package com.example.mobiletermproject;  
  
import android.Manifest;  
import android.annotation.TargetApi;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Bitmap;  
import android.graphics.drawable.BitmapDrawable;  
import android.location.Address;  
import android.location.Geocoder;  
import android.location.Location;  
import android.location.LocationManager;  
import android.net.Uri;  
import android.os.Build;  
import android.os.StrictMode;  
import android.provider.Settings;  
import android.support.annotation.NonNull;  
import android.support.v4.app.ActivityCompat;  
import android.support.v4.content.ContextCompat;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.WindowManager;  
import android.widget.Toast;  
  
import com.google.android.gms.common.ConnectionResult;  
import com.google.android.gms.common.api.GoogleApiClient;  
import com.google.android.gms.location.LocationListener;  
import com.google.android.gms.location.LocationRequest;  
import com.google.android.gms.location.LocationServices;  
import com.google.android.gms.maps.CameraUpdate;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.MapFragment;  
import com.google.android.gms.maps.MapsInitializer;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.model.BitmapDescriptorFactory;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.Marker;  
import com.google.android.gms.maps.model.MarkerOptions;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Locale;  
  
  
public class MainActivity extends AppCompatActivity  
 implements OnMapReadyCallback,  
 GoogleApiClient.ConnectionCallbacks,  
 GoogleApiClient.OnConnectionFailedListener,  
 LocationListener {  
  
  
 private GoogleApiClient mGoogleApiClient = null;  
 private GoogleMap mGoogleMap = null;  
 private GoogleMap mMap = null;  
 private Marker currentMarker = null;  
  
 private static final String *TAG* = "googlemap\_example";  
 private static final int *GPS\_ENABLE\_REQUEST\_CODE* = 2001;  
 private static final int *PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION* = 2002;  
 private static final int *UPDATE\_INTERVAL\_MS* = 1000; // 1초  
 private static final int *FASTEST\_UPDATE\_INTERVAL\_MS* = 500; // 0.5초  
  
 private AppCompatActivity mActivity;  
 boolean askPermissionOnceAgain = false;  
 boolean mRequestingLocationUpdates = false;  
 Location mCurrentLocatiion;  
 boolean mMoveMapByUser = true;  
 boolean mMoveMapByAPI = true;  
 LatLng currentPosition;  
  
 LocationRequest locationRequest = new LocationRequest()  
 .setPriority(LocationRequest.*PRIORITY\_HIGH\_ACCURACY*)  
 .setInterval(*UPDATE\_INTERVAL\_MS*)  
 .setFastestInterval(*FASTEST\_UPDATE\_INTERVAL\_MS*);  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*,  
 WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);  
 setContentView(R.layout.*activity\_main*);  
  
  
 Log.*d*(*TAG*, "onCreate");  
 mActivity = this;  
  
  
 mGoogleApiClient = new GoogleApiClient.Builder(this)  
 .addConnectionCallbacks(this)  
 .addOnConnectionFailedListener(this)  
 .addApi(LocationServices.*API*)  
 .build();  
  
  
 MapFragment mapFragment = (MapFragment) getFragmentManager()  
 .findFragmentById(R.id.map);  
 mapFragment.getMapAsync(this);  
 StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
  
 }  
  
  
 @Override  
 public void onResume() {  
  
 super.onResume();  
  
 if (mGoogleApiClient.isConnected()) {  
  
 Log.*d*(*TAG*, "onResume : call startLocationUpdates");  
 if (!mRequestingLocationUpdates) startLocationUpdates();  
 }  
  
  
 //앱 정보에서 퍼미션을 허가했는지를 다시 검사해봐야 한다.  
 if (askPermissionOnceAgain) {  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 askPermissionOnceAgain = false;  
  
 checkPermissions();  
 }  
 }  
 }  
  
  
 private void startLocationUpdates() {  
  
 if (!checkLocationServicesStatus()) {  
  
 Log.*d*(*TAG*, "startLocationUpdates : call showDialogForLocationServiceSetting");  
 showDialogForLocationServiceSetting();  
 }else {  
  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_COARSE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*) {  
  
 Log.*d*(*TAG*, "startLocationUpdates : 퍼미션 안가지고 있음");  
 return;  
 }  
  
  
 Log.*d*(*TAG*, "startLocationUpdates : call FusedLocationApi.requestLocationUpdates");  
 LocationServices.*FusedLocationApi*.requestLocationUpdates(mGoogleApiClient, locationRequest, this);  
 mRequestingLocationUpdates = true;  
  
 mGoogleMap.setMyLocationEnabled(true);  
  
 }  
  
 }  
  
  
  
 private void stopLocationUpdates() {  
  
 Log.*d*(*TAG*,"stopLocationUpdates : LocationServices.FusedLocationApi.removeLocationUpdates");  
 LocationServices.*FusedLocationApi*.removeLocationUpdates(mGoogleApiClient, this);  
 mRequestingLocationUpdates = false;  
 }  
  
  
  
 @Override  
 public void onMapReady(GoogleMap googleMap) {  
  
 Log.*d*(*TAG*, "onMapReady :");  
  
 mGoogleMap = googleMap;  
  
  
 //런타임 퍼미션 요청 대화상자나 GPS 활성 요청 대화상자 보이기전에  
 //지도의 초기위치를 서울로 이동  
 setDefaultLocation();  
  
 //mGoogleMap.getUiSettings().setZoomControlsEnabled(false);  
 mGoogleMap.getUiSettings().setMyLocationButtonEnabled(true);  
 mGoogleMap.animateCamera(CameraUpdateFactory.*zoomTo*(15));  
 mGoogleMap.setOnMyLocationButtonClickListener(new GoogleMap.OnMyLocationButtonClickListener(){  
  
 @Override  
 public boolean onMyLocationButtonClick() {  
  
 Log.*d*( *TAG*, "onMyLocationButtonClick : 위치에 따른 카메라 이동 활성화");  
 mMoveMapByAPI = true;  
 return true;  
 }  
 });  
 mGoogleMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {  
  
 @Override  
 public void onMapClick(LatLng latLng) {  
  
 Log.*d*( *TAG*, "onMapClick :");  
 }  
 });  
  
 mGoogleMap.setOnCameraMoveStartedListener(new GoogleMap.OnCameraMoveStartedListener() {  
  
 @Override  
 public void onCameraMoveStarted(int i) {  
  
 if (mMoveMapByUser == true && mRequestingLocationUpdates){  
  
 Log.*d*(*TAG*, "onCameraMove : 위치에 따른 카메라 이동 비활성화");  
 mMoveMapByAPI = false;  
 }  
  
 mMoveMapByUser = true;  
  
 }  
 });  
  
  
 mGoogleMap.setOnCameraMoveListener(new GoogleMap.OnCameraMoveListener() {  
  
 @Override  
 public void onCameraMove() {  
  
  
 }  
 });  
  
 //CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngZoom(new LatLng(37.498664, 127.0280419), 14);  
  
 // googleMap.animateCamera(cameraUpdate);  
  
 BitmapDrawable bitmapdraw=(BitmapDrawable)getResources().getDrawable(R.drawable.*pillsmarker*);  
 Bitmap b=bitmapdraw.getBitmap();  
 Bitmap smallMarker = Bitmap.*createScaledBitmap*(b, 200, 200, false);  
  
 PharmParser parser = new PharmParser();  
 ArrayList<PharmDTO> list = null;  
 try {  
 list =parser.apiParserSearch();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 /\*for(PharmDTO entity : list) {  
 googleMap.addMarker(new MarkerOptions()  
 .position(new LatLng(entity.getYpos(), entity.getXpos()))  
 .title(entity.getName()))  
 .setIcon(BitmapDescriptorFactory.fromBitmap(smallMarker));  
 MarkerOptions options = new MarkerOptions();  
 options.position(new LatLng(entity.getYpos(), entity.getXpos()));  
 options.title(entity.getName());  
 options.icon(BitmapDescriptorFactory.fromResource(R.drawable.pillsmarker));  
  
 googleMap.addMarker(options);  
 }\*/  
  
 }  
  
  
 @Override  
 public void onLocationChanged(Location location) {  
  
 currentPosition  
 = new LatLng( location.getLatitude(), location.getLongitude());  
  
  
 Log.*d*(*TAG*, "onLocationChanged : ");  
  
 String markerTitle = getCurrentAddress(currentPosition);  
 String markerSnippet = "위도:" + String.*valueOf*(location.getLatitude())  
 + " 경도:" + String.*valueOf*(location.getLongitude());  
  
 //현재 위치에 마커 생성하고 이동  
 setCurrentLocation(location, markerTitle, markerSnippet);  
  
 mCurrentLocatiion = location;  
 }  
  
  
 @Override  
 protected void onStart() {  
  
 if(mGoogleApiClient != null && mGoogleApiClient.isConnected() == false){  
  
 Log.*d*(*TAG*, "onStart: mGoogleApiClient connect");  
 mGoogleApiClient.connect();  
 }  
  
 super.onStart();  
 }  
  
 @Override  
 protected void onStop() {  
  
 if (mRequestingLocationUpdates) {  
  
 Log.*d*(*TAG*, "onStop : call stopLocationUpdates");  
 stopLocationUpdates();  
 }  
  
 if ( mGoogleApiClient.isConnected()) {  
  
 Log.*d*(*TAG*, "onStop : mGoogleApiClient disconnect");  
 mGoogleApiClient.disconnect();  
 }  
  
 super.onStop();  
 }  
  
  
 @Override  
 public void onConnected(Bundle connectionHint) {  
  
  
 if ( mRequestingLocationUpdates == false ) {  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
  
 int hasFineLocationPermission = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*);  
  
 if (hasFineLocationPermission == PackageManager.*PERMISSION\_DENIED*) {  
  
 ActivityCompat.*requestPermissions*(mActivity,  
 new String[]{android.Manifest.permission.*ACCESS\_FINE\_LOCATION*},  
 *PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION*);  
  
 } else {  
  
 Log.*d*(*TAG*, "onConnected : 퍼미션 가지고 있음");  
 Log.*d*(*TAG*, "onConnected : call startLocationUpdates");  
 startLocationUpdates();  
 mGoogleMap.setMyLocationEnabled(true);  
 }  
  
 }else{  
  
 Log.*d*(*TAG*, "onConnected : call startLocationUpdates");  
 startLocationUpdates();  
 mGoogleMap.setMyLocationEnabled(true);  
 }  
 }  
 }  
  
  
 @Override  
 public void onConnectionFailed(ConnectionResult connectionResult) {  
  
 Log.*d*(*TAG*, "onConnectionFailed");  
 setDefaultLocation();  
 }  
  
  
 @Override  
 public void onConnectionSuspended(int cause) {  
  
 Log.*d*(*TAG*, "onConnectionSuspended");  
 if (cause == *CAUSE\_NETWORK\_LOST*)  
 Log.*e*(*TAG*, "onConnectionSuspended(): Google Play services " +  
 "connection lost. Cause: network lost.");  
 else if (cause == *CAUSE\_SERVICE\_DISCONNECTED*)  
 Log.*e*(*TAG*, "onConnectionSuspended(): Google Play services " +  
 "connection lost. Cause: service disconnected");  
 }  
  
  
 public String getCurrentAddress(LatLng latlng) {  
  
 //지오코더... GPS를 주소로 변환  
 Geocoder geocoder = new Geocoder(this, Locale.*getDefault*());  
  
 List<Address> addresses;  
  
 try {  
  
 addresses = geocoder.getFromLocation(  
 latlng.latitude,  
 latlng.longitude,  
 1);  
 } catch (IOException ioException) {  
 //네트워크 문제  
 Toast.*makeText*(this, "지오코더 서비스 사용불가", Toast.*LENGTH\_LONG*).show();  
 return "지오코더 서비스 사용불가";  
 } catch (IllegalArgumentException illegalArgumentException) {  
 Toast.*makeText*(this, "잘못된 GPS 좌표", Toast.*LENGTH\_LONG*).show();  
 return "잘못된 GPS 좌표";  
  
 }  
  
  
 if (addresses == null || addresses.size() == 0) {  
 Toast.*makeText*(this, "주소 미발견", Toast.*LENGTH\_LONG*).show();  
 return "주소 미발견";  
  
 } else {  
 Address address = addresses.get(0);  
 return address.getAddressLine(0).toString();  
 }  
  
 }  
  
  
 public boolean checkLocationServicesStatus() {  
 LocationManager locationManager = (LocationManager) getSystemService(*LOCATION\_SERVICE*);  
  
 return locationManager.isProviderEnabled(LocationManager.*GPS\_PROVIDER*)  
 || locationManager.isProviderEnabled(LocationManager.*NETWORK\_PROVIDER*);  
 }  
  
  
 public void setCurrentLocation(Location location, String markerTitle, String markerSnippet) {  
  
 mMoveMapByUser = false;  
  
  
 if (currentMarker != null) currentMarker.remove();  
  
  
 LatLng currentLatLng = new LatLng(location.getLatitude(), location.getLongitude());  
  
 MarkerOptions markerOptions = new MarkerOptions();  
 markerOptions.position(currentLatLng);  
 markerOptions.title(markerTitle);  
 markerOptions.snippet(markerSnippet);  
 markerOptions.draggable(true);  
  
  
 currentMarker = mGoogleMap.addMarker(markerOptions);  
  
  
 if ( mMoveMapByAPI ) {  
  
 Log.*d*( *TAG*, "setCurrentLocation : mGoogleMap moveCamera "  
 + location.getLatitude() + " " + location.getLongitude() ) ;  
 // CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngZoom(currentLatLng, 15);  
 CameraUpdate cameraUpdate = CameraUpdateFactory.*newLatLng*(currentLatLng);  
 mGoogleMap.moveCamera(cameraUpdate);  
 }  
  
 }  
  
  
 public void setDefaultLocation() {  
  
 mMoveMapByUser = false;  
  
  
 //디폴트 위치, Seoul  
 LatLng DEFAULT\_LOCATION = new LatLng(37.56, 126.97);  
 String markerTitle = "위치정보 가져올 수 없음";  
 String markerSnippet = "위치 퍼미션과 GPS 활성 요부 확인하세요";  
  
  
 if (currentMarker != null) currentMarker.remove();  
  
 MarkerOptions markerOptions = new MarkerOptions();  
 markerOptions.position(DEFAULT\_LOCATION);  
 markerOptions.title(markerTitle);  
 markerOptions.snippet(markerSnippet);  
 markerOptions.draggable(true);  
 markerOptions.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_RED*));  
 currentMarker = mGoogleMap.addMarker(markerOptions);  
  
 CameraUpdate cameraUpdate = CameraUpdateFactory.*newLatLngZoom*(DEFAULT\_LOCATION, 15);  
 mGoogleMap.moveCamera(cameraUpdate);  
  
 }  
  
  
 //여기부터는 런타임 퍼미션 처리을 위한 메소드들  
 @TargetApi(Build.VERSION\_CODES.*M*)  
 private void checkPermissions() {  
 boolean fineLocationRationale = ActivityCompat  
 .*shouldShowRequestPermissionRationale*(this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*);  
 int hasFineLocationPermission = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*);  
  
 if (hasFineLocationPermission == PackageManager  
 .*PERMISSION\_DENIED* && fineLocationRationale)  
 showDialogForPermission("앱을 실행하려면 퍼미션을 허가하셔야합니다.");  
  
 else if (hasFineLocationPermission  
 == PackageManager.*PERMISSION\_DENIED* && !fineLocationRationale) {  
 showDialogForPermissionSetting("퍼미션 거부 + Don't ask again(다시 묻지 않음) " +  
 "체크 박스를 설정한 경우로 설정에서 퍼미션 허가해야합니다.");  
 } else if (hasFineLocationPermission == PackageManager.*PERMISSION\_GRANTED*) {  
  
  
 Log.*d*(*TAG*, "checkPermissions : 퍼미션 가지고 있음");  
  
 if ( mGoogleApiClient.isConnected() == false) {  
  
 Log.*d*(*TAG*, "checkPermissions : 퍼미션 가지고 있음");  
 mGoogleApiClient.connect();  
 }  
 }  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int permsRequestCode,  
 @NonNull String[] permissions,  
 @NonNull int[] grantResults) {  
  
 if (permsRequestCode  
 == *PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION* && grantResults.length > 0) {  
  
 boolean permissionAccepted = grantResults[0] == PackageManager.*PERMISSION\_GRANTED*;  
  
 if (permissionAccepted) {  
  
  
 if ( mGoogleApiClient.isConnected() == false) {  
  
 Log.*d*(*TAG*, "onRequestPermissionsResult : mGoogleApiClient connect");  
 mGoogleApiClient.connect();  
 }  
  
  
  
 } else {  
  
 checkPermissions();  
 }  
 }  
 }  
  
  
 @TargetApi(Build.VERSION\_CODES.*M*)  
 private void showDialogForPermission(String msg) {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setTitle("알림");  
 builder.setMessage(msg);  
 builder.setCancelable(false);  
 builder.setPositiveButton("예", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 ActivityCompat.*requestPermissions*(mActivity,  
 new String[]{android.Manifest.permission.*ACCESS\_FINE\_LOCATION*},  
 *PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION*);  
 }  
 });  
  
 builder.setNegativeButton("아니오", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 finish();  
 }  
 });  
 builder.create().show();  
 }  
  
 private void showDialogForPermissionSetting(String msg) {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setTitle("알림");  
 builder.setMessage(msg);  
 builder.setCancelable(true);  
 builder.setPositiveButton("예", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 askPermissionOnceAgain = true;  
  
 Intent myAppSettings = new Intent(Settings.*ACTION\_APPLICATION\_DETAILS\_SETTINGS*,  
 Uri.*parse*("package:" + mActivity.getPackageName()));  
 myAppSettings.addCategory(Intent.*CATEGORY\_DEFAULT*);  
 myAppSettings.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 mActivity.startActivity(myAppSettings);  
 }  
 });  
 builder.setNegativeButton("아니오", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 finish();  
 }  
 });  
 builder.create().show();  
 }  
  
  
 //여기부터는 GPS 활성화를 위한 메소드들  
 private void showDialogForLocationServiceSetting() {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setTitle("위치 서비스 비활성화");  
 builder.setMessage("앱을 사용하기 위해서는 위치 서비스가 필요합니다.\n"  
 + "위치 설정을 수정하실래요?");  
 builder.setCancelable(true);  
 builder.setPositiveButton("설정", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 Intent callGPSSettingIntent  
 = new Intent(android.provider.Settings.*ACTION\_LOCATION\_SOURCE\_SETTINGS*);  
 startActivityForResult(callGPSSettingIntent, *GPS\_ENABLE\_REQUEST\_CODE*);  
 }  
 });  
 builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
 builder.create().show();  
 }  
  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 switch (requestCode) {  
  
 case *GPS\_ENABLE\_REQUEST\_CODE*:  
  
 //사용자가 GPS 활성 시켰는지 검사  
 if (checkLocationServicesStatus()) {  
 if (checkLocationServicesStatus()) {  
  
 Log.*d*(*TAG*, "onActivityResult : 퍼미션 가지고 있음");  
  
  
 if ( mGoogleApiClient.isConnected() == false ) {  
  
 Log.*d*( *TAG*, "onActivityResult : mGoogleApiClient connect ");  
 mGoogleApiClient.connect();  
 }  
 return;  
 }  
 }  
  
 break;  
 }  
 }  
  
}

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#000000">

<!-- Activity에서는 fragment로 구현해야되지만

Fragment로 구현하기위해서는 MapView를 사용해야함-->

<com.google.android.gms.maps.MapView

android:id="@+id/map"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:name="com.google.android.gms.maps.MapFragment"

/>

<android.support.v7.widget.CardView

xmlns:card\_view="http://schemas.android.com/apk/res-auto"

android:name="@+id/card\_view"

android:layout\_width="400dp"

android:layout\_height="40dp"

android:layout\_gravity="center"

card\_view:cardCornerRadius="4dp"

>

<fragment

android:id="@+id/place\_autocomplete\_fragment"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"

>

</fragment>

</android.support.v7.widget.CardView>

</RelativeLayout>

package com.example.kjh.viewpager\_fragment;

import android.Manifest;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.location.Address;

import android.location.Geocoder;

import android.location.Location;

import android.location.LocationManager;

import android.os.Build;

import android.os.Bundle;

import android.provider.Settings;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.app.ActivityCompat;

import android.support.v4.app.Fragment;

import android.support.v4.content.ContextCompat;

import android.support.v7.app.AlertDialog;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.EditText;

import android.widget.RelativeLayout;

import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;

import com.google.android.gms.common.api.GoogleApiClient;

import com.google.android.gms.common.api.PendingResult;

import com.google.android.gms.common.api.ResultCallback;

import com.google.android.gms.common.api.Status;

import com.google.android.gms.location.LocationListener;

import com.google.android.gms.location.LocationRequest;

import com.google.android.gms.location.LocationServices;

import com.google.android.gms.location.places.Place;

import com.google.android.gms.location.places.PlaceLikelihood;

import com.google.android.gms.location.places.PlaceLikelihoodBuffer;

import com.google.android.gms.location.places.Places;

import com.google.android.gms.location.places.ui.PlaceAutocompleteFragment;

import com.google.android.gms.location.places.ui.PlaceSelectionListener;

import com.google.android.gms.maps.CameraUpdateFactory;

import com.google.android.gms.maps.GoogleMap;

import com.google.android.gms.maps.MapFragment;

import com.google.android.gms.maps.MapView;

import com.google.android.gms.maps.MapsInitializer;

import com.google.android.gms.maps.OnMapReadyCallback;

import com.google.android.gms.maps.model.BitmapDescriptorFactory;

import com.google.android.gms.maps.model.LatLng;

import com.google.android.gms.maps.model.Marker;

import com.google.android.gms.maps.model.MarkerOptions;

import java.io.IOException;

import java.util.List;

import java.util.Locale;

/\*\*

\* Created by KJH on 2017-05-15.

\* Fragment Life Style

\* 1. Fragment is added

\* 2. onAttach() Fragment가 Activty에 붙을때 호출

\* 3. onCreate() Activty에서의 onCreate()와 비슷하나, ui 관련 작업은 할 수 없다.

\* 4. onCreateView() Layout을 inflater을 하여 View 작업을 하는 곳

\* 5. onActivityCreated() Activity에서 Fragment를 모두 생성하고난 다음에 호출됨. Activty의 onCreate()에서 setContentView()한 다음과 같다

\* 6. onStart() Fragment가 화면에 표시될때 호출, 사용자의 Action과 상호 작용이 불가능함

\* 7. onResume() Fragment가 화면에 완전히 그렸으며, 사용자의 Action과 상호 작용이 가능함

\* 8. Fragment is active

\* 9. User navigates backward or fragment is removed/replaced or Fragment is added to the back stack, then removed/replaced

\* 10. onPause()

\* 11. onStop() Fragment가 화면에서 더이상 보여지지 않게됬을때

\* 12. onDestroy() View 리소스를 해제할수있도록 호출. backstack을 사용했다면 Fragment를 다시 돌아갈때 onCreateView()가 호출됨

\* 13. onDetached()

\* 14. Fragment is destroyed

\*/

/\*\*

\* Google Map CallStack

\* 1. onCreate()

\* 2. onCreateView()

\* 3. onActivityCreated()

\* 4. onStart();

\* 5. onResume();

\* 5-2. onMapReady();

\* 6. onPause();

\* 7. onSaveInstanceState();

\* 8. onMapReady();

\*/

public class Fragment1 extends Fragment

implements OnMapReadyCallback ,

GoogleApiClient.ConnectionCallbacks,

GoogleApiClient.OnConnectionFailedListener,

LocationListener

{

private static final LatLng DEFAULT\_LOCATION = new LatLng(37.56, 126.97);

private static final String TAG = "googlemap\_example";

private static final int GPS\_ENABLE\_REQUEST\_CODE = 2001;

private static final int PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION = 2002;

private static final int UPDATE\_INTERVAL\_MS = 15000;

private static final int FASTEST\_UPDATE\_INTERVAL\_MS = 15000;

private GoogleMap googleMap = null;

private MapView mapView = null;

private GoogleApiClient googleApiClient = null;

private Marker currentMarker = null;

private final static int MAXENTRIES = 5;

private String[] LikelyPlaceNames = null;

private String[] LikelyAddresses = null;

private String[] LikelyAttributions = null;

private LatLng[] LikelyLatLngs = null;

public Fragment1()

{

// required

}

public void setCurrentLocation(Location location, String markerTitle, String markerSnippet) {

if ( currentMarker != null ) currentMarker.remove();

if ( location != null) {

//현재위치의 위도 경도 가져옴

LatLng currentLocation = new LatLng( location.getLatitude(), location.getLongitude());

MarkerOptions markerOptions = new MarkerOptions();

markerOptions.position(currentLocation);

markerOptions.title(markerTitle);

markerOptions.snippet(markerSnippet);

markerOptions.draggable(true);

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_BLUE));

currentMarker = this.googleMap.addMarker(markerOptions);

this.googleMap.moveCamera(CameraUpdateFactory.newLatLng(currentLocation));

return;

}

MarkerOptions markerOptions = new MarkerOptions();

markerOptions.position(DEFAULT\_LOCATION);

markerOptions.title(markerTitle);

markerOptions.snippet(markerSnippet);

markerOptions.draggable(true);

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_RED));

currentMarker = this.googleMap.addMarker(markerOptions);

this.googleMap.moveCamera(CameraUpdateFactory.newLatLng(DEFAULT\_LOCATION));

}

@Override

public void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

}

@Nullable

@Override

public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View layout = inflater.inflate(R.layout.fragment\_fragment1, container, false);

mapView = (MapView)layout.findViewById(R.id.map);

mapView.getMapAsync(this);

PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)

getActivity().getFragmentManager().findFragmentById(R.id.place\_autocomplete\_fragment);

autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {

@Override

public void onPlaceSelected(Place place) {

Location location = new Location("");

location.setLatitude(place.getLatLng().latitude);

location.setLongitude(place.getLatLng().longitude);

setCurrentLocation(location, place.getName().toString(), place.getAddress().toString());

}

@Override

public void onError(Status status) {

Log.i(TAG, "An error occurred: " + status);

}

});

return layout;

}

@Override

public void onStart() {

super.onStart();

mapView.onStart();

}

@Override

public void onStop() {

super.onStop();

mapView.onStop();

if ( googleApiClient != null && googleApiClient.isConnected())

googleApiClient.disconnect();

}

@Override

public void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

mapView.onSaveInstanceState(outState);

}

@Override

public void onResume() {

super.onResume();

mapView.onResume();

if ( googleApiClient != null)

googleApiClient.connect();

}

@Override

public void onPause() {

super.onPause();

mapView.onPause();

if ( googleApiClient != null && googleApiClient.isConnected()) {

LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient, this);

googleApiClient.disconnect();

}

}

@Override

public void onLowMemory() {

super.onLowMemory();

mapView.onLowMemory();

}

@Override

public void onDestroy() {

super.onDestroy();

mapView.onLowMemory();

if ( googleApiClient != null ) {

googleApiClient.unregisterConnectionCallbacks(this);

googleApiClient.unregisterConnectionFailedListener(this);

if ( googleApiClient.isConnected()) {

LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient, this);

googleApiClient.disconnect();

}

}

}

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

//액티비티가 처음 생성될 때 실행되는 함수

MapsInitializer.initialize(getActivity().getApplicationContext());

if(mapView != null)

{

mapView.onCreate(savedInstanceState);

}

}

@Override

public void onMapReady(GoogleMap googleMap) {

// OnMapReadyCallback implements 해야 mapView.getMapAsync(this); 사용가능. this 가 OnMapReadyCallback

this.googleMap = googleMap;

//런타임 퍼미션 요청 대화상자나 GPS 활성 요청 대화상자 보이기전에 지도의 초기위치를 서울로 이동

setCurrentLocation(null, "위치정보 가져올 수 없음", "위치 퍼미션과 GPS 활성 여부 확인");

//나침반이 나타나도록 설정

googleMap.getUiSettings().setCompassEnabled(true);

// 매끄럽게 이동함

googleMap.animateCamera(CameraUpdateFactory.zoomTo(15));

// API 23 이상이면 런타임 퍼미션 처리 필요

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

// 사용권한체크

int hasFineLocationPermission = ContextCompat.checkSelfPermission(getActivity(), Manifest.permission.ACCESS\_FINE\_LOCATION);

if ( hasFineLocationPermission == PackageManager.PERMISSION\_DENIED) {

//사용권한이 없을경우

//권한 재요청

ActivityCompat.requestPermissions(getActivity(), new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION}, PERMISSIONS\_REQUEST\_ACCESS\_FINE\_LOCATION);

} else {

//사용권한이 있는경우

if ( googleApiClient == null) {

buildGoogleApiClient();

}

if ( ActivityCompat.checkSelfPermission(getActivity(), Manifest.permission.ACCESS\_FINE\_LOCATION) == PackageManager.PERMISSION\_GRANTED)

{

googleMap.setMyLocationEnabled(true);

}

}

} else {

if ( googleApiClient == null) {

buildGoogleApiClient();

}

googleMap.setMyLocationEnabled(true);

}

}

private void buildGoogleApiClient() {

googleApiClient = new GoogleApiClient.Builder(getActivity())

.addConnectionCallbacks(this)

.addOnConnectionFailedListener(this)

.addApi(LocationServices.API)

.addApi(Places.GEO\_DATA\_API)

.addApi(Places.PLACE\_DETECTION\_API)

.enableAutoManage(getActivity(), this)

.build();

googleApiClient.connect();

}

public boolean checkLocationServicesStatus() {

LocationManager locationManager = (LocationManager) getActivity().getSystemService(Context.LOCATION\_SERVICE);

return locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER) ||

locationManager.isProviderEnabled(LocationManager.NETWORK\_PROVIDER);

}

@Override

public void onConnected(@Nullable Bundle bundle) {

if ( !checkLocationServicesStatus()) {

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

builder.setTitle("위치 서비스 비활성화");

builder.setMessage("앱을 사용하기 위해서는 위치 서비스가 필요합니다.\n" +

"위치 설정을 수정하십시오.");

builder.setCancelable(true);

builder.setPositiveButton("설정", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

Intent callGPSSettingIntent =

new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

startActivityForResult(callGPSSettingIntent, GPS\_ENABLE\_REQUEST\_CODE);

}

});

builder.setNegativeButton("취소", new DialogInterface.OnClickListener(){

@Override

public void onClick(DialogInterface dialogInterface, int i) {

dialogInterface.cancel();

}

});

builder.create().show();

}

LocationRequest locationRequest = new LocationRequest();

locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);

locationRequest.setInterval(UPDATE\_INTERVAL\_MS);

locationRequest.setFastestInterval(FASTEST\_UPDATE\_INTERVAL\_MS);

if ( Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

if ( ActivityCompat.checkSelfPermission(getActivity(),

Manifest.permission.ACCESS\_FINE\_LOCATION) == PackageManager.PERMISSION\_GRANTED) {

LocationServices.FusedLocationApi

.requestLocationUpdates(googleApiClient, locationRequest, this);

}

} else {

LocationServices.FusedLocationApi

.requestLocationUpdates(googleApiClient, locationRequest, this);

this.googleMap.getUiSettings().setCompassEnabled(true);

this.googleMap.animateCamera(CameraUpdateFactory.zoomTo(15));

}

}

@Override

public void onConnectionSuspended(int cause) {

if ( cause == CAUSE\_NETWORK\_LOST )

Log.e(TAG, "onConnectionSuspended(): Google Play services " +

"connection lost. Cause: network lost.");

else if (cause == CAUSE\_SERVICE\_DISCONNECTED )

Log.e(TAG,"onConnectionSuspended(): Google Play services " +

"connection lost. Cause: service disconnected");

}

@Override

public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

Location location = new Location("");

location.setLatitude(DEFAULT\_LOCATION.latitude);

location.setLongitude((DEFAULT\_LOCATION.longitude));

setCurrentLocation(location, "위치정보 가져올 수 없음",

"위치 퍼미션과 GPS활성 여부 확인");

}

@Override

public void onLocationChanged(Location location) {

Log.i(TAG, "onLocationChanged call..");

searchCurrentPlaces();

}

private void searchCurrentPlaces() {

@SuppressWarnings("MissingPermission")

PendingResult<PlaceLikelihoodBuffer> result = Places.PlaceDetectionApi

.getCurrentPlace(googleApiClient, null);

result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>(){

@Override

public void onResult(@NonNull PlaceLikelihoodBuffer placeLikelihoods) {

int i = 0;

LikelyPlaceNames = new String[MAXENTRIES];

LikelyAddresses = new String[MAXENTRIES];

LikelyAttributions = new String[MAXENTRIES];

LikelyLatLngs = new LatLng[MAXENTRIES];

for(PlaceLikelihood placeLikelihood : placeLikelihoods) {

LikelyPlaceNames[i] = (String) placeLikelihood.getPlace().getName();

LikelyAddresses[i] = (String) placeLikelihood.getPlace().getAddress();

LikelyAttributions[i] = (String) placeLikelihood.getPlace().getAttributions();

LikelyLatLngs[i] = placeLikelihood.getPlace().getLatLng();

i++;

if(i > MAXENTRIES - 1 ) {

break;

}

}

placeLikelihoods.release();

Location location = new Location("");

location.setLatitude(LikelyLatLngs[0].latitude);

location.setLongitude(LikelyLatLngs[0].longitude);

setCurrentLocation(location, LikelyPlaceNames[0], LikelyAddresses[0]);

}

});

}

}