**package** com.example.navya.googlemapapi;  
  
**import** android.os.AsyncTask;  
  
**import** com.google.android.gms.maps.model.LatLng;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.io.UnsupportedEncodingException;  
**import** java.net.MalformedURLException;  
**import** java.net.URL;  
**import** java.net.URLEncoder;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by Navya on 9/20/17.  
 \*/***public class** DirectionFinder {  
 **private static final** String ***DIRECTION\_URL\_API*** = **"https://maps.googleapis.com/maps/api/directions/json?"**;  
 **private static final** String ***GOOGLE\_API\_KEY*** = **"AIzaSyDnwLF2-WfK8cVZt9OoDYJ9Y8kspXhEHfI"**;  
 **private** DirectionFinderListener **listener**;  
 **private** String **origin**;  
 **private** String destination;  
  
 **public** DirectionFinder(DirectionFinderListener listener, String origin, String destination) {  
 **this**.**listener** = listener;  
 **this**.**origin** = origin;  
 **this**.destination = destination;  
 }  
  
 **public void** execute() **throws** UnsupportedEncodingException {  
 **listener**.onDirectionFinderStart();  
 **new** DownloadRawData().execute(createUrl());  
 }  
  
 **private** String createUrl() **throws** UnsupportedEncodingException {  
 String urlOrigin = URLEncoder.*encode*(**origin**, **"utf-8"**);  
 String urlDestination = URLEncoder.*encode*(destination, **"utf-8"**);  
  
 **return *DIRECTION\_URL\_API*** + **"origin="** + urlOrigin + **"&destination="** + urlDestination + **"&key="** + ***GOOGLE\_API\_KEY***;  
 }  
  
 **private class** DownloadRawData **extends** AsyncTask<String, Void, String> {  
  
 @Override  
 **protected** String doInBackground(String... params) {  
 String link = params[0];  
 **try** {  
 URL url = **new** URL(link);  
 InputStream is = url.openConnection().getInputStream();  
 StringBuffer buffer = **new** StringBuffer();  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));  
  
 String line;  
 **while** ((line = reader.readLine()) != **null**) {  
 buffer.append(line + **"\n"**);  
 }  
  
 **return** buffer.toString();  
  
 } **catch** (MalformedURLException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String res) {  
 **try** {  
 parseJSon(res);  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 **private void** parseJSon(String data) **throws** JSONException {  
 **if** (data == **null**)  
 **return**;  
  
 List<Route> routes = **new** ArrayList<Route>();  
 JSONObject jsonData = **new** JSONObject(data);  
 JSONArray jsonRoutes = jsonData.getJSONArray(**"routes"**);  
 **for** (**int** i = 0; i < jsonRoutes.length(); i++) {  
 JSONObject jsonRoute = jsonRoutes.getJSONObject(i);  
 Route route = **new** Route();  
  
 JSONObject overview\_polylineJson = jsonRoute.getJSONObject(**"overview\_polyline"**);  
 JSONArray jsonLegs = jsonRoute.getJSONArray(**"legs"**);  
 JSONObject jsonLeg = jsonLegs.getJSONObject(0);  
 JSONObject jsonDistance = jsonLeg.getJSONObject(**"distance"**);  
 JSONObject jsonDuration = jsonLeg.getJSONObject(**"duration"**);  
 JSONObject jsonEndLocation = jsonLeg.getJSONObject(**"end\_location"**);  
 JSONObject jsonStartLocation = jsonLeg.getJSONObject(**"start\_location"**);  
  
 route.**distance** = **new** Distance(jsonDistance.getString(**"text"**), jsonDistance.getInt(**"value"**));  
 route.**duration** = **new** Duration(jsonDuration.getString(**"text"**), jsonDuration.getInt(**"value"**));  
 route.**endAddress** = jsonLeg.getString(**"end\_address"**);  
 route.**startAddress** = jsonLeg.getString(**"start\_address"**);  
 route.**startLocation** = **new** LatLng(jsonStartLocation.getDouble(**"lat"**), jsonStartLocation.getDouble(**"lng"**));  
 route.**endLocation** = **new** LatLng(jsonEndLocation.getDouble(**"lat"**), jsonEndLocation.getDouble(**"lng"**));  
 route.**points** = decodePolyLine(overview\_polylineJson.getString(**"points"**));  
  
 routes.add(route);  
 }  
  
 **listener**.onDirectionFinderSuccess(routes);  
 }  
  
 **private** List<LatLng> decodePolyLine(**final** String poly) {  
 **int** len = poly.length();  
 **int** index = 0;  
 List<LatLng> decoded = **new** ArrayList<LatLng>();  
 **int** lat = 0;  
 **int** lng = 0;  
  
 **while** (index < len) {  
 **int** b;  
 **int** shift = 0;  
 **int** result = 0;  
 **do** {  
 b = poly.charAt(index++) - 63;  
 result |= (b & 0x1f) << shift;  
 shift += 5;  
 } **while** (b >= 0x20);  
 **int** dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
 lat += dlat;  
  
 shift = 0;  
 result = 0;  
 **do** {  
 b = poly.charAt(index++) - 63;  
 result |= (b & 0x1f) << shift;  
 shift += 5;  
 } **while** (b >= 0x20);  
 **int** dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
 lng += dlng;  
  
 decoded.add(**new** LatLng(  
 lat / 100000d, lng / 100000d  
 ));  
 }  
  
 **return** decoded;  
 }  
}