAsInt();
    //connect to sql to get the name of the address according to its distance
    Connection conn = Map.connectToSql();

    //get vertex from the distance equal to the minimum distance
    String queryVertex = "SELECT vertex FROM distances WHERE dist="+minDist+" AND user_id="+user_id;
    PreparedStatement stmtVertex;
    stmtVertex = conn.prepareStatement(queryVertex);
    ResultSet rsVertex = stmtVertex.executeQuery(queryVertex);
    if(rsVertex.next()){
        vertex = rsVertex.getInt("vertex");
        System.out.println("GOT VERTEX! "+vertex);
    }
    //minus 1 to get the vertex accordingly
    vertexGraph = vertex-1;
    System.out.println("Vertex(in Graph): "+vertexGraph);
    System.out.println("Vertex(in SQL): "+vertex);

    //get address from the vertex acquired
    String queryAddress = "SELECT stations_address FROM stations WHERE stations_id="+vertex;
    PreparedStatement stmtAddress = conn.prepareStatement(queryAddress);
    ResultSet rsAddress = stmtAddress.executeQuery(queryAddress);
    if(rsAddress.next()){
        acquiredAddress = rsAddress.getString("stations_address");
    }
    System.out.println("Vertex: "+vertex+" Station name: "+acquiredAddress);
    near_vertexGraph = vertexGraph;
}
```

This method is also similar to the getNearestStore() in customer landing page. We this to know which node is the closest to user's node. There's another way of doing this since we have a list of all stations location leading to the user's location (see addressPath[]) we can just get an element in the last index.



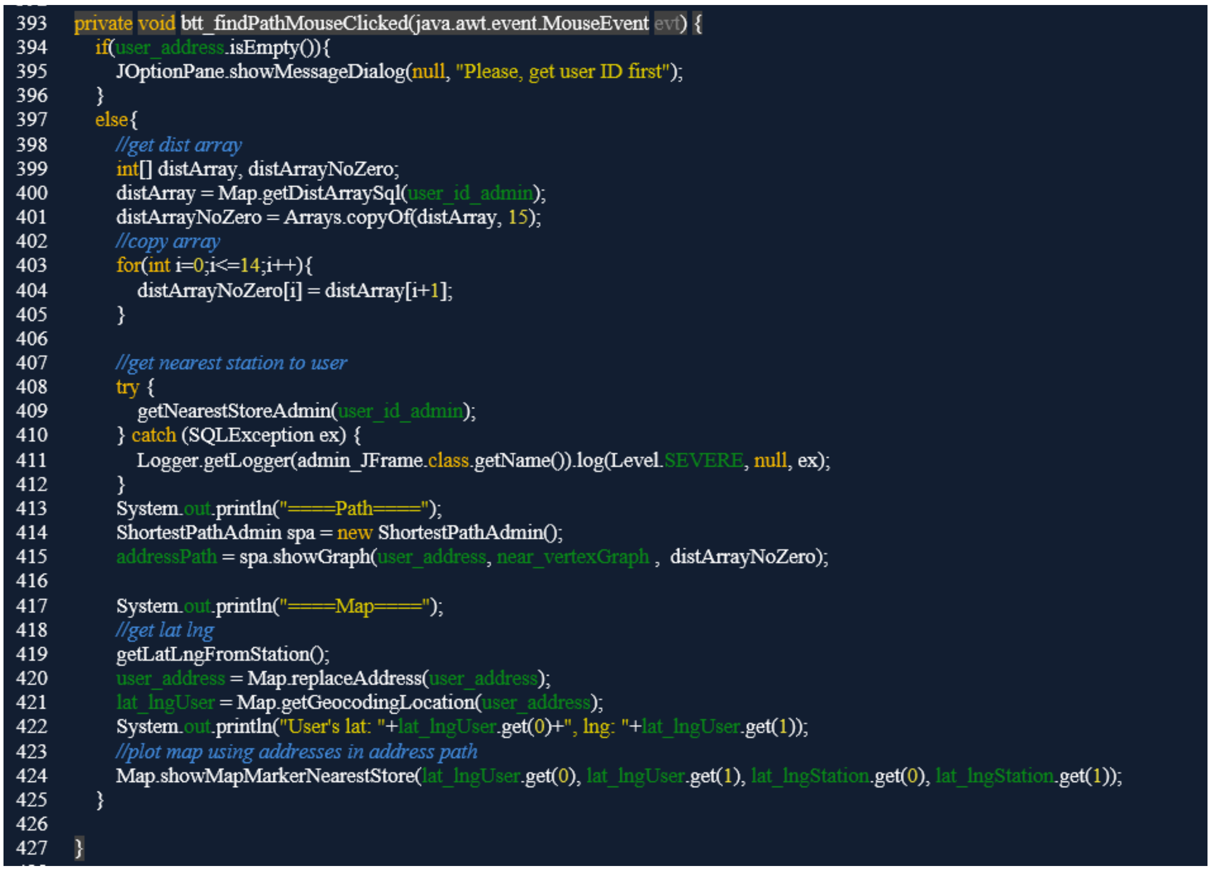
```java
private void btt_logoutActionPerformed(java.awt.event.ActionEvent evt) {
    login_JFrame login_page = new login_JFrame();
    login_page.setVisible(true);
    login_page.setLocationRelativeTo(null);
    this.dispose();
}
```

Logout button.



```java
private void btt_updateTableMouseClicked(java.awt.event.MouseEvent evt) {
    //connect to mySql
    Connection conn = Map.connectToSql();
    //prepare query
    String query = "SELECT * FROM orders";
    //create statement and result set to store
    Statement stmt;
    ResultSet rs;
    try {
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        //capture the result of this statement
        rs = stmt.executeQuery(query);
        //use DbUtils
        jTable_admin.setModel(DbUtils.resultSetToTableModel(rs));
        JOptionPane.showMessageDialog(null, "Table has been updated!");
    } catch (SQLException ex) {
        Logger.getLogger(admin_JFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Update button will simply take all data in orders table from MySQL database and then transfer them into the jTable.



```java
private void btt_findPathMouseClicked(java.awt.event.MouseEvent evt) {
    if(user_address.isEmpty()){
        JOptionPane.showMessageDialog(null, "Please, get user ID first");
    }
    else{
        //get dist array
        int[] distArray, distArrayNoZero;
        distArray = Map.getDistArraySql(user_id_admin);
        distArrayNoZero = Arrays.copyOf(distArray, 15);
        //copy array
        for(int i=0;i<=14;i++){
            distArrayNoZero[i] = distArray[i+1];
        }

        //get nearest station to user
        try {
            getNearestStoreAdmin(user_id_admin);
        } catch (SQLException ex) {
            Logger.getLogger(admin_JFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("=====Path=====");
        ShortestPathAdmin spa = new ShortestPathAdmin();
        addressPath = spa.showGraph(user_address, near_vertexGraph ,  distArrayNoZero);

        System.out.println("=====Map=====");
        //get lat lng
        getLatLngFromStation();
        user_address = Map.replaceAddress(user_address);
        lat_lngUser = Map.getGeocodingLocation(user_address);
        System.out.println("User's lat: "+lat_lngUser.get(0)+", lng: "+lat_lngUser.get(1));
        //plot map using addresses in address path
        Map.showMapMarkerNearestStore(lat_lngUser.get(0), lat_lngUser.get(1), lat_lngStation.get(0), lat_lngStation.get(1));
    }

}
```

Find path button first get distArray[] containing user’s distance to all other nodes(stores) that we got from distance API. Then run getNearestStoreAdmin(). After that we’ll get addressPath[] to be use in getLatLngFromStation(). Lastly the showMapMarkerNearestStroe() in Map class will run showing the map with marker placed at the place where PC is built(Bangkok) and where PC must be shipped to(user’s location).

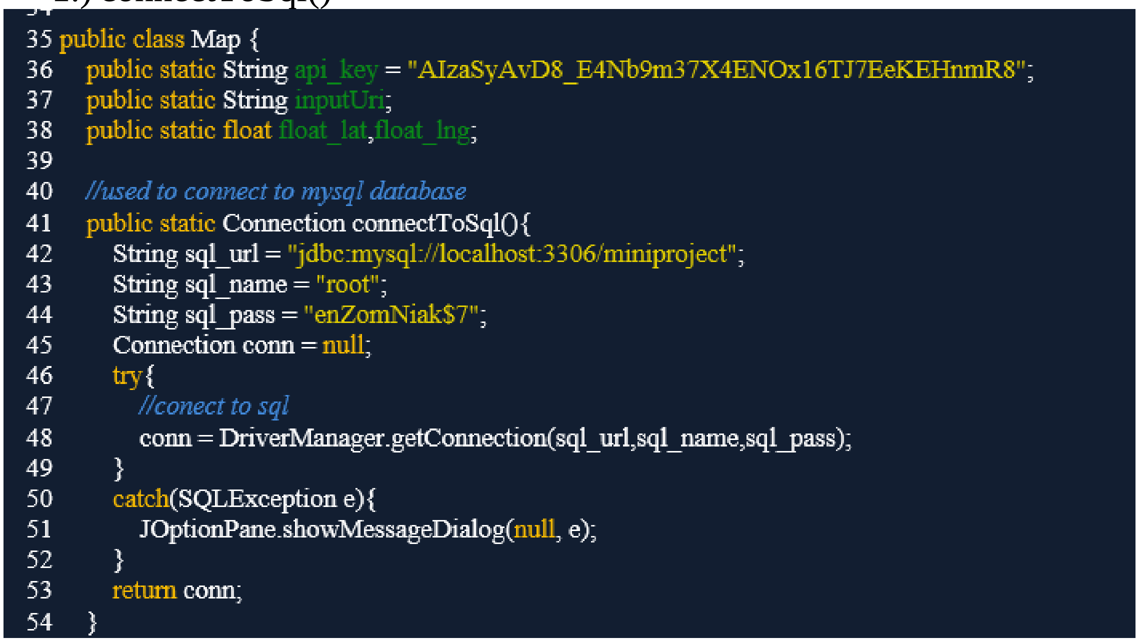
There’s limitation in using the JXMapviewer2 because of how we have to create an instance for each marker it can be hard to have marker placed on each and every node leading to the user’s node. And so, in this case we decided to placed only two markers (Bangkok to user) and have the shortest path show in the JOptionPane instead.

Classes

Map

This class is the main class that contains many frequently use methods.

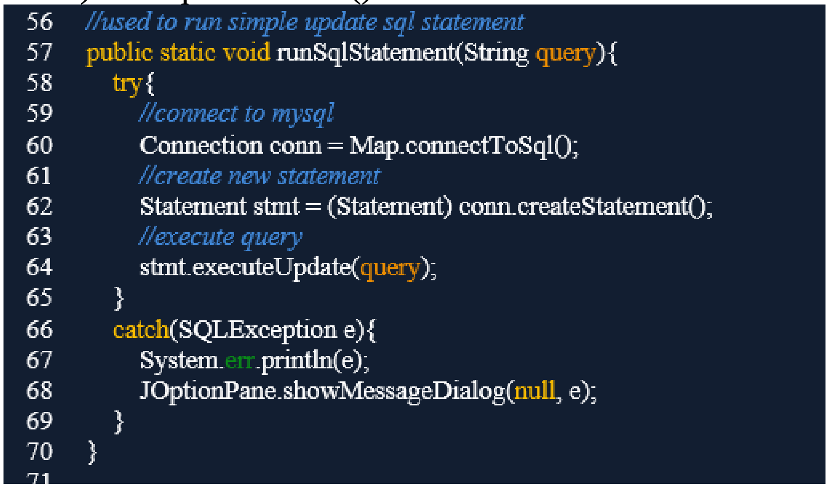
1.) connectToSql()



```
public class Map {
    public static String api_key = "AIzaSyAvD8_E4Nb9m37X4ENOx16TJ7EeKEHnmR8";
    public static String inputUri;
    public static float float_lat,float_lng;

    //used to connect to mysql database
    public static Connection connectToSql(){
        String sql_url = "jdbc:mysql://localhost:3306/miniproject";
        String sql_name = "root";
        String sql_pass = "enZomNiak$7";
        Connection conn = null;
        try{
            //conect to sql
            conn = DriverManager.getConnection(sql_url,sql_name,sql_pass);
        }
        catch(SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
        return conn;
    }
```

connectToSql method will return a connection to MySQL.

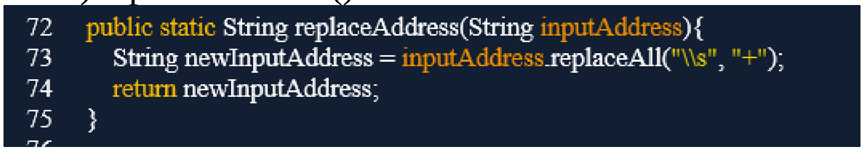
2.) runSqlStatement()



```
    //used to run simple update sql statement
    public static void runSqlStatement(String query){
        try{
            //connect to mysql
            Connection conn = Map.connectToSql();
            //create new statement
            Statement stmt = (Statement) conn.createStatement();
            //execute query
            stmt.executeUpdate(query);
        }
        catch(SQLException e){
            System.err.println(e);
            JOptionPane.showMessageDialog(null, e);
        }
    }
```

Used to run basic SQL statement like update, insert, or delete.

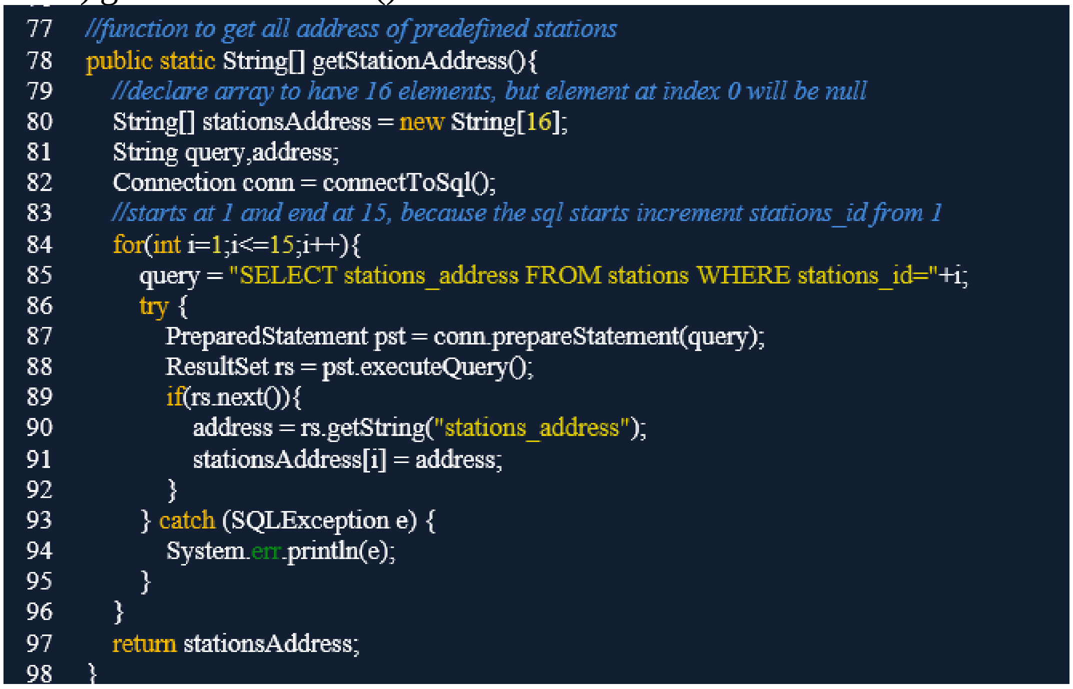
3.) replaceAddress()



```java
public static String replaceAddress(String inputAddress){
    String newInputAddress = inputAddress.replaceAll("\\s", "+");
    return newInputAddress;
}
```

Simple replace function.

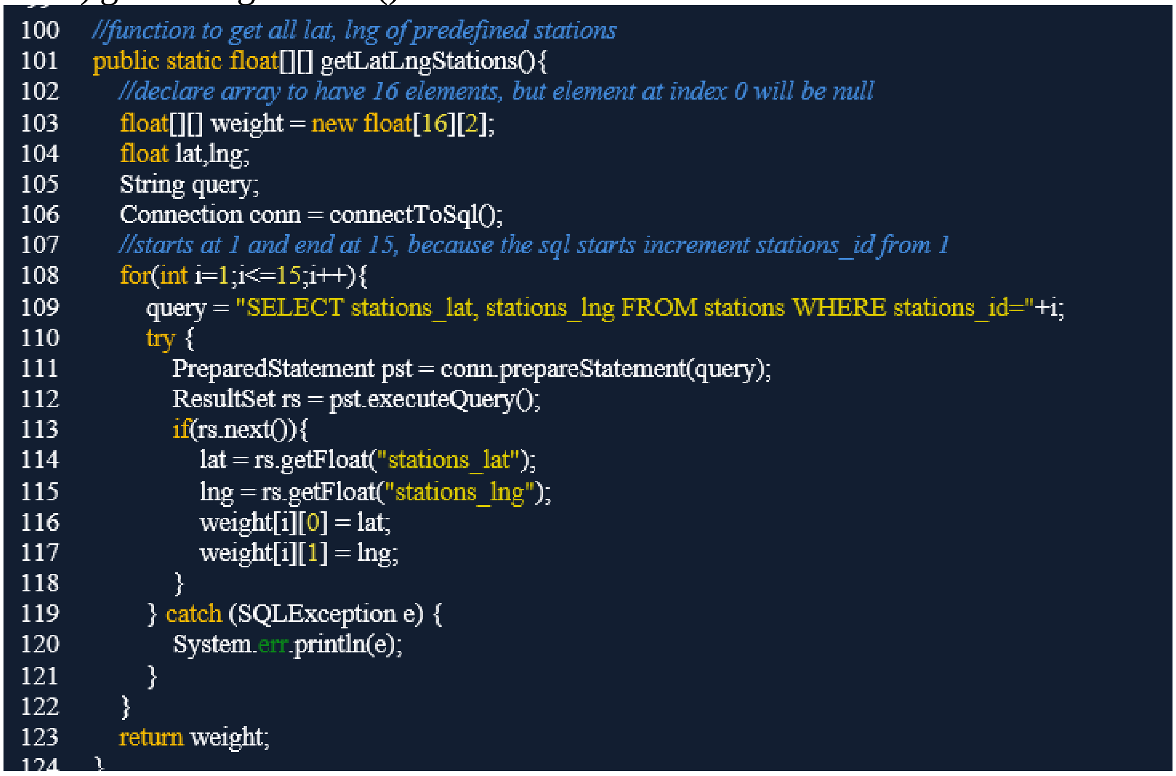
4.) getStationAddress()



```java
//function to get all address of predefined stations
public static String[] getStationAddress(){
    //declare array to have 16 elements, but element at index 0 will be null
    String[] stationsAddress = new String[16];
    String query,address;
    Connection conn = connectToSql();
    //starts at 1 and end at 15, because the sql starts increment stations_id from 1
    for(int i=1;i<=15;i++){
        query = "SELECT stations_address FROM stations WHERE stations_id="+i;
        try {
            PreparedStatement pst = conn.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            if(rs.next()){
                address = rs.getString("stations_address");
                stationsAddress[i] = address;
            }
        } catch (SQLException e) {
            System.err.println(e);
        }
    }
    return stationsAddress;
}
```

This method will get a station address from MySQL

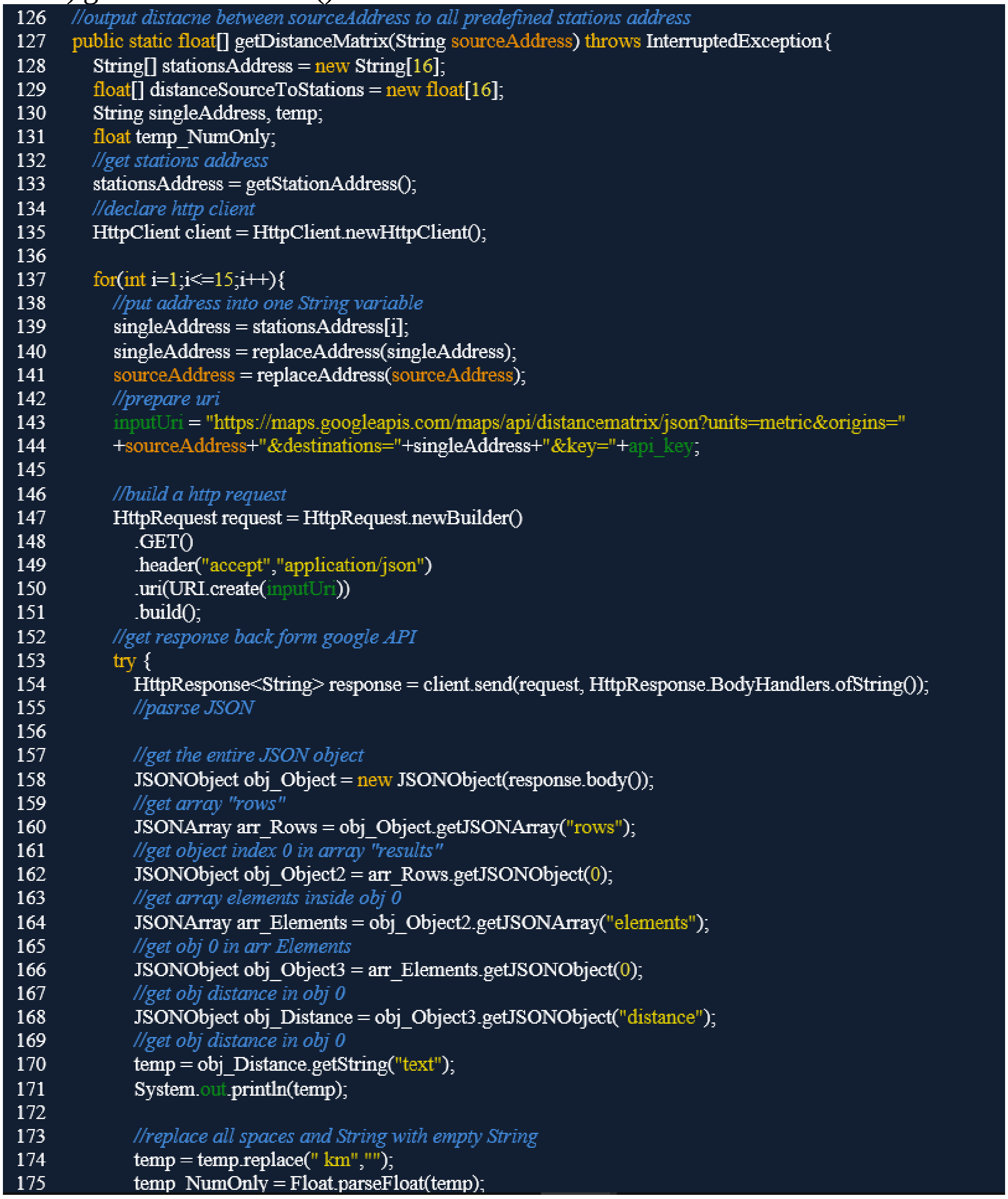
5.) getLatLngStations()



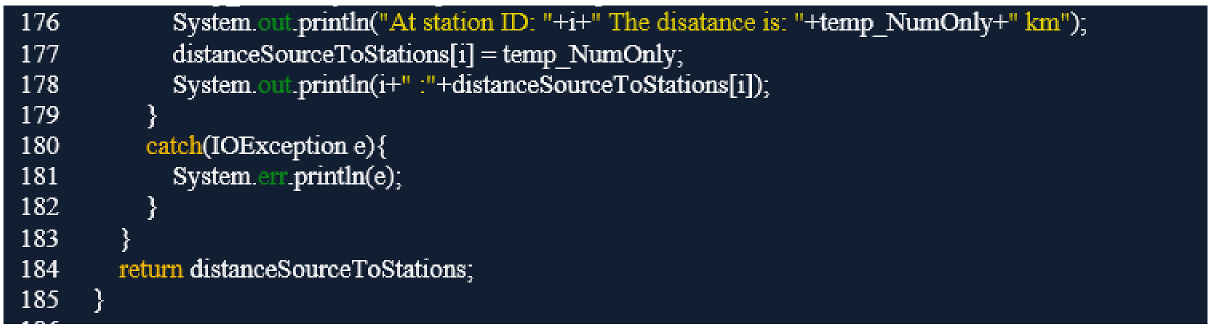
```java
//function to get all lat, lng of predefined stations
public static float[][] getLatLngStations(){
    //declare array to have 16 elements, but element at index 0 will be null
    float[][] weight = new float[16][2];
    float lat,lng;
    String query;
    Connection conn = connectToSql();
    //starts at 1 and end at 15, because the sql starts increment stations_id from 1
    for(int i=1;i<=15;i++){
        query = "SELECT stations_lat, stations_lng FROM stations WHERE stations_id="+i;
        try {
            PreparedStatement pst = conn.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            if(rs.next()){
                lat = rs.getFloat("stations_lat");
                lng = rs.getFloat("stations_lng");
                weight[i][0] = lat;
                weight[i][1] = lng;
            }
        } catch (SQLException e) {
            System.err.println(e);
        }
    }
    return weight;
}
```

This method will get latitude and longitude of all stations(store)

6.) getDistanceMatrix()



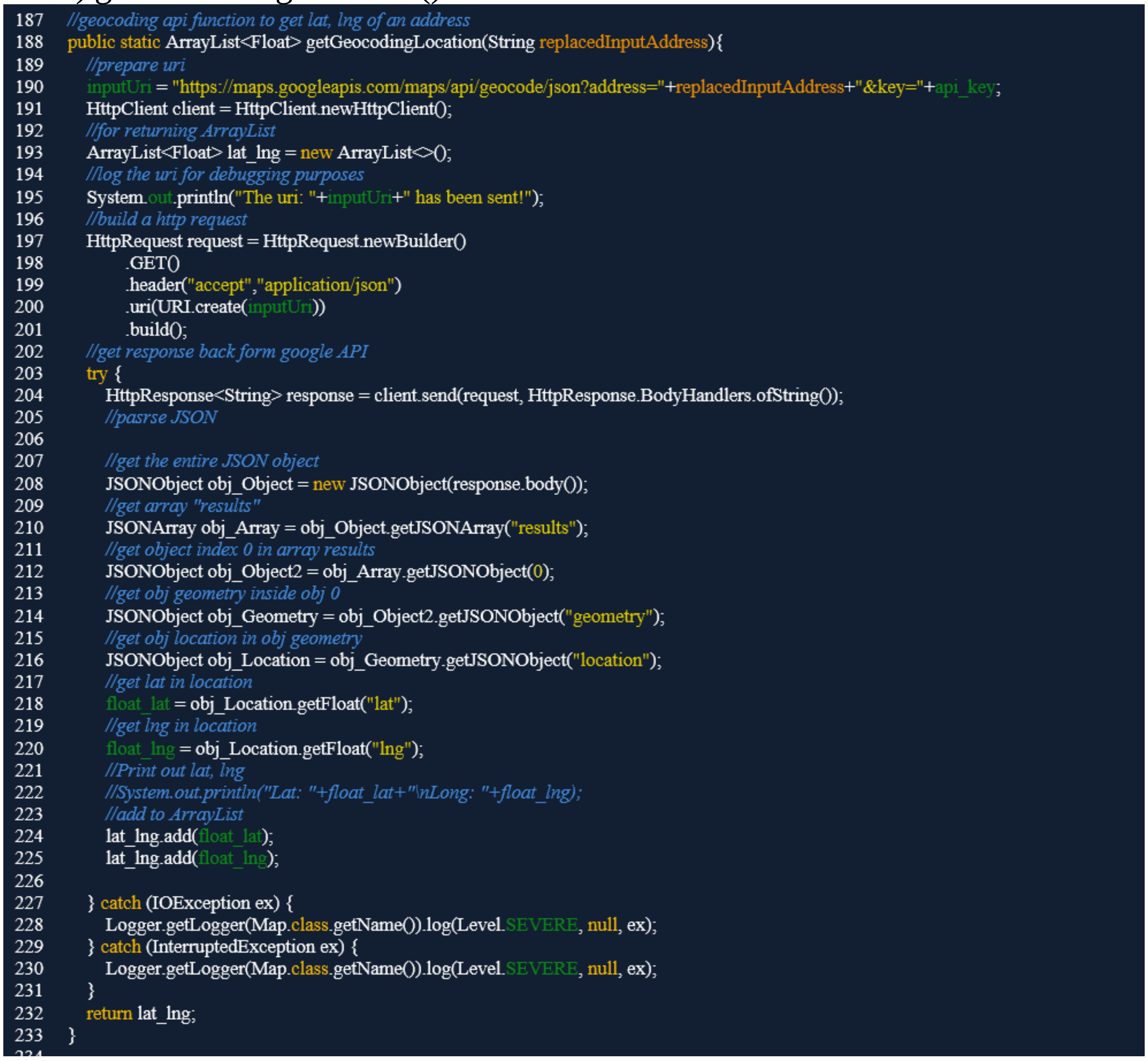
```
//output distacne between sourceAddress to all predefined stations address
public static float[] getDistanceMatrix(String sourceAddress) throws InterruptedException{
    String[] stationsAddress = new String[16];
    float[] distanceSourceToStations = new float[16];
    String singleAddress, temp;
    float temp_NumOnly;
    //get stations address
    stationsAddress = getStationAddress();
    //declare http client
    HttpClient client = HttpClient.newHttpClient();

    for(int i=1;i<=15;i++){
        //put address into one String variable
        singleAddress = stationsAddress[i];
        singleAddress = replaceAddress(singleAddress);
        sourceAddress = replaceAddress(sourceAddress);
        //prepare uri
        inputUri = "https://maps.googleapis.com/maps/api/distancematrix/json?units=metric&origins="
        +sourceAddress+"&destinations="+singleAddress+"&key="+api_key;

        //build a http request
        HttpRequest request = HttpRequest.newBuilder()
            .GET()
            .header("accept","application/json")
            .uri(URI.create(inputUri))
            .build();
        //get response back form google API
        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
            //pasrse JSON

            //get the entire JSON object
            JSONObject obj_Object = new JSONObject(response.body());
            //get array "rows"
            JSONArray arr_Rows = obj_Object.getJSONArray("rows");
            //get object index 0 in array "results"
            JSONObject obj_Object2 = arr_Rows.getJSONObject(0);
            //get array elements inside obj 0
            JSONArray arr_Elements = obj_Object2.getJSONArray("elements");
            //get obj 0 in arr Elements
            JSONObject obj_Object3 = arr_Elements.getJSONObject(0);
            //get obj distance in obj 0
            JSONObject obj_Distance = obj_Object3.getJSONObject("distance");
            //get obj distance in obj 0
            temp = obj_Distance.getString("text");
            System.out.println(temp);

            //replace all spaces and String with empty String
            temp = temp.replace(" km","");
            temp_NumOnly = Float.parseFloat(temp);
```



```
                System.out.println("At station ID: "+i+" The disatance is: "+temp_NumOnly+" km");
                distanceSourceToStations[i] = temp_NumOnly;
                System.out.println(i+" :"+distanceSourceToStations[i]);
            }
            catch(IOException e){
                System.err.println(e);
            }
        }
        return distanceSourceToStations;
    }
```

This method will send a request to Google distance matrix API (compare distance between two places). It has 1 parameter (users address) It'll compare user's location with all other nodes (15 stores).

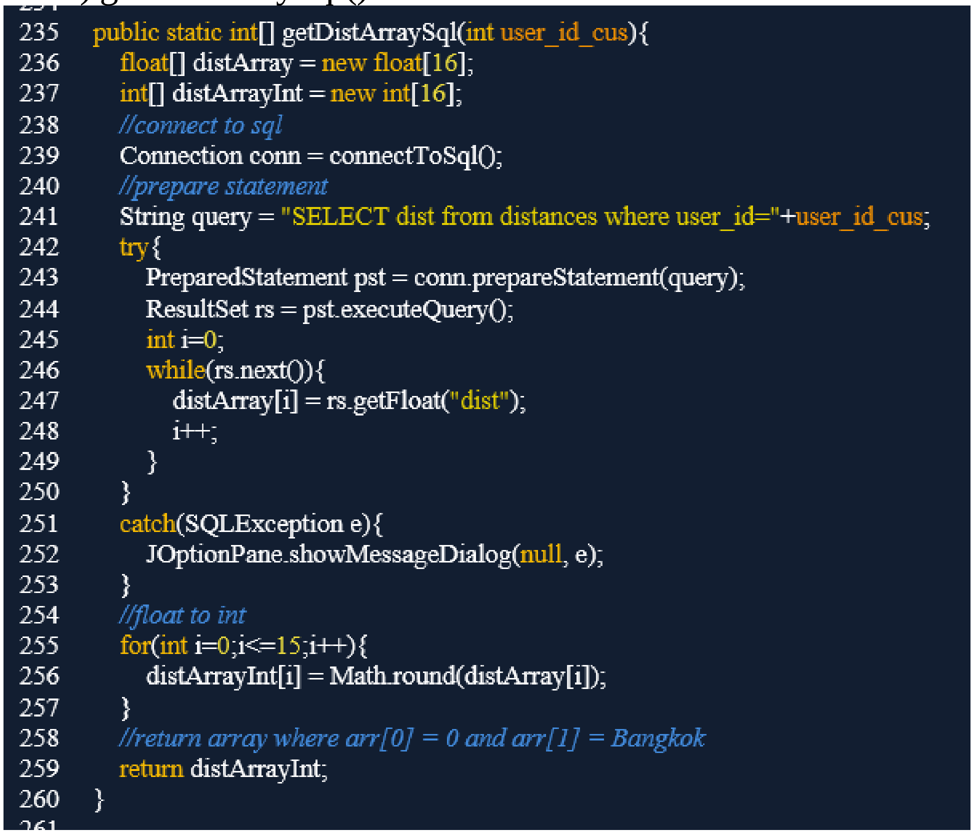
7.) getGeocodingLocation()



```
    //geocoding api function to get lat, lng of an address
    public static ArrayList<Float> getGeocodingLocation(String replacedInputAddress){
        //prepare uri
        inputUri = "https://maps.googleapis.com/maps/api/geocode/json?address="+replacedInputAddress+"&key="+api_key;
        HttpClient client = HttpClient.newHttpClient();
        //for returning ArrayList
        ArrayList<Float> lat_lng = new ArrayList<>();
        //log the uri for debugging purposes
        System.out.println("The uri: "+inputUri+" has been sent!");
        //build a http request
        HttpRequest request = HttpRequest.newBuilder()
                .GET()
                .header("accept","application/json")
                .uri(URI.create(inputUri))
                .build();
        //get response back form google API
        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
            //pasrse JSON

            //get the entire JSON object
            JSONObject obj_Object = new JSONObject(response.body());
            //get array "results"
            JSONArray obj_Array = obj_Object.getJSONArray("results");
            //get object index 0 in array results
            JSONObject obj_Object2 = obj_Array.getJSONObject(0);
            //get obj geometry inside obj 0
            JSONObject obj_Geometry = obj_Object2.getJSONObject("geometry");
            //get obj location in obj geometry
            JSONObject obj_Location = obj_Geometry.getJSONObject("location");
            //get lat in location
            float_lat = obj_Location.getFloat("lat");
            //get lng in location
            float_lng = obj_Location.getFloat("lng");
            //Print out lat, lng
            //System.out.println("Lat: "+float_lat+"\nLong: "+float_lng);
            //add to ArrayList
            lat_lng.add(float_lat);
            lat_lng.add(float_lng);

        } catch (IOException ex) {
            Logger.getLogger(Map.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InterruptedException ex) {
            Logger.getLogger(Map.class.getName()).log(Level.SEVERE, null, ex);
        }
        return lat_lng;
    }
```

This function will send request to Google geocoding API and get the latitude and longitude of the input address.

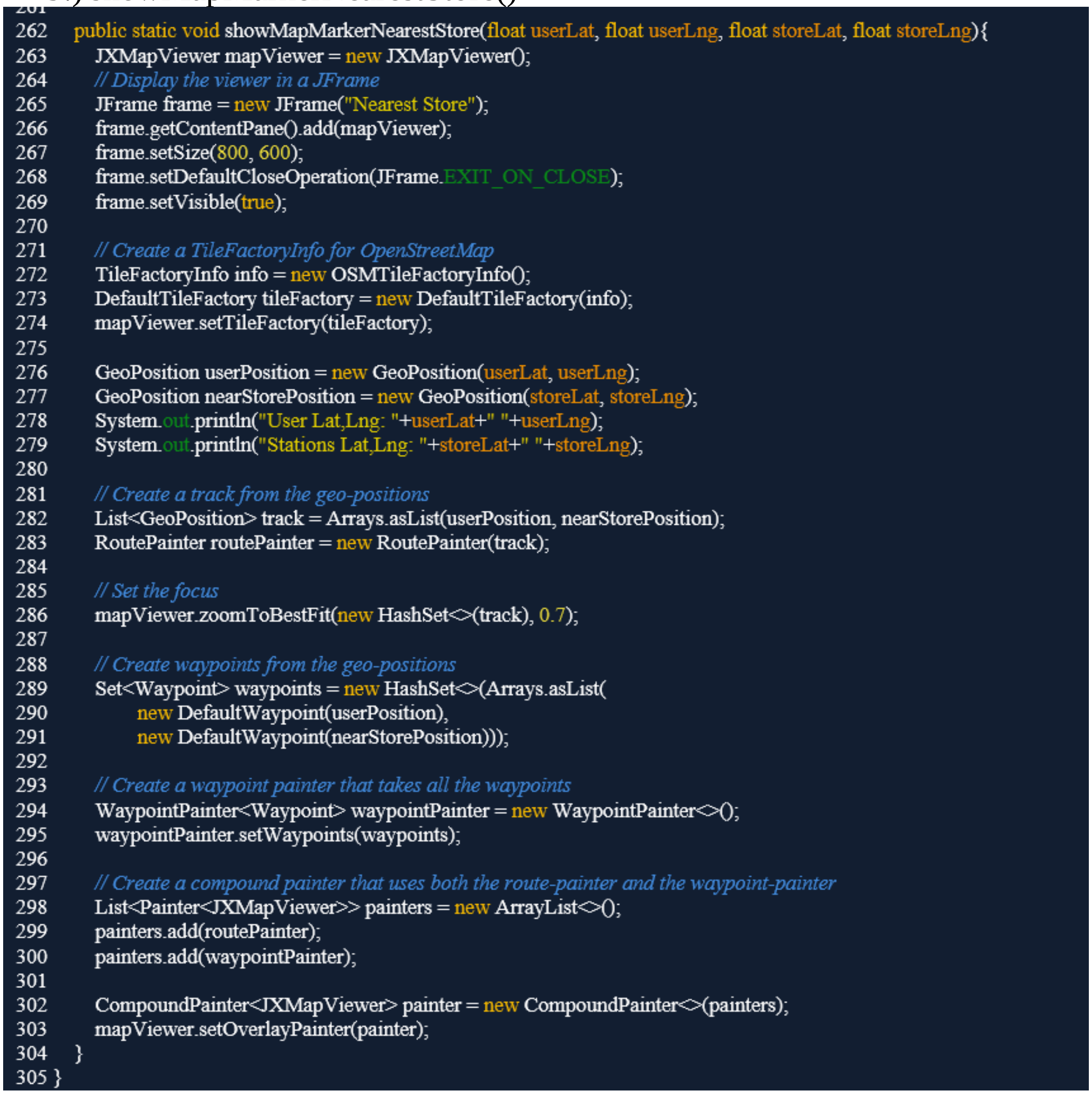
8.) getDistArraySql()



```java
public static int[] getDistArraySql(int user_id_cus){
    float[] distArray = new float[16];
    int[] distArrayInt = new int[16];
    //connect to sql
    Connection conn = connectToSql();
    //prepare statement
    String query = "SELECT dist from distances where user_id="+user_id_cus;
    try{
        PreparedStatement pst = conn.prepareStatement(query);
        ResultSet rs = pst.executeQuery();
        int i=0;
        while(rs.next()){
            distArray[i] = rs.getFloat("dist");
            i++;
        }
    }
    catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
    //float to int
    for(int i=0;i<=15;i++){
        distArrayInt[i] = Math.round(distArray[i]);
    }
    //return array where arr[0] = 0 and arr[1] = Bangkok
    return distArrayInt;
}
```

This method will get stations distances(to user) that store in MySQL.

9.) showMapMarkerNearestStore()

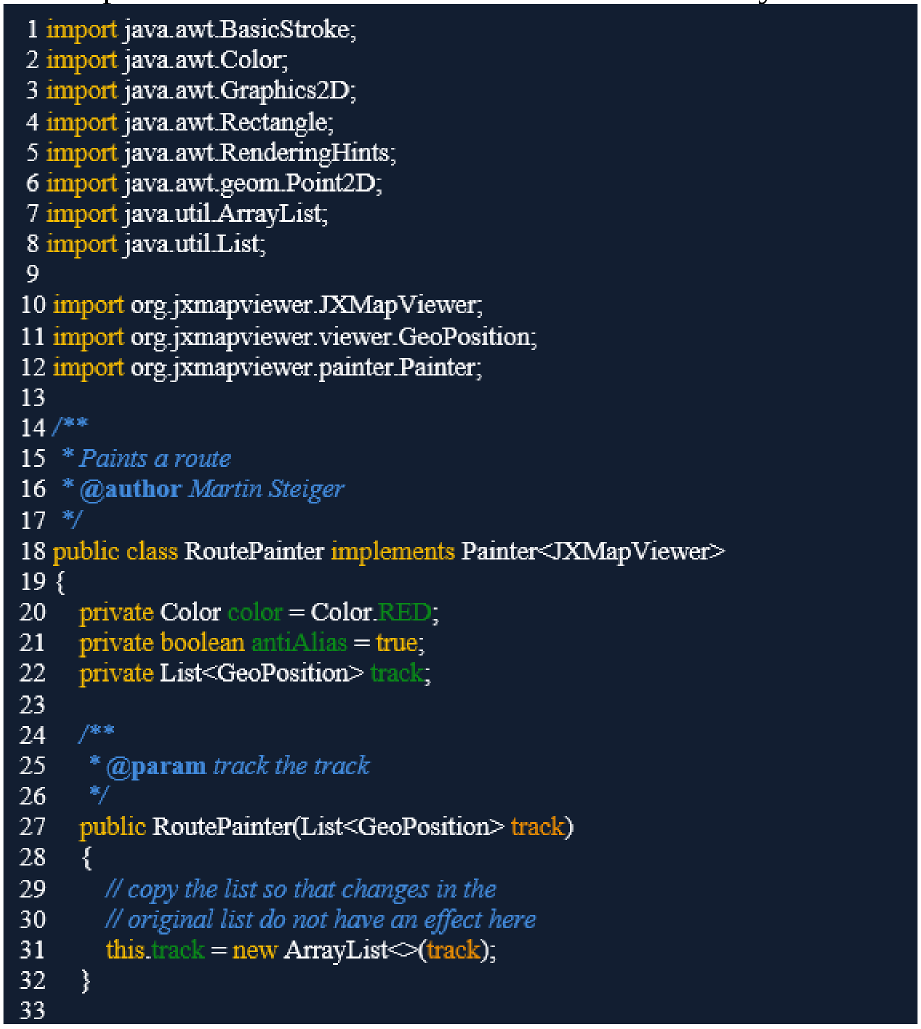


```
    public static void showMapMarkerNearestStore(float userLat, float userLng, float storeLat, float storeLng){
        JXMapViewer mapViewer = new JXMapViewer();
        // Display the viewer in a JFrame
        JFrame frame = new JFrame("Nearest Store");
        frame.getContentPane().add(mapViewer);
        frame.setSize(800, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        // Create a TileFactoryInfo for OpenStreetMap
        TileFactoryInfo info = new OSMTileFactoryInfo();
        DefaultTileFactory tileFactory = new DefaultTileFactory(info);
        mapViewer.setTileFactory(tileFactory);

        GeoPosition userPosition = new GeoPosition(userLat, userLng);
        GeoPosition nearStorePosition = new GeoPosition(storeLat, storeLng);
        System.out.println("User Lat,Lng: "+userLat+" "+userLng);
        System.out.println("Stations Lat,Lng: "+storeLat+" "+storeLng);

        // Create a track from the geo-positions
        List<GeoPosition> track = Arrays.asList(userPosition, nearStorePosition);
        RoutePainter routePainter = new RoutePainter(track);

        // Set the focus
        mapViewer.zoomToBestFit(new HashSet<>(track), 0.7);

        // Create waypoints from the geo-positions
        Set<Waypoint> waypoints = new HashSet<>(Arrays.asList(
                new DefaultWaypoint(userPosition),
                new DefaultWaypoint(nearStorePosition)));

        // Create a waypoint painter that takes all the waypoints
        WaypointPainter<Waypoint> waypointPainter = new WaypointPainter<>();
        waypointPainter.setWaypoints(waypoints);

        // Create a compound painter that uses both the route-painter and the waypoint-painter
        List<Painter<JXMapViewer>> painters = new ArrayList<>();
        painters.add(routePainter);
        painters.add(waypointPainter);

        CompoundPainter<JXMapViewer> painter = new CompoundPainter<>(painters);
        mapViewer.setOverlayPainter(painter);
    }
}
```

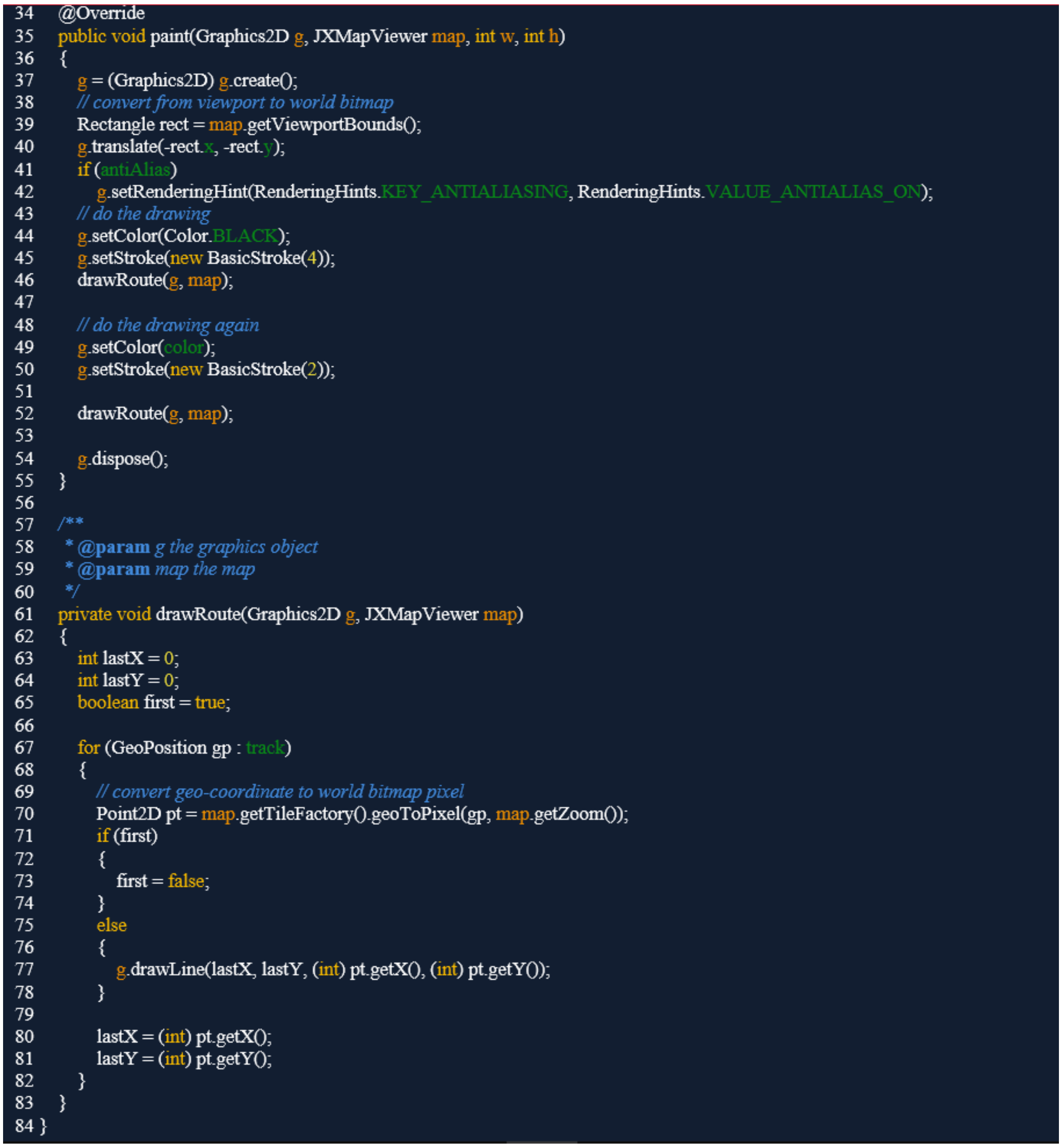
This function will print out a map with 2 markers that mark user's location and store's location.

RoutePainter

Route painter is a class that the author of the library share to be used as a guide.



```java
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.RenderingHints;
import java.awt.geom.Point2D;
import java.util.ArrayList;
import java.util.List;

import org.jxmapviewer.JXMapViewer;
import org.jxmapviewer.viewer.GeoPosition;
import org.jxmapviewer.painter.Painter;

/**
 * Paints a route
 * @author Martin Steiger
 */
public class RoutePainter implements Painter<JXMapViewer>
{
    private Color color = Color.RED;
    private boolean antiAlias = true;
    private List<GeoPosition> track;

    /**
     * @param track the track
     */
    public RoutePainter(List<GeoPosition> track)
    {
        // copy the list so that changes in the
        // original list do not have an effect here
        this.track = new ArrayList<>(track);
    }

```

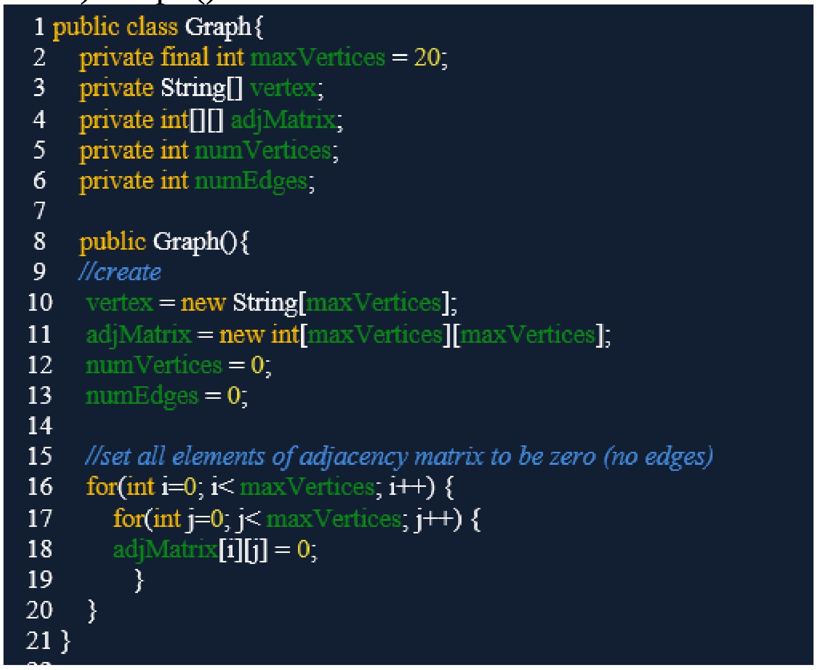


```
    @Override
    public void paint(Graphics2D g, JXMapViewer map, int w, int h)
    {
        g = (Graphics2D) g.create();
        // convert from viewport to world bitmap
        Rectangle rect = map.getViewportBounds();
        g.translate(-rect.x, -rect.y);
        if (antiAlias)
            g.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        // do the drawing
        g.setColor(Color.BLACK);
        g.setStroke(new BasicStroke(4));
        drawRoute(g, map);

        // do the drawing again
        g.setColor(color);
        g.setStroke(new BasicStroke(2));

        drawRoute(g, map);

        g.dispose();
    }

    /**
     * @param g the graphics object
     * @param map the map
     */
    private void drawRoute(Graphics2D g, JXMapViewer map)
    {
        int lastX = 0;
        int lastY = 0;
        boolean first = true;

        for (GeoPosition gp : track)
        {
            // convert geo-coordinate to world bitmap pixel
            Point2D pt = map.getTileFactory().geoToPixel(gp, map.getZoom());
            if (first)
            {
                first = false;
            }
            else
            {
                g.drawLine(lastX, lastY, (int) pt.getX(), (int) pt.getY());
            }

            lastX = (int) pt.getX();
            lastY = (int) pt.getY();
        }
    }
}
```

Graph

Graph class is the same class from the lecture.

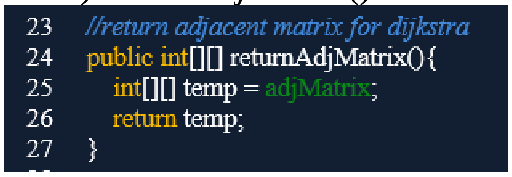
1.) Graph()



```
public class Graph{
    private final int maxVertices = 20;
    private String[] vertex;
    private int[][] adjMatrix;
    private int numVertices;
    private int numEdges;

    public Graph(){
    //create
     vertex = new String[maxVertices];
     adjMatrix = new int[maxVertices][maxVertices];
     numVertices = 0;
     numEdges = 0;

     //set all elements of adjacency matrix to be zero (no edges)
     for(int i=0; i< maxVertices; i++) {
        for(int j=0; j< maxVertices; j++) {
        adjMatrix[i][j] = 0;
          }
     }
}
```

At the top we have a constructor class and some variables.

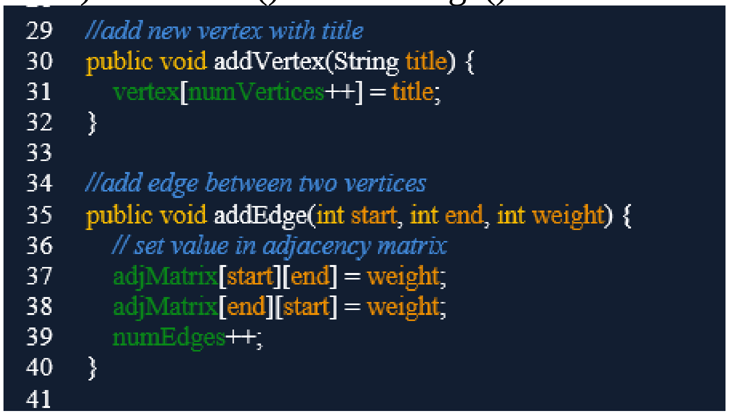
2.) returnAdjMatrix()



```
     //return adjacent matrix for dijkstra
     public int[][] returnAdjMatrix(){
        int[][] temp = adjMatrix;
        return temp;
     }
```

This function is very useful as it’s use to make a “Clone” adjacency matrix to be used in other functions. For example in adminJFrame it is use with dijkstra() to find the nearest node(store) to the user node.

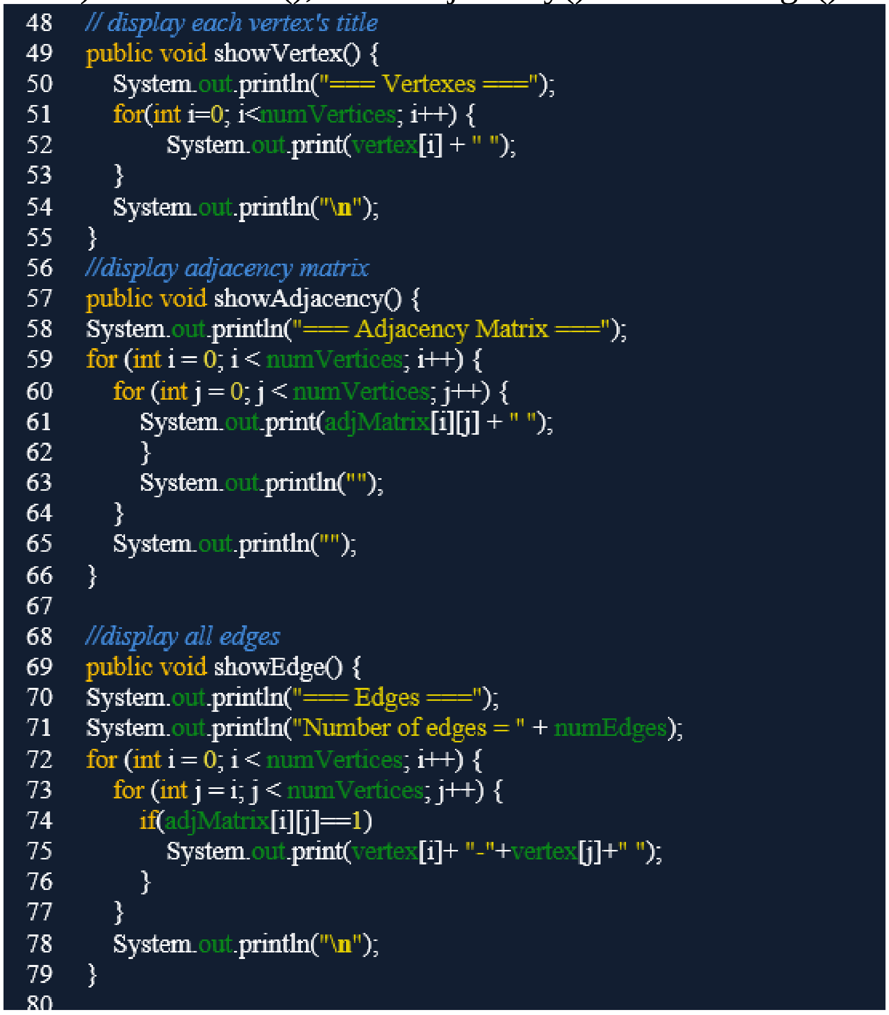
3.) addVertex() and addEdge()



```
//add new vertex with title
public void addVertex(String title) {
    vertex[numVertices++] = title;
}

//add edge between two vertices
public void addEdge(int start, int end, int weight) {
    // set value in adjacency matrix
    adjMatrix[start][end] = weight;
    adjMatrix[end][start] = weight;
    numEdges++;
}

```

Used for adding more vertex(node) and edge.

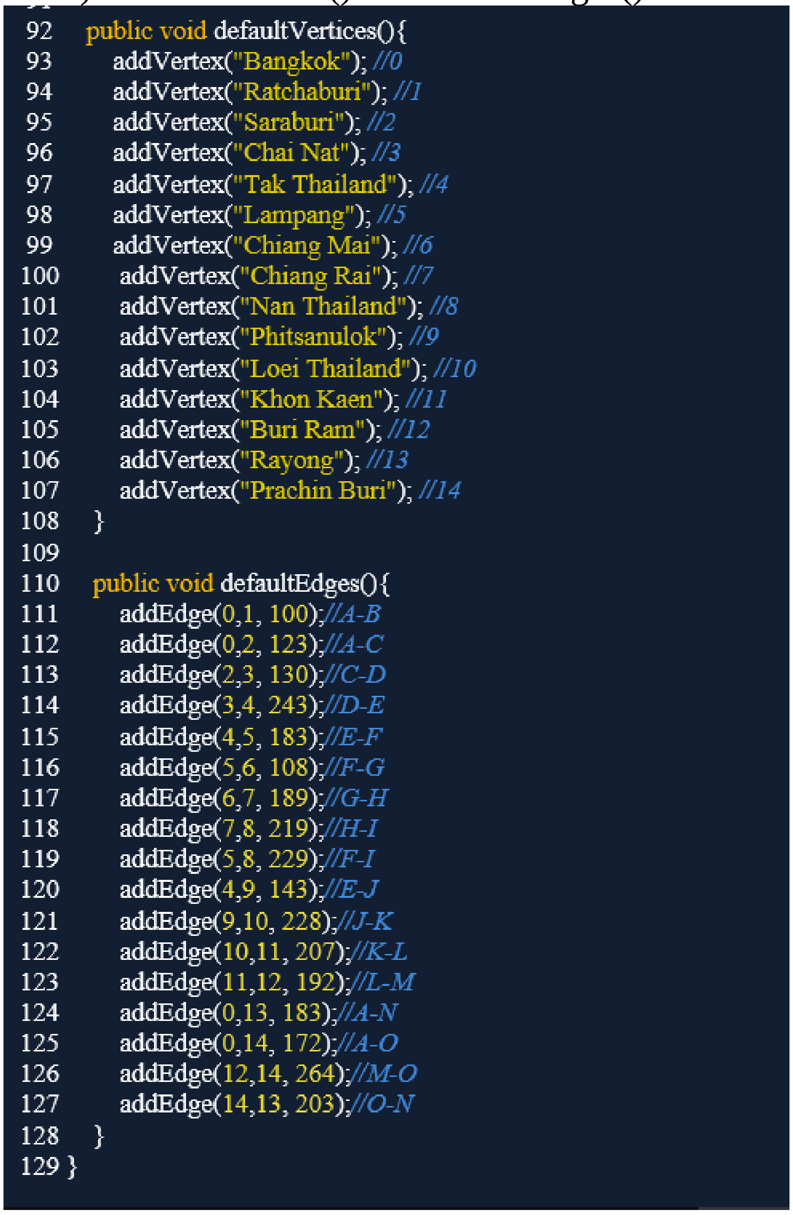
4.) showVertex(), showAdjacency() and showEdge()



```
// display each vertex's title
public void showVertex() {
    System.out.println("=== Vertexes ===");
    for(int i=0; i<numVertices; i++) {
            System.out.print(vertex[i] + " ");
    }
    System.out.println("\n");
}
//display adjacency matrix
public void showAdjacency() {
System.out.println("=== Adjacency Matrix ===");
for (int i = 0; i < numVertices; i++) {
    for (int j = 0; j < numVertices; j++) {
        System.out.print(adjMatrix[i][j] + " ");
        }
        System.out.println("");
    }
    System.out.println("");
}

//display all edges
public void showEdge() {
System.out.println("=== Edges ===");
System.out.println("Number of edges = " + numEdges);
for (int i = 0; i < numVertices; i++) {
    for (int j = i; j < numVertices; j++) {
        if(adjMatrix[i][j]==1)
            System.out.print(vertex[i]+ "-"+vertex[j]+" ");
        }
    }
    System.out.println("\n");
}

```

Their job is pretty self-explanatory.

5.) defaultVertices() and defaultEdges()

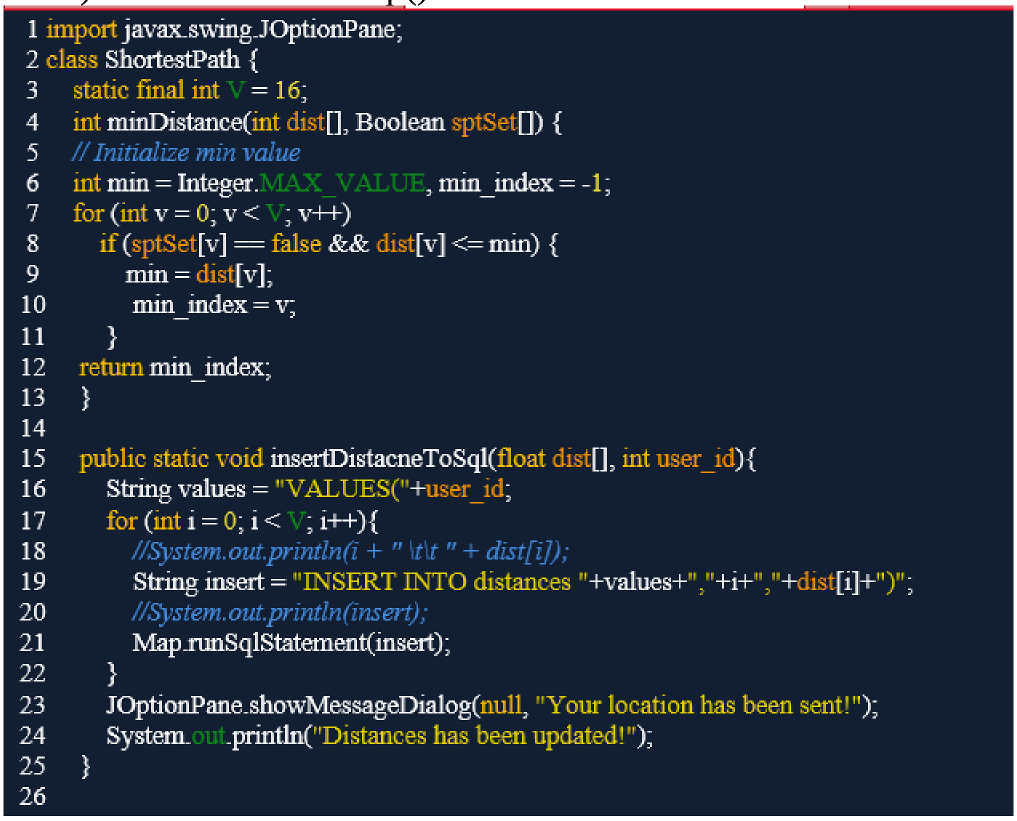


```
    public void defaultVertices(){
        addVertex("Bangkok"); //0
        addVertex("Ratchaburi"); //1
        addVertex("Saraburi"); //2
        addVertex("Chai Nat"); //3
        addVertex("Tak Thailand"); //4
        addVertex("Lampang"); //5
        addVertex("Chiang Mai"); //6
        addVertex("Chiang Rai"); //7
        addVertex("Nan Thailand"); //8
        addVertex("Phitsanulok"); //9
        addVertex("Loei Thailand"); //10
        addVertex("Khon Kaen"); //11
        addVertex("Buri Ram"); //12
        addVertex("Rayong"); //13
        addVertex("Prachin Buri"); //14
    }

    public void defaultEdges(){
        addEdge(0,1, 100);//A-B
        addEdge(0,2, 123);//A-C
        addEdge(2,3, 130);//C-D
        addEdge(3,4, 243);//D-E
        addEdge(4,5, 183);//E-F
        addEdge(5,6, 108);//F-G
        addEdge(6,7, 189);//G-H
        addEdge(7,8, 219);//H-I
        addEdge(5,8, 229);//F-I
        addEdge(4,9, 143);//E-J
        addEdge(9,10, 228);//J-K
        addEdge(10,11, 207);//K-L
        addEdge(11,12, 192);//L-M
        addEdge(0,13, 183);//A-N
        addEdge(0,14, 172);//A-O
        addEdge(12,14, 264);//M-O
        addEdge(14,13, 203);//O-N
    }
}
```

These two method will add all default vertices(all 15 stores) and all default edges(how those 15 stores are connected in delivery route).

ShortestPath

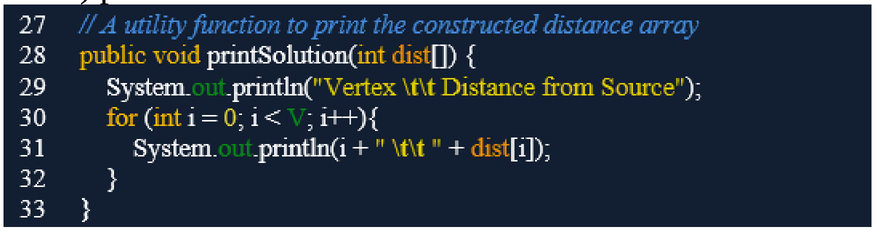
1.) insertDistanceToSql()



```java
import javax.swing.JOptionPane;
class ShortestPath {
    static final int V = 16;
    int minDistance(int dist[], Boolean sptSet[]) {
    // Initialize min value
    int min = Integer.MAX_VALUE, min_index = -1;
    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min) {
            min = dist[v];
             min_index = v;
        }
     return min_index;
     }

     public static void insertDistacneToSql(float dist[], int user_id){
         String values = "VALUES("+user_id;
         for (int i = 0; i < V; i++){
             //System.out.println(i + " \t\t " + dist[i]);
             String insert = "INSERT INTO distances "+values+","+i+","+dist[i]+")";
             //System.out.println(insert);
             Map.runSqlStatement(insert);
         }
         JOptionPane.showMessageDialog(null, "Your location has been sent!");
         System.out.println("Distances has been updated!");
     }

```

We declared some variables at the top. InsertDistanceToSql() will take distance From getDistanceMatrix() and put them in the database with user ID as its reference.

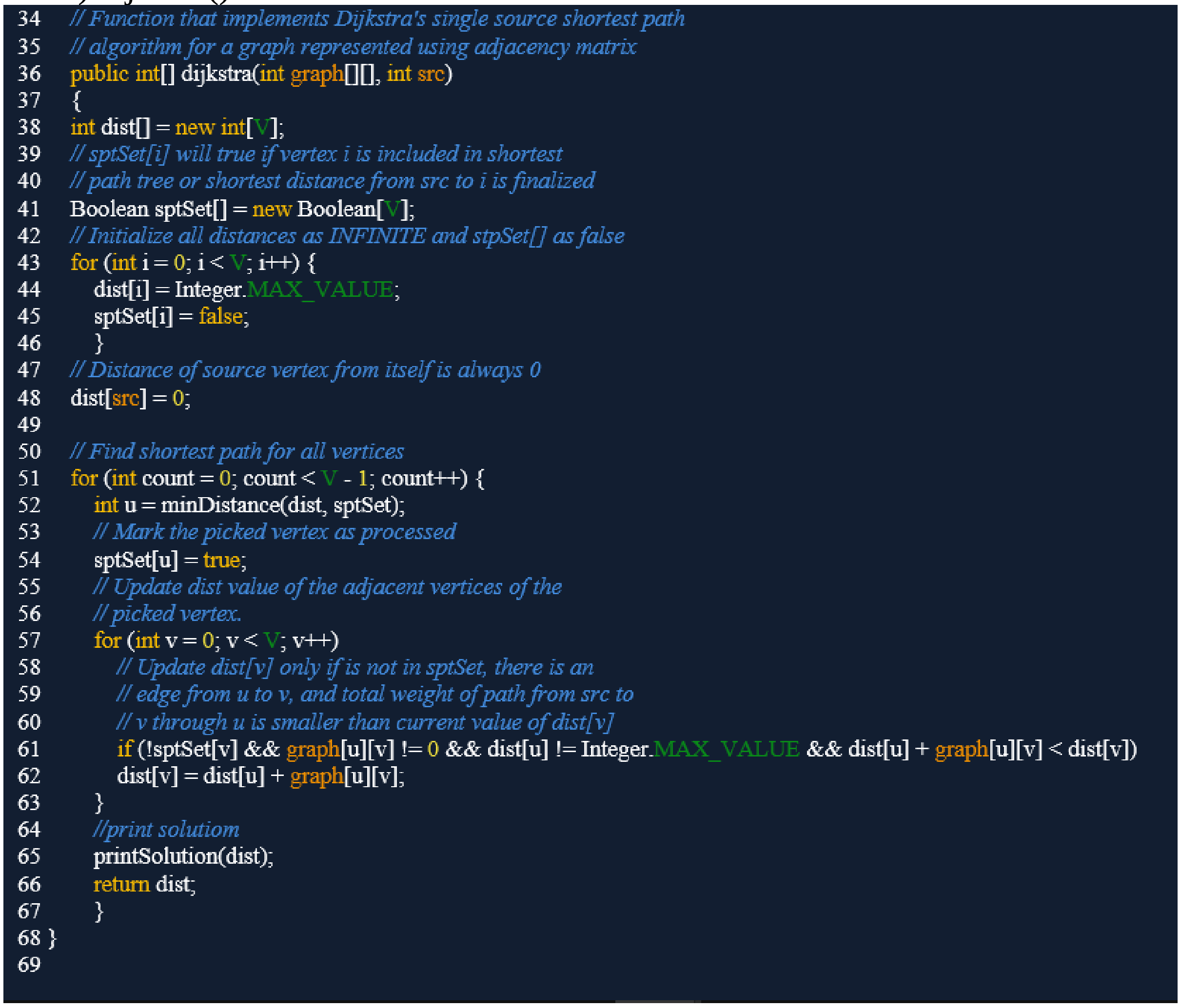
2.) printSolution



```java
     // A utility function to print the constructed distance array
     public void printSolution(int dist[]) {
         System.out.println("Vertex \t\t Distance from Source");
         for (int i = 0; i < V; i++){
             System.out.println(i + " \t\t " + dist[i]);
         }
     }
```

Just use to print the output.

3.) dijkstra()



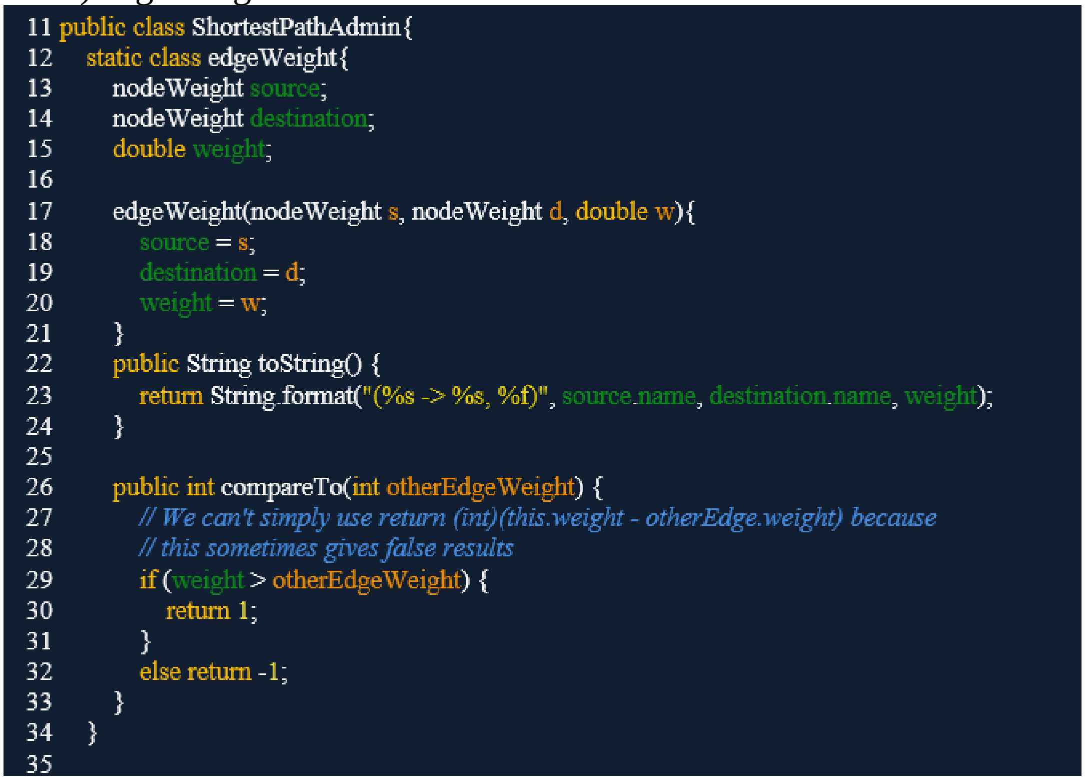
```
// Function that implements Dijkstra's single source shortest path
// algorithm for a graph represented using adjacency matrix
public int[] dijkstra(int graph[][], int src)
{
int dist[] = new int[V];
// sptSet[i] will true if vertex i is included in shortest
// path tree or shortest distance from src to i is finalized
Boolean sptSet[] = new Boolean[V];
// Initialize all distances as INFINITE and stpSet[] as false
for (int i = 0; i < V; i++) {
    dist[i] = Integer.MAX_VALUE;
    sptSet[i] = false;
    }
// Distance of source vertex from itself is always 0
dist[src] = 0;

// Find shortest path for all vertices
for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, sptSet);
    // Mark the picked vertex as processed
    sptSet[u] = true;
    // Update dist value of the adjacent vertices of the
    // picked vertex.
    for (int v = 0; v < V; v++)
        // Update dist[v] only if is not in sptSet, there is an
        // edge from u to v, and total weight of path from src to
        // v through u is smaller than current value of dist[v]
        if (!sptSet[v] && graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE && dist[u] + graph[u][v] < dist[v])
        dist[v] = dist[u] + graph[u][v];
    }
    //print solutiom
    printSolution(dist);
    return dist;
    }
}
```

The main function used to find distance from user to all 15 stores located in 15 provinces.

ShortestPathAdmin

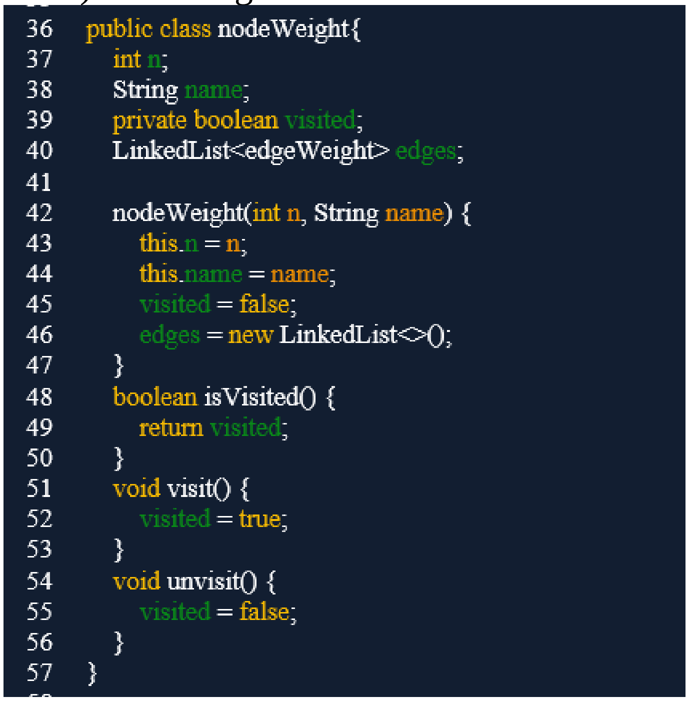
There are many sub-classes inside this class

1.) edgeWeight class



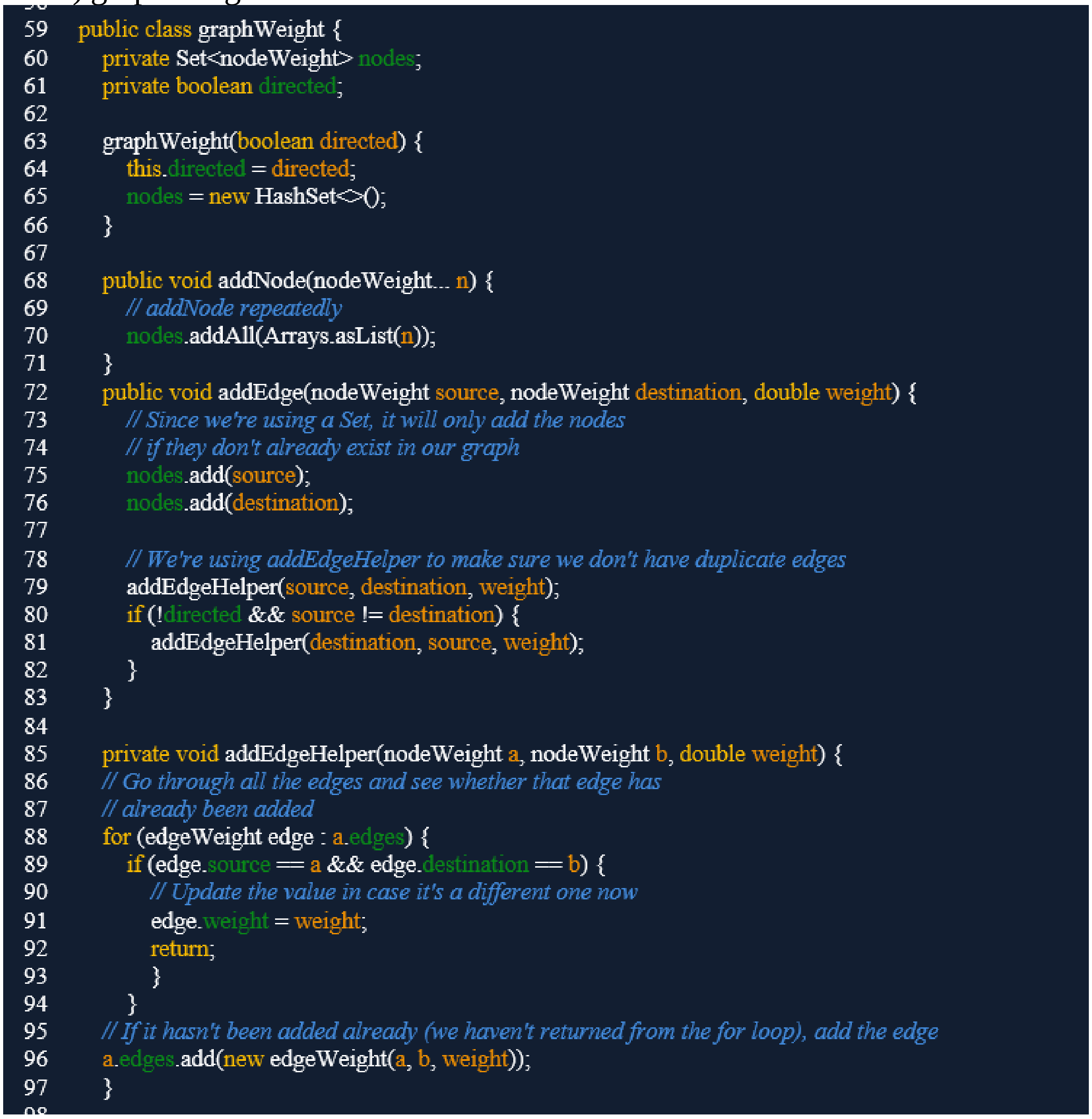
```
public class ShortestPathAdmin{
    static class edgeWeight{
        nodeWeight source;
        nodeWeight destination;
        double weight;

        edgeWeight(nodeWeight s, nodeWeight d, double w){
            source = s;
            destination = d;
            weight = w;
        }
        public String toString() {
            return String.format("(%s -> %s, %f)", source.name, destination.name, weight);
        }

        public int compareTo(int otherEdgeWeight) {
            // We can't simply use return (int)(this.weight - otherEdge.weight) because
            // this sometimes gives false results
            if (weight > otherEdgeWeight) {
                return 1;
            }
            else return -1;
        }
    }

```

2.) nodeWeight class

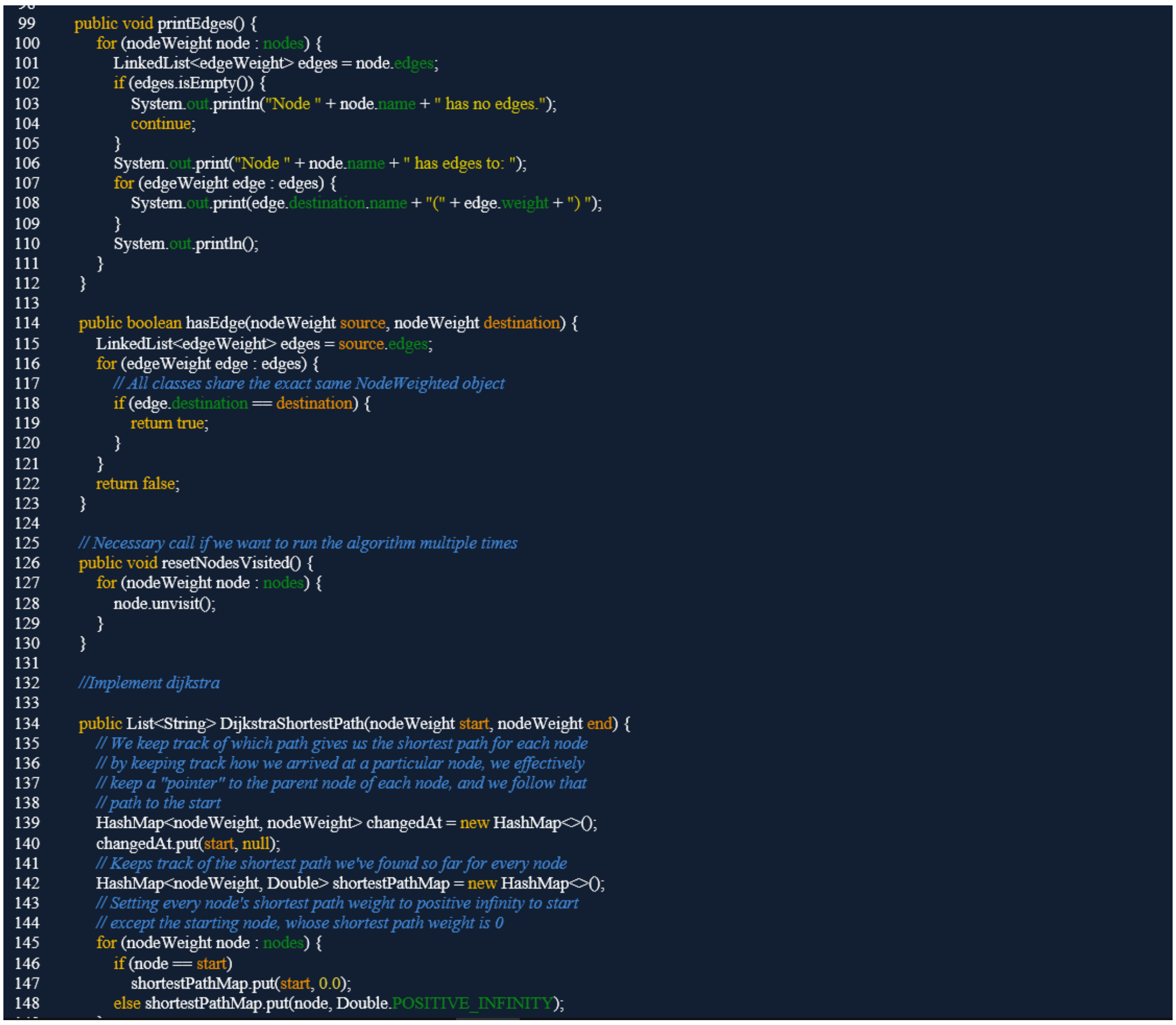


```
    public class nodeWeight{
        int n;
        String name;
        private boolean visited;
        LinkedList<edgeWeight> edges;

        nodeWeight(int n, String name) {
            this.n = n;
            this.name = name;
            visited = false;
            edges = new LinkedList<>();
        }
        boolean isVisited() {
            return visited;
        }
        void visit() {
            visited = true;
        }
        void unvisit() {
            visited = false;
        }
    }
```

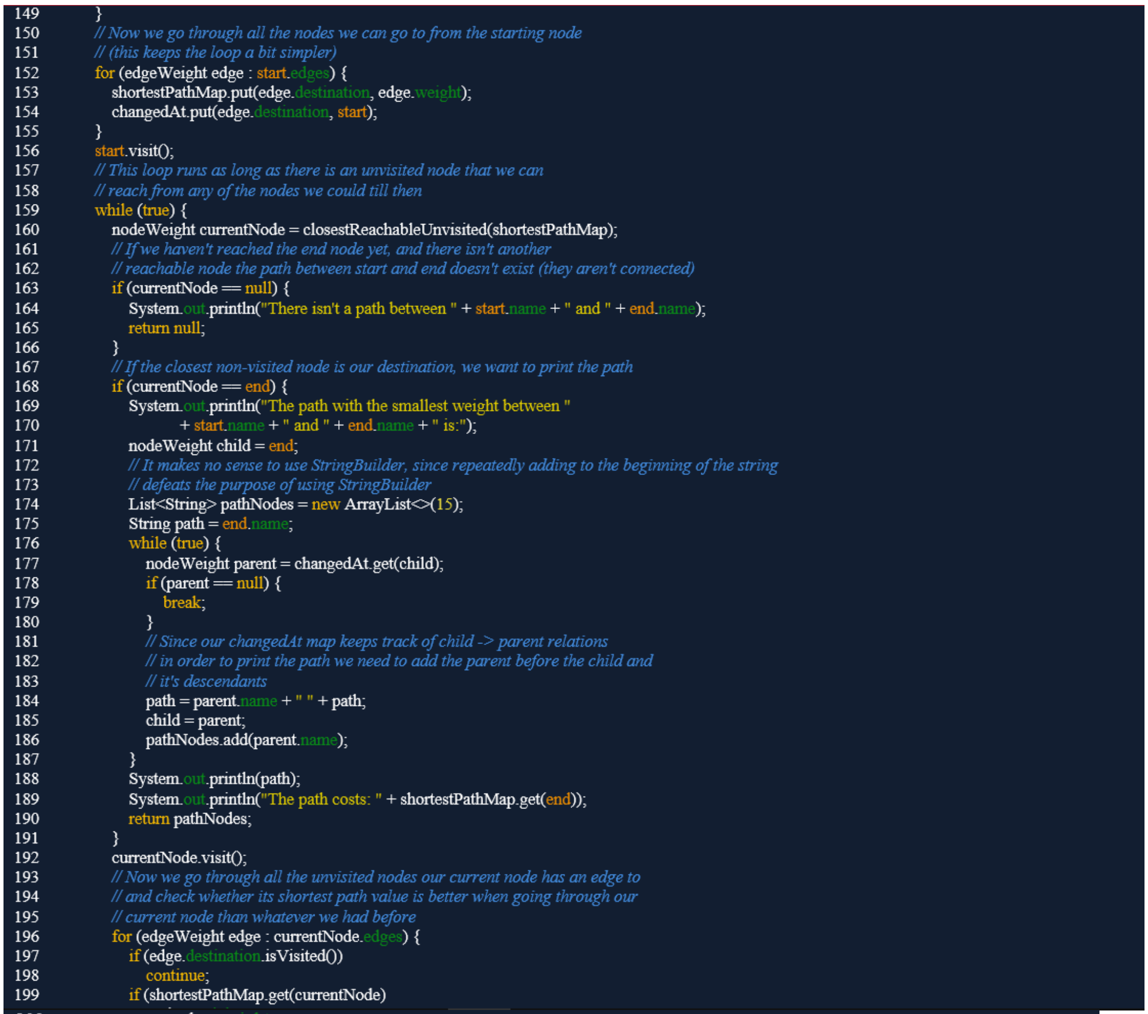
3.) graphWeight class



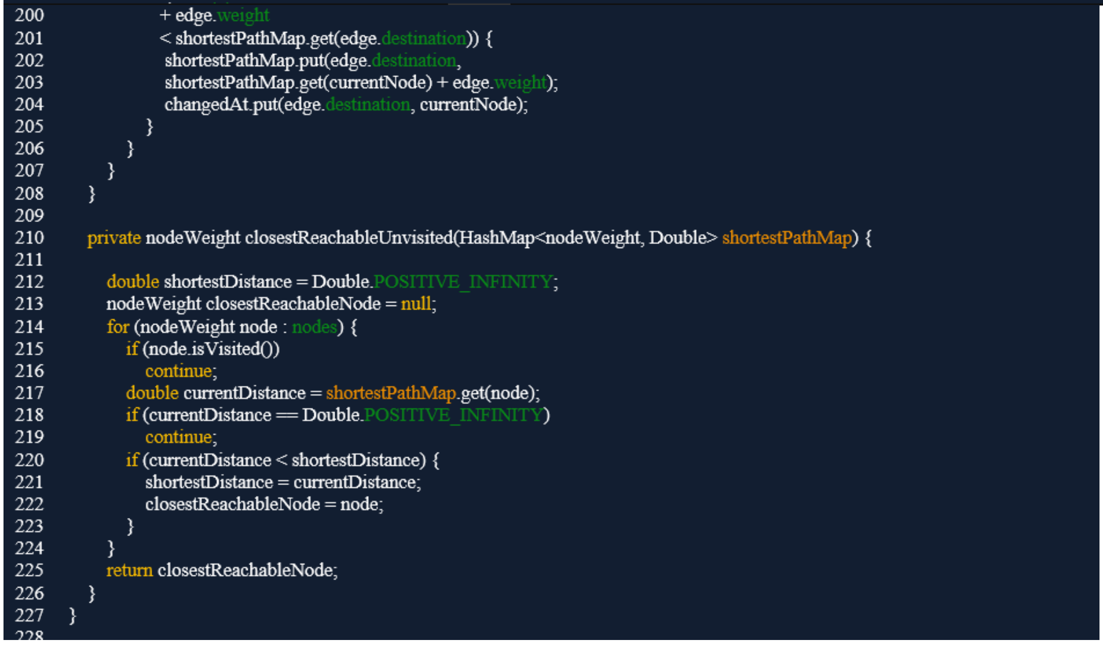
```java
public class graphWeight {
    private Set<nodeWeight> nodes;
    private boolean directed;

    graphWeight(boolean directed) {
        this.directed = directed;
        nodes = new HashSet<>();
    }

    public void addNode(nodeWeight... n) {
        // addNode repeatedly
        nodes.addAll(Arrays.asList(n));
    }
    public void addEdge(nodeWeight source, nodeWeight destination, double weight) {
        // Since we're using a Set, it will only add the nodes
        // if they don't already exist in our graph
        nodes.add(source);
        nodes.add(destination);

        // We're using addEdgeHelper to make sure we don't have duplicate edges
        addEdgeHelper(source, destination, weight);
        if (!directed && source != destination) {
            addEdgeHelper(destination, source, weight);
        }
    }

    private void addEdgeHelper(nodeWeight a, nodeWeight b, double weight) {
    // Go through all the edges and see whether that edge has
    // already been added
    for (edgeWeight edge : a.edges) {
        if (edge.source == a && edge.destination == b) {
            // Update the value in case it's a different one now
            edge.weight = weight;
            return;
            }
        }
    // If it hasn't been added already (we haven't returned from the for loop), add the edge
    a.edges.add(new edgeWeight(a, b, weight));
    }
```



```java
    public void printEdges() {
        for (nodeWeight node : nodes) {
            LinkedList<edgeWeight> edges = node.edges;
            if (edges.isEmpty()) {
                System.out.println("Node " + node.name + " has no edges.");
                continue;
            }
            System.out.print("Node " + node.name + " has edges to: ");
            for (edgeWeight edge : edges) {
                System.out.print(edge.destination.name + "(" + edge.weight + ") ");
            }
            System.out.println();
        }
    }

    public boolean hasEdge(nodeWeight source, nodeWeight destination) {
        LinkedList<edgeWeight> edges = source.edges;
        for (edgeWeight edge : edges) {
            // All classes share the exact same NodeWeighted object
            if (edge.destination == destination) {
                return true;
            }
        }
        return false;
    }

    // Necessary call if we want to run the algorithm multiple times
    public void resetNodesVisited() {
        for (nodeWeight node : nodes) {
            node.unvisit();
        }
    }

    //Implement dijkstra

    public List<String> DijkstraShortestPath(nodeWeight start, nodeWeight end) {
        // We keep track of which path gives us the shortest path for each node
        // by keeping track how we arrived at a particular node, we effectively
        // keep a "pointer" to the parent node of each node, and we follow that
        // path to the start
        HashMap<nodeWeight, nodeWeight> changedAt = new HashMap<>();
        changedAt.put(start, null);
        // Keeps track of the shortest path we've found so far for every node
        HashMap<nodeWeight, Double> shortestPathMap = new HashMap<>();
        // Setting every node's shortest path weight to positive infinity to start
        // except the starting node, whose shortest path weight is 0
        for (nodeWeight node : nodes) {
            if (node == start)
                shortestPathMap.put(start, 0.0);
            else shortestPathMap.put(node, Double.POSITIVE_INFINITY);
```

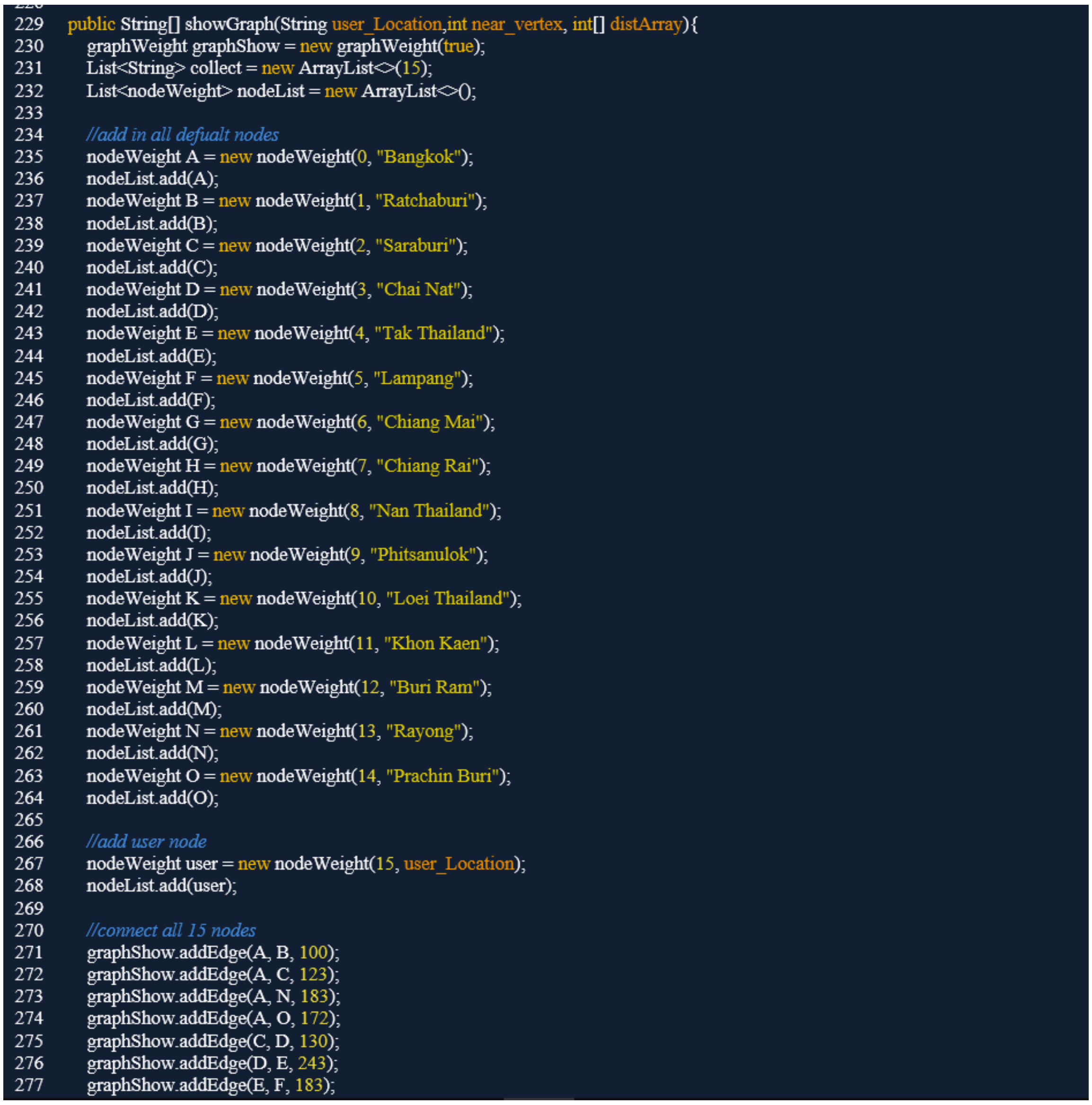


```
        }
        // Now we go through all the nodes we can go to from the starting node
        // (this keeps the loop a bit simpler)
        for (edgeWeight edge : start.edges) {
            shortestPathMap.put(edge.destination, edge.weight);
            changedAt.put(edge.destination, start);
        }
        start.visit();
        // This loop runs as long as there is an unvisited node that we can
        // reach from any of the nodes we could till then
        while (true) {
            nodeWeight currentNode = closestReachableUnvisited(shortestPathMap);
            // If we haven't reached the end node yet, and there isn't another
            // reachable node the path between start and end doesn't exist (they aren't connected)
            if (currentNode == null) {
                System.out.println("There isn't a path between " + start.name + " and " + end.name);
                return null;
            }
            // If the closest non-visited node is our destination, we want to print the path
            if (currentNode == end) {
                System.out.println("The path with the smallest weight between "
                            + start.name + " and " + end.name + " is:");
                nodeWeight child = end;
                // It makes no sense to use StringBuilder, since repeatedly adding to the beginning of the string
                // defeats the purpose of using StringBuilder
                List<String> pathNodes = new ArrayList<>(15);
                String path = end.name;
                while (true) {
                    nodeWeight parent = changedAt.get(child);
                    if (parent == null) {
                        break;
                    }
                    // Since our changedAt map keeps track of child -> parent relations
                    // in order to print the path we need to add the parent before the child and
                    // it's descendants
                    path = parent.name + " " + path;
                    child = parent;
                    pathNodes.add(parent.name);
                }
                System.out.println(path);
                System.out.println("The path costs: " + shortestPathMap.get(end));
                return pathNodes;
            }
            currentNode.visit();
            // Now we go through all the unvisited nodes our current node has an edge to
            // and check whether its shortest path value is better when going through our
            // current node than whatever we had before
            for (edgeWeight edge : currentNode.edges) {
                if (edge.destination.isVisited())
                    continue;
                if (shortestPathMap.get(currentNode)
```

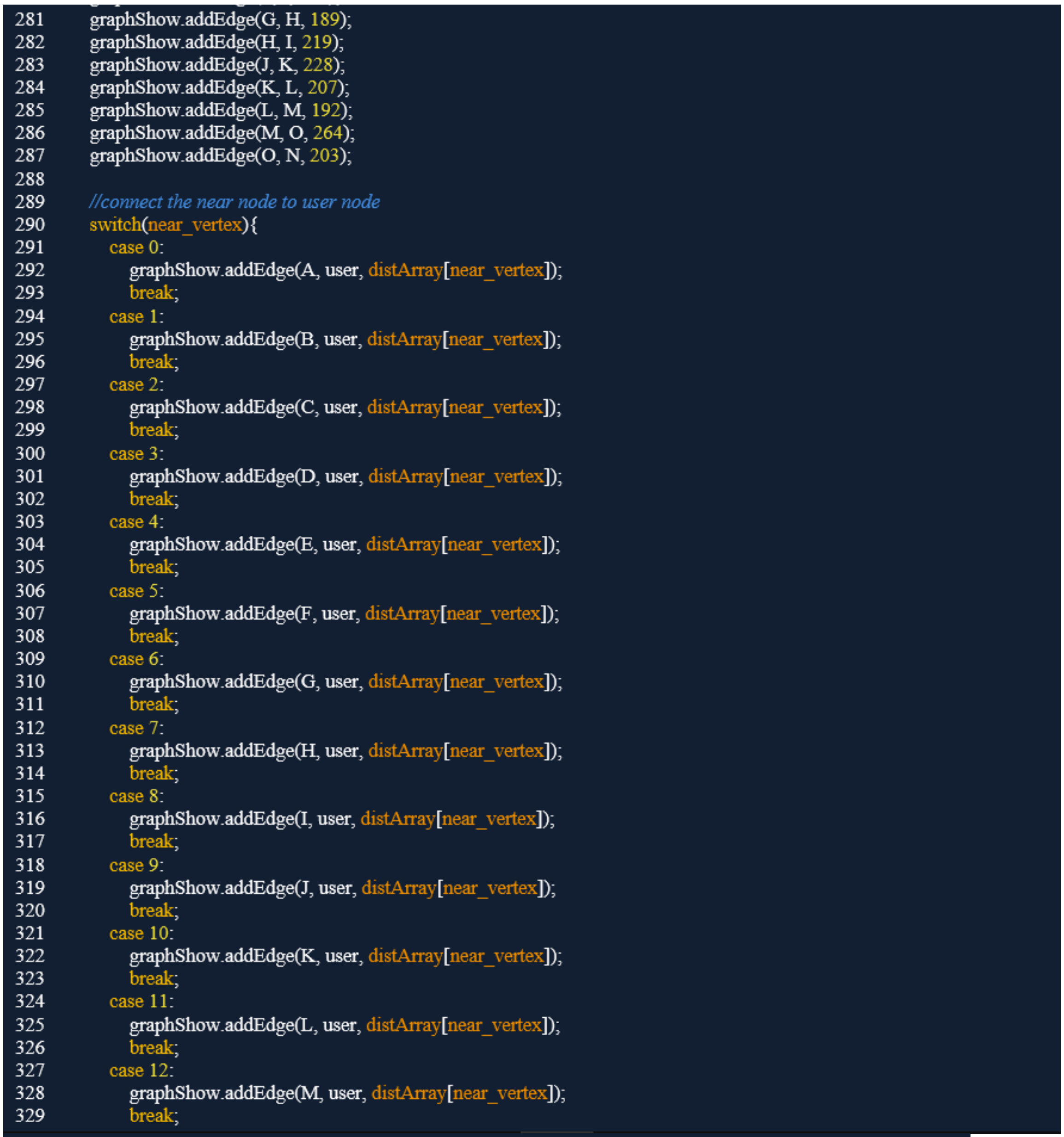


```
                        + edge.weight
                        < shortestPathMap.get(edge.destination)) {
                    shortestPathMap.put(edge.destination,
                    shortestPathMap.get(currentNode) + edge.weight);
                    changedAt.put(edge.destination, currentNode);
                }
            }
        }
    }

    private nodeWeight closestReachableUnvisited(HashMap<nodeWeight, Double> shortestPathMap) {

        double shortestDistance = Double.POSITIVE_INFINITY;
        nodeWeight closestReachableNode = null;
        for (nodeWeight node : nodes) {
            if (node.isVisited())
                continue;
            double currentDistance = shortestPathMap.get(node);
            if (currentDistance == Double.POSITIVE_INFINITY)
                continue;
            if (currentDistance < shortestDistance) {
                shortestDistance = currentDistance;
                closestReachableNode = node;
            }
        }
        return closestReachableNode;
    }
}
```

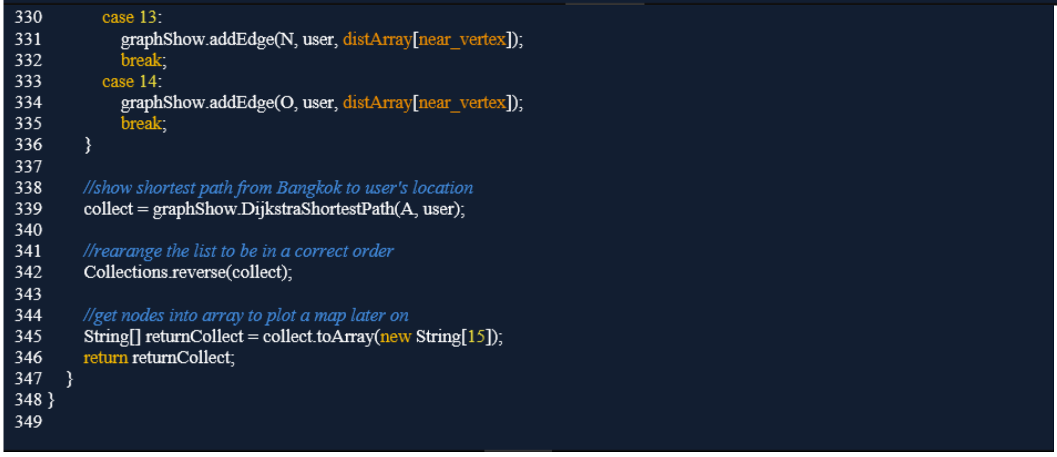
showGraph()



```
public String[] showGraph(String user_Location,int near_vertex, int[] distArray){
    graphWeight graphShow = new graphWeight(true);
    List<String> collect = new ArrayList<>(15);
    List<nodeWeight> nodeList = new ArrayList<>();

    //add in all defualt nodes
    nodeWeight A = new nodeWeight(0, "Bangkok");
    nodeList.add(A);
    nodeWeight B = new nodeWeight(1, "Ratchaburi");
    nodeList.add(B);
    nodeWeight C = new nodeWeight(2, "Saraburi");
    nodeList.add(C);
    nodeWeight D = new nodeWeight(3, "Chai Nat");
    nodeList.add(D);
    nodeWeight E = new nodeWeight(4, "Tak Thailand");
    nodeList.add(E);
    nodeWeight F = new nodeWeight(5, "Lampang");
    nodeList.add(F);
    nodeWeight G = new nodeWeight(6, "Chiang Mai");
    nodeList.add(G);
    nodeWeight H = new nodeWeight(7, "Chiang Rai");
    nodeList.add(H);
    nodeWeight I = new nodeWeight(8, "Nan Thailand");
    nodeList.add(I);
    nodeWeight J = new nodeWeight(9, "Phitsanulok");
    nodeList.add(J);
    nodeWeight K = new nodeWeight(10, "Loei Thailand");
    nodeList.add(K);
    nodeWeight L = new nodeWeight(11, "Khon Kaen");
    nodeList.add(L);
    nodeWeight M = new nodeWeight(12, "Buri Ram");
    nodeList.add(M);
    nodeWeight N = new nodeWeight(13, "Rayong");
    nodeList.add(N);
    nodeWeight O = new nodeWeight(14, "Prachin Buri");
    nodeList.add(O);

    //add user node
    nodeWeight user = new nodeWeight(15, user_Location);
    nodeList.add(user);

    //connect all 15 nodes
    graphShow.addEdge(A, B, 100);
    graphShow.addEdge(A, C, 123);
    graphShow.addEdge(A, N, 183);
    graphShow.addEdge(A, O, 172);
    graphShow.addEdge(C, D, 130);
    graphShow.addEdge(D, E, 243);
    graphShow.addEdge(E, F, 183);
```



```
        graphShow.addEdge(G, H, 189);
        graphShow.addEdge(H, I, 219);
        graphShow.addEdge(J, K, 228);
        graphShow.addEdge(K, L, 207);
        graphShow.addEdge(L, M, 192);
        graphShow.addEdge(M, O, 264);
        graphShow.addEdge(O, N, 203);

        //connect the near node to user node
        switch(near_vertex){
            case 0:
                graphShow.addEdge(A, user, distArray[near_vertex]);
                break;
            case 1:
                graphShow.addEdge(B, user, distArray[near_vertex]);
                break;
            case 2:
                graphShow.addEdge(C, user, distArray[near_vertex]);
                break;
            case 3:
                graphShow.addEdge(D, user, distArray[near_vertex]);
                break;
            case 4:
                graphShow.addEdge(E, user, distArray[near_vertex]);
                break;
            case 5:
                graphShow.addEdge(F, user, distArray[near_vertex]);
                break;
            case 6:
                graphShow.addEdge(G, user, distArray[near_vertex]);
                break;
            case 7:
                graphShow.addEdge(H, user, distArray[near_vertex]);
                break;
            case 8:
                graphShow.addEdge(I, user, distArray[near_vertex]);
                break;
            case 9:
                graphShow.addEdge(J, user, distArray[near_vertex]);
                break;
            case 10:
                graphShow.addEdge(K, user, distArray[near_vertex]);
                break;
            case 11:
                graphShow.addEdge(L, user, distArray[near_vertex]);
                break;
            case 12:
                graphShow.addEdge(M, user, distArray[near_vertex]);
                break;
```



```
            case 13:
                graphShow.addEdge(N, user, distArray[near_vertex]);
                break;
            case 14:
                graphShow.addEdge(O, user, distArray[near_vertex]);
                break;
        }

        //show shortest path from Bangkok to user's location
        collect = graphShow.DijkstraShortestPath(A, user);

        //rearange the list to be in a correct order
        Collections.reverse(collect);

        //get nodes into array to plot a map later on
        String[] returnCollect = collect.toArray(new String[15]);
        return returnCollect;
    }
}
```

This method will set default nodesw, edges, show path from source(Bangkok) to the user’s location