```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package seafarm;

import com.teamdev.jxmaps.GeocoderCallback;
import com.teamdev.jxmaps.GeocoderRequest;
import com.teamdev.jxmaps.GeocoderResult;
import com.teamdev.jxmaps.GeocoderStatus;
import com.teamdev.jxmaps.InfoWindow;
import com.teamdev.jxmaps.LatLng;
import com.teamdev.jxmaps.Map;
import com.teamdev.jxmaps.Icon;
import com.teamdev.jxmaps.MapMouseEvent;
import com.teamdev.jxmaps.MapReadyHandler;
import com.teamdev.jxmaps.MapStatus;
import com.teamdev.jxmaps.MapViewOptions;
import com.teamdev.jxmaps.Marker;
import com.teamdev.jxmaps.Size;
import com.teamdev.jxmaps.PlaceSearchRequest;
import com.teamdev.jxmaps.MouseEvent;
import com.teamdev.jxmaps.Point;
import com.teamdev.jxmaps.Polyline;
import com.teamdev.jxmaps.PolylineOptions;
import com.teamdev.jxmaps.javafx.MapView;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

/**
 *
 * @author Platform
 */
public class maps {
    public MapView mapView;
    public Map map;
    public Marker platformMarker;
    public List<Marker> listMarker;
    public int lngCnt = 0;
    public int latCnt = 0;
```

```
    public double lat = 62.471948;
    public double lng = 6.235196;
    double nextPos1 = 0.0;
    private Icon platformIcon;
    public maps() {

        listMarker = new LinkedList<>();

        MapViewOptions options = new MapViewOptions();
        options.importPlaces();
        options.setApiKey("AIzaSyB_sLP3KV14XnGFYL5gAFAC6ErzJ-FrO48");
        mapView = new MapView(options);

        mapView.setOnMapReadyHandler(new MapReadyHandler() {
            @Override
            public void onMapReady(MapStatus status) {
                if (status == MapStatus.MAP_STATUS_OK) {
                    map = mapView.getMap();
                    map.setZoom(15.0);
                    GeocoderRequest request = new GeocoderRequest();
                    request.setAddress("Nørvevika, NO");

                    mapView.getServices().getGeocoder().geocode(request,
new GeocoderCallback(map) {
                        @Override
                        public void onComplete(GeocoderResult[] result,
GeocoderStatus status) {
                            if (status == GeocoderStatus.OK) {
                                //map.setCenter(result[0].getGeometry().get
Location());
                                map.setCenter(new LatLng(lat, lng));
                                platformMarker = new Marker(map);
                                  //Creating Icon for the platform
                                platformIcon = new Icon();
                                platformIcon.loadFromStream((getClass().get
ResourceAsStream("PlatformImg.png")),"png");

                                Size size=new Size(200,200);
                                platformIcon.setScaledSize(size);
                                platformIcon.setAnchor(new Point(100,100));



                                platformMarker.setIcon(platformIcon);
```

```
                                        //marker.setPosition(result[0].getG
eometry().getLocation());
                                platformMarker.setPosition(new LatLng(lat,
lng));

                                listMarker.add(platformMarker);


                                //

                    map.addEventListener("click", new MapMouseEvent() {
                      @Override
                      public void onEvent(MouseEvent mouseEvent) {
                          // Closing initially created info window

                          // Creating a new marker
                          final Marker marker2 = new Marker(map);
                          // Move marker to the position where user
clicked
                          marker2.setPosition(mouseEvent.latLng());
                          listMarker.add(marker2);

                          System.out.println("Adding position: " +
marker2.getPosition().toString());

                          // Adding event listener that intercepts
clicking on marker
                          marker2.addEventListener("click", new
MapMouseEvent() {
                              @Override
                              public void onEvent(MouseEvent mouseEvent)
{
                                  // Removing marker from the map
                                  marker2.remove();


                                  System.out.println("Removing position:
" + marker2.getPosition().toString());
                                  listMarker.remove(marker2);
                                  latCnt--;
                                  lngCnt--;
                              }
                          });
```

```
                        }
                    });

                            }
                        }
                    });
                }
            }
        });
    }

    public MapView getMapView() {
        return mapView;
    }
    public void printList(){
        Iterator<Marker> iterator = listMarker.iterator();
        while(iterator.hasNext()){
            System.out.println("Positions in list at closing: " +
iterator.next().getPosition().toString());
        }
    }

    public double getNextPosLat(){
        System.out.println("get next Pos lat called MAPSclass ");
        double nextPos = 0.00;
        if(listMarker.iterator().hasNext()){
          if(latCnt == listMarker.size()){
              latCnt = listMarker.size() - 1;
             // System.out.println(latCnt);
              nextPos = listMarker.get(latCnt).getPosition().getLat();
          }
          else if(latCnt  < listMarker.size()){
              nextPos = listMarker.get(latCnt).getPosition().getLat();
              latCnt++;
              //System.out.println(latCnt);
          }
        }
        return nextPos;
    }
    public double getNextPosLng(){
        double nextPos = 0.00;
        if(listMarker.iterator().hasNext()){
          if(lngCnt == listMarker.size()){
              lngCnt = listMarker.size() - 1;
```

```
            // System.out.println(lngCnt);
             nextPos = listMarker.get(lngCnt).getPosition().getLng();
          }
          else if(lngCnt  < listMarker.size()){
              nextPos = listMarker.get(lngCnt).getPosition().getLng();
              lngCnt++;
             // System.out.println(lngCnt);
          }
        }
        return nextPos;
    }

    public double getNext(){

        for(int i = 0; i < listMarker.size();i++){
            if(nextPos1 == 0.0){
                nextPos1 = listMarker.get(0).getPosition().getLat();
                //System.out.println("If1");
                break;

            }
            if((listMarker.get(i).getPosition().getLat() == nextPos1) && i
+ 1 != listMarker.size()){
                nextPos1 = listMarker.get(i+1).getPosition().getLat();
                //System.out.println("If2");
                break;
            }
            else if(i + 1 == listMarker.size()){
                System.out.println("else");
                break;
            }
        }

        return nextPos1;
    }

    public void setLat(float latitude){
        lat = (float) Double.parseDouble(Float.toString(latitude));


    }
    public void setLng(float longitude){
        lng = (float) Double.parseDouble(Float.toString(longitude));
    }
```

```
    public void setRovPosition(){
      //  platformMarker.setCenter(new LatLng(lat, lng));









        //marker.setPosition(result[0].getGeometry().getLocation());

        platformMarker.setPosition(new LatLng(lat, lng));
        //listMarker.add(marker);
    }



}
```