APPENDIX

BridgeMenu.java

```
1.  package com.finderapp;
2.
3.  import com.baidu.navi.sdkdemo.R;
4.
5.  import android.Manifest;
6.  import android.annotation.TargetApi;
7.  import android.app.Activity;
8.  import android.content.Intent;
9.  import android.content.pm.PackageManager;
10. import android.os.Build;
11. import android.os.Bundle;
12. import android.view.View;
13. import android.view.Window;
14. import android.widget.Button;
15. import android.widget.ImageView;
16.
17. public class BridgeMenu extends Activity {
18.
19.     private final static String authBaseArr[] = { Manifest.permission.WRI
    TE_EXTERNAL_STORAGE,
20.             Manifest.permission.ACCESS_FINE_LOCATION };
21.     private final static int authBaseRequestCode = 1;
22.     ImageView v1;
23.     ImageView v2;
24.
25.     @TargetApi(23)
26.     @Override
27.     protected void onCreate(Bundle savedInstanceState) {
28.         super.onCreate(savedInstanceState);
29.         this.requestWindowFeature(Window.FEATURE_NO_TITLE);
30.         setContentView(R.layout.layout_menu);
31.
32.         v1 = (ImageView) findViewById(R.id.buttonRes);
33.         v2 = (ImageView) findViewById(R.id.buttonMos);
34.
35.         v1.setOnClickListener(new View.OnClickListener() {
36.             @Override
37.             public void onClick(View v) {
38.                 Intent in = new Intent(BridgeMenu.this, MainActivity.class);
```

```
            in.putExtra("type", 0);
            startActivity(in);
        }
    });

    v2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent in = new Intent(BridgeMenu.this, MainActivity.class);
            in.putExtra("type", 1);
            startActivity(in);
        }
    });

    if (android.os.Build.VERSION.SDK_INT >= 23) {

        if (!hasBasePhoneAuth()) {
            this.requestPermissions(authBaseArr, authBaseRequestCode);
            return;

        }
    }
}

private boolean hasBasePhoneAuth() {

    PackageManager pm = this.getPackageManager();
    for (String auth : authBaseArr) {
        if (pm.checkPermission(auth, this.getPackageName()) != Package
Manager.PERMISSION_GRANTED) {
            return false;
        }
    }
    return true;
}
}
```

BNaviMainActivity.java

```
package com.finderapp;

```

```
3.  import java.io.File;
4.  import java.util.ArrayList;
5.  import java.util.LinkedList;
6.  import java.util.List;
7.
8.  import com.baidu.navi.sdkdemo.R;
9.  import com.baidu.navisdk.adapter.BNCommonSettingParam;
10. import com.baidu.navisdk.adapter.BNOuterLogUtil;
11. import com.baidu.navisdk.adapter.BNOuterTTSPlayerCallback;
12. import com.baidu.navisdk.adapter.BNRouteGuideManager;
13. import com.baidu.navisdk.adapter.BNRoutePlanNode;
14. import com.baidu.navisdk.adapter.BNRoutePlanNode.CoordinateType;
15. import com.baidu.navisdk.adapter.BNaviCommonParams;
16. import com.baidu.navisdk.adapter.BNaviSettingManager;
17. import com.baidu.navisdk.adapter.BaiduNaviManager;
18. import com.baidu.navisdk.adapter.BaiduNaviManager.NaviInitListener;
19. import com.baidu.navisdk.adapter.BaiduNaviManager.RoutePlanListener;

20. import com.baidu.navisdk.adapter.PackageUtil;
21. import com.baidu.navisdk.adapter.base.BaiduNaviSDKProxy;
22.
23. import android.Manifest;
24. import android.annotation.TargetApi;
25. import android.app.Activity;
26. import android.content.Intent;
27. import android.content.pm.PackageManager;
28. import android.os.Build;
29. import android.os.Bundle;
30. import android.os.Environment;
31. import android.os.Handler;
32. import android.os.Message;
33. import android.util.Log;
34. import android.view.View;
35. import android.view.View.OnClickListener;
36. import android.widget.AdapterView;
37. import android.widget.AdapterView.OnItemSelectedListener;
38. import android.widget.Button;
39. import android.widget.Spinner;
40. import android.widget.TextView;
41. import android.widget.Toast;
42.
43. @TargetApi(23)
44. public class BNaviMainActivity extends Activity {
```

```
45.
46.     public static List<Activity> activityList = new LinkedList<Activity>();
47.
48.     private static final String APP_FOLDER_NAME = "BNSDKSimpleDemo";
49.
50.     private Button mWgsNaviBtn = null;
51.     private Button mGcjNaviBtn = null;
52.     private Button mBdmcNaviBtn = null;
53.     private Button mDb06ll = null;
54.     private String mSDCardPath = null;
55.
56.     public static final String ROUTE_PLAN_NODE = "routePlanNode";
57.     public static final String SHOW_CUSTOM_ITEM = "showCustomItem";
58.     public static final String RESET_END_NODE = "resetEndNode";
59.     public static final String VOID_MODE = "voidMode";
60.
61.     private final static String authBaseArr[] = { Manifest.permission.WRITE_EXTERNAL_STORAGE,
62.             Manifest.permission.ACCESS_FINE_LOCATION };
63.     private final static String authComArr[] = { Manifest.permission.READ_PHONE_STATE };
64.     private final static int authBaseRequestCode = 1;
65.     private final static int authComRequestCode = 2;
66.
67.     private boolean hasInitSuccess = false;
68.     private boolean hasRequestComAuth = false;
69.
70.     @Override
71.     protected void onCreate(Bundle savedInstanceState) {
72.         // TODO Auto-generated method stub
73.         super.onCreate(savedInstanceState);
74.
75.         activityList.add(this);
76.
77.         setContentView(R.layout.restaurant_layout);
78.
79.         Handler h = new Handler();
80.         h.postDelayed(new Runnable() {
81.
82.             @Override
```

```
83.             public void run() {
84.                 String name = Thread.currentThread().getName();
85.                 Log.i("crug", name);
86.                 delayTest();
87.             }
88.         }, 500);
89.
90.         if (initDirs()) {
91.             initNavi();
92.         }
93.     }
94.
95.     @Override
96.     protected void onResume() {
97.         super.onResume();
98.     }
99.
100.    public void delayTest() {
101.    }
102.
103.    private boolean initDirs() {
104.        mSDCardPath = getSdcardDir();
105.        if (mSDCardPath == null) {
106.            return false;
107.        }
108.        File f = new File(mSDCardPath, APP_FOLDER_NAME);
109.        if (!f.exists()) {
110.            try {
111.                f.mkdir();
112.            } catch (Exception e) {
113.                e.printStackTrace();
114.                return false;
115.            }
116.        }
117.        return true;
118.    }
119.
120.    String authinfo = null;
121.
122.    private Handler ttsHandler = new Handler() {
123.        public void handleMessage(Message msg) {
124.            int type = msg.what;
125.            switch (type) {
```

```
126.             case BaiduNaviManager.TTSPlayMsgType.PLAY_START_MSG:
    {
127.                 break;
128.             }
129.             case BaiduNaviManager.TTSPlayMsgType.PLAY_END_MSG: {

130.                 break;
131.             }
132.             default:
133.                 break;
134.             }
135.         }
136.     };
137.
138.     private BaiduNaviManager.TTSPlayStateListener ttsPlayStateListener
    = new BaiduNaviManager.TTSPlayStateListener() {
139.
140.         @Override
141.         public void playEnd() {
142.         }
143.
144.         @Override
145.         public void playStart() {
146.         }
147.     };
148.
149.     public void showToastMsg(final String msg) {
150.         BNaviMainActivity.this.runOnUiThread(new Runnable() {
151.
152.             @Override
153.             public void run() {
154.                 Toast.makeText(BNaviMainActivity.this, msg, Toast.LENGTH
    _SHORT).show();
155.             }
156.         });
157.     }
158.
159.     private boolean hasBasePhoneAuth() {
160.
161.         PackageManager pm = this.getPackageManager();
162.         for (String auth : authBaseArr) {
163.             if (pm.checkPermission(auth, this.getPackageName()) != Package
    Manager.PERMISSION_GRANTED) {
```

```
164.                 return false;
165.             }
166.         }
167.         return true;
168.     }
169.
170.     private boolean hasCompletePhoneAuth() {
171.         // TODO Auto-generated method stub
172.
173.         PackageManager pm = this.getPackageManager();
174.         for (String auth : authComArr) {
175.             if (pm.checkPermission(auth, this.getPackageName()) != Package
    Manager.PERMISSION_GRANTED) {
176.                 return false;
177.             }
178.         }
179.         return true;
180.     }
181.
182.     private void initNavi() {
183.
184.         BNOuterTTSPlayerCallback ttsCallback = null;
185.
186.         if (android.os.Build.VERSION.SDK_INT >= 23) {
187.
188.             if (!hasBasePhoneAuth()) {
189.
190.                 this.requestPermissions(authBaseArr, authBaseRequestCode);
191.                 return;
192.
193.             }
194.         }
195.
196.         BaiduNaviManager.getInstance().init(this, mSDCardPath, APP_FOL
    DER_NAME, new NaviInitListener() {
197.             @Override
198.             public void onAuthResult(int status, String msg) {
199.                 if (0 == status) {
200.                     authinfo = "";
201.                 } else {
202.                     authinfo = "" + msg;
203.                 }
204.                 BNaviMainActivity.this.runOnUiThread(new Runnable() {
```

```
205.
206.                 @Override
207.                 public void run() {
208.                     Toast.makeText(BNaviMainActivity.this, authinfo, Toast.L
    ENGTH_LONG).show();
209.                 }
210.             });
211.         }
212.
213.         public void initSuccess() {
214.             Toast.makeText(BNaviMainActivity.this, "Init Success", Toast.
    LENGTH_SHORT).show();
215.             hasInitSuccess = true;
216.             initSetting();
217.         }
218.
219.         public void initStart() {
220.             Toast.makeText(BNaviMainActivity.this, "Init Start", Toast.LE
    NGTH_SHORT).show();
221.         }
222.
223.         public void initFailed() {
224.             Toast.makeText(BNaviMainActivity.this, "", Toast.LENGTH_S
    HORT).show();
225.         }
226.
227.     }, null, ttsHandler, ttsPlayStateListener);
228.
229.   }
230.
231.   private String getSdcardDir() {
232.     if (Environment.getExternalStorageState().equalsIgnoreCase(Environ
    ment.MEDIA_MOUNTED)) {
233.       return Environment.getExternalStorageDirectory().toString();
234.     }
235.     return null;
236.   }
237.
238.   private CoordinateType mCoordinateType = null;
239.
240.   private void routeplanToNavi(CoordinateType coType) {
241.     mCoordinateType = coType;
242.     if (!hasInitSuccess) {
```

```
243.             Toast.makeText(BNaviMainActivity.this, "", Toast.LENGTH_SH
    ORT).show();
244.         }
245.
246.         if (android.os.Build.VERSION.SDK_INT >= 23) {
247.             if (!hasCompletePhoneAuth()) {
248.                 if (!hasRequestComAuth) {
249.                     hasRequestComAuth = true;
250.                     this.requestPermissions(authComArr, authComRequestCode);

251.                     return;
252.                 } else {
253.                     Toast.makeText(BNaviMainActivity.this, "", Toast.LENGTH
    _SHORT).show();
254.                 }
255.             }
256.
257.         }
258.         BNRoutePlanNode sNode = null;
259.         BNRoutePlanNode eNode = null;
260.
261.         sNode = new BNRoutePlanNode(116.300821, 40.050969, "", null, c
    oType);
262.         eNode = new BNRoutePlanNode(116.397491, 39.908749, "", null, c
    oType);
263.
264.         if (sNode != null && eNode != null) {
265.
266.             List<BNRoutePlanNode> list = new ArrayList<BNRoutePlanNod
    e>();
267.             list.add(sNode);
268.             list.add(eNode);
269.
270.             BaiduNaviManager.getInstance().launchNavigator(this, list, 1, tru
    e, new DemoRoutePlanListener(sNode),
271.                     eventListerner);
272.         }
273.     }
274.
275.     BaiduNaviManager.NavEventListener eventListerner = new BaiduNavi
    Manager.NavEventListener() {
276.
277.         @Override
```

```
278.         public void onCommonEventCall(int what, int arg1, int arg2, Bundl
     e bundle) {
279.             BNEventHandler.getInstance().handleNaviEvent(what, arg1, arg2,
     bundle);
280.         }
281.     };
282.
283.     public class DemoRoutePlanListener implements RoutePlanListener {

284.
285.         private BNRoutePlanNode mBNRoutePlanNode = null;
286.
287.         public DemoRoutePlanListener(BNRoutePlanNode node) {
288.             mBNRoutePlanNode = node;
289.         }
290.
291.         @Override
292.         public void onJumpToNavigator() {
293.
294.             for (Activity ac : activityList) {
295.
296.                 if (ac.getClass().getName().endsWith("BNDemoGuideActivity")
     ) {
297.
298.                     return;
299.                 }
300.             }
301.             Intent intent = new Intent(BNaviMainActivity.this, BNaviGuideA
     ctivity.class);
302.             Bundle bundle = new Bundle();
303.             bundle.putSerializable(ROUTE_PLAN_NODE, (BNRoutePlanNo
     de) mBNRoutePlanNode);
304.             intent.putExtras(bundle);
305.             startActivity(intent);
306.
307.         }
308.
309.         @Override
310.         public void onRoutePlanFailed() {
311.             Toast.makeText(BNaviMainActivity.this, "ç®—
     è·¯å¤±è´¥", Toast.LENGTH_SHORT).show();
312.         }
313.     }
```

```
314.
315.    private void initSetting() {
316.        BNaviSettingManager
317.                .setShowTotalRoadConditionBar(BNaviSettingManager.PreVie
    wRoadCondition.ROAD_CONDITION_BAR_SHOW_ON);
318.        BNaviSettingManager.setVoiceMode(BNaviSettingManager.VoiceM
    ode.Veteran);
319.        BNaviSettingManager.setRealRoadCondition(BNaviSettingManager.
    RealRoadCondition.NAVI_ITS_ON);
320.        BNaviSettingManager.setIsAutoQuitWhenArrived(true);
321.        Bundle bundle = new Bundle();
322.        bundle.putString(BNCommonSettingParam.TTS_APP_ID, "9354030
    ");
323.        BNaviSettingManager.setNaviSdkParam(bundle);
324.    }
325.
326.    private BNOuterTTSPlayerCallback mTTSCallback = new BNOuterT
    TSPlayerCallback() {
327.
328.        @Override
329.        public void stopTTS() {
330.            Log.e("test_TTS", "stopTTS");
331.        }
332.
333.        @Override
334.        public void resumeTTS() {
335.            Log.e("test_TTS", "resumeTTS");
336.        }
337.
338.        @Override
339.        public void releaseTTSPlayer() {
340.            Log.e("test_TTS", "releaseTTSPlayer");
341.        }
342.
343.        @Override
344.        public int playTTSText(String speech, int bPreempt) {
345.            Log.e("test_TTS", "playTTSText" + "_" + speech + "_" + bPreemp
    t);
346.
347.            return 1;
348.        }
349.
350.        @Override
```

```
351.        public void phoneHangUp() {
352.            Log.e("test_TTS", "phoneHangUp");
353.        }
354.
355.        @Override
356.        public void phoneCalling() {
357.            Log.e("test_TTS", "phoneCalling");
358.        }
359.
360.        @Override
361.        public void pauseTTS() {
362.            Log.e("test_TTS", "pauseTTS");
363.        }
364.
365.        @Override
366.        public void initTTSPlayer() {
367.            Log.e("test_TTS", "initTTSPlayer");
368.        }
369.
370.        @Override
371.        public int getTTSState() {
372.            Log.e("test_TTS", "getTTSState");
373.            return 1;
374.        }
375.    };
376.
377.    @Override
378.    public void onRequestPermissionsResult(int requestCode, String[] per
    missions, int[] grantResults) {
379.        super.onRequestPermissionsResult(requestCode, permissions, grant
    Results);
380.        if (requestCode == authBaseRequestCode) {
381.            for (int ret : grantResults) {
382.                if (ret == 0) {
383.                    continue;
384.                } else {
385.                    Toast.makeText(BNaviMainActivity.this, "", Toast.LENGTH
    _SHORT).show();
386.                    return;
387.                }
388.            }
389.            initNavi();
390.        } else if (requestCode == authComRequestCode) {
```

```
            for (int ret : grantResults) {
                if (ret == 0) {
                    continue;
                }
            }
            routeplanToNavi(mCoordinateType);
        }

    }
}
```

LocationApplication.java

```
package com.finderapp;

import com.baidu.mapapi.SDKInitializer;
import com.finderapp.service.LocationService;

import android.app.Application;
import android.app.Service;
import android.os.Vibrator;

public class LocationApplication extends Application {
    public LocationService locationService;
    public Vibrator mVibrator;

    @Override
    public void onCreate() {
        super.onCreate();

        locationService = new LocationService(getApplicationContext());
        mVibrator = (Vibrator) getApplicationContext().getSystemService(Service.VIBRATOR_SERVICE);
        SDKInitializer.initialize(getApplicationContext());

    }
}

```

PoiOverlay.java

```
1.  package com.finderapp;
2.
3.  import android.os.Bundle;
4.
5.  import com.baidu.mapapi.map.BaiduMap;
6.  import com.baidu.mapapi.map.BitmapDescriptorFactory;
7.  import com.baidu.mapapi.map.Marker;
8.  import com.baidu.mapapi.map.MarkerOptions;
9.  import com.baidu.mapapi.map.OverlayOptions;
10. import com.baidu.mapapi.map.Polyline;
11. import com.baidu.mapapi.search.poi.PoiResult;
12.
13. import java.util.ArrayList;
14. import java.util.List;
15.
16. public class PoiOverlay extends OverlayManager {
17.
18.     private static final int MAX_POI_SIZE = 10;
19.
20.     private PoiResult mPoiResult = null;
21.
22.     public PoiOverlay(BaiduMap baiduMap) {
23.         super(baiduMap);
24.     }
25.
26.     public void setData(PoiResult poiResult) {
27.         this.mPoiResult = poiResult;
28.     }
29.
30.     @Override
31.     public final List<OverlayOptions> getOverlayOptions() {
32.         if (mPoiResult == null || mPoiResult.getAllPoi() == null) {
33.             return null;
34.         }
35.         List<OverlayOptions> markerList = new ArrayList<OverlayOptions
    >();
36.         int markerSize = 0;
37.         for (int i = 0; i < mPoiResult.getAllPoi().size() && markerSize < M
    AX_POI_SIZE; i++) {
38.             if (mPoiResult.getAllPoi().get(i).location == null) {
39.                 continue;
40.             }
41.             markerSize++;
```

```
            Bundle bundle = new Bundle();
            bundle.putInt("index", i);
            markerList.add(new MarkerOptions()
                    .icon(BitmapDescriptorFactory.fromAssetWithDpi("Icon_mar
k" + markerSize + ".png")).extraInfo(bundle)
                    .position(mPoiResult.getAllPoi().get(i).location));

        }
        return markerList;
    }

    public PoiResult getPoiResult() {
        return mPoiResult;
    }

    public boolean onPoiClick(int i) {
        return false;
    }

    @Override
    public final boolean onMarkerClick(Marker marker) {
        if (!mOverlayList.contains(marker)) {
            return false;
        }
        if (marker.getExtraInfo() != null) {
            return onPoiClick(marker.getExtraInfo().getInt("index"));
        }
        return false;
    }

    @Override
    public boolean onPolylineClick(Polyline polyline) {
        return false;
    }
}
```

MainActivity.java

```
package com.finderapp;

```

```
3.  import android.annotation.TargetApi;
4.  import android.app.Activity;
5.  import android.content.Intent;
6.  import android.os.Bundle;
7.  import android.os.Environment;
8.  import android.os.Handler;
9.  import android.os.Message;
10. import android.support.v4.app.FragmentActivity;
11. import android.text.Editable;
12. import android.text.TextWatcher;
13. import android.util.Log;
14. import android.view.View;
15. import android.view.Window;
16. import android.view.View.OnClickListener;
17. import android.widget.ArrayAdapter;
18. import android.widget.AutoCompleteTextView;
19. import android.widget.Button;
20. import android.widget.EditText;
21. import android.widget.ImageView;
22. import android.widget.Toast;
23. import com.baidu.location.BDLocation;
24. import com.baidu.location.BDLocationListener;
25. import com.baidu.mapapi.map.BaiduMap;
26. import com.baidu.mapapi.map.BitmapDescriptor;
27. import com.baidu.mapapi.map.BitmapDescriptorFactory;
28. import com.baidu.mapapi.map.MapStatusUpdate;
29. import com.baidu.mapapi.map.MapStatusUpdateFactory;
30. import com.baidu.mapapi.map.MarkerOptions;
31. import com.baidu.mapapi.map.OverlayOptions;
32. import com.baidu.mapapi.map.SupportMapFragment;
33. import com.baidu.mapapi.model.LatLng;
34. import com.baidu.mapapi.model.LatLngBounds;
35. import com.baidu.mapapi.search.core.CityInfo;
36. import com.baidu.mapapi.search.core.PoiInfo;
37. import com.baidu.mapapi.search.core.SearchResult;
38. import com.baidu.mapapi.search.poi.OnGetPoiSearchResultListener;
39. import com.baidu.mapapi.search.poi.PoiCitySearchOption;
40. import com.baidu.mapapi.search.poi.PoiDetailResult;
41. import com.baidu.mapapi.search.poi.PoiDetailSearchOption;
42. import com.baidu.mapapi.search.poi.PoiIndoorResult;
43. import com.baidu.mapapi.search.poi.PoiNearbySearchOption;
44. import com.baidu.mapapi.search.poi.PoiResult;
45. import com.baidu.mapapi.search.poi.PoiSearch;
```

```
46. import com.baidu.mapapi.search.poi.PoiSortType;
47. import com.baidu.mapapi.search.sug.OnGetSuggestionResultListener;
48. import com.baidu.mapapi.search.sug.SuggestionResult;
49. import com.baidu.mapapi.search.sug.SuggestionSearch;
50. import com.baidu.mapapi.search.sug.SuggestionSearchOption;
51. import com.baidu.navi.sdkdemo.R;
52. import com.baidu.navisdk.adapter.BNCommonSettingParam;
53. import com.baidu.navisdk.adapter.BNOuterLogUtil;
54. import com.baidu.navisdk.adapter.BNOuterTTSPlayerCallback;
55. import com.baidu.navisdk.adapter.BNRoutePlanNode;
56. import com.baidu.navisdk.adapter.BNRoutePlanNode.CoordinateType;
57. import com.baidu.navisdk.adapter.BNaviSettingManager;
58. import com.baidu.navisdk.adapter.BaiduNaviManager.NaviInitListener;
59. import com.baidu.navisdk.adapter.BaiduNaviManager.RoutePlanListener;

60. import com.finderapp.service.LocationService;
61. import com.baidu.navisdk.adapter.BaiduNaviManager;
62. import java.io.File;
63. import java.util.ArrayList;
64. import java.util.LinkedList;
65. import java.util.List;
66.
67. public class MainActivity extends FragmentActivity
68.         implements OnGetPoiSearchResultListener, OnGetSuggestionResult
    Listener {
69.
70.     private final String TAG = MainActivity.this.getPackageName();
71.     private LocationService locationService;
72.     private PoiSearch mPoiSearch = null;
73.     private SuggestionSearch mSuggestionSearch = null;
74.     private BaiduMap mBaiduMap = null;
75.     private List<String> suggest;
76.     // ------------------------
77.     public static List<Activity> activityList = new LinkedList<Activity>();

78.     private static final String APP_FOLDER_NAME = "BNSDKSimpleD
    emo";
79.     private String mSDCardPath = null;
80.     public static final String ROUTE_PLAN_NODE = "routePlanNode";
81.     public static final String SHOW_CUSTOM_ITEM = "showCustomIte
    m";
82.     public static final String RESET_END_NODE = "resetEndNode";
83.     public static final String VOID_MODE = "voidMode";
```

```
84.     private final static int authBaseRequestCode = 1;
85.     private final static int authComRequestCode = 2;
86.     private boolean hasInitSuccess = false;
87.
88.     private ImageView img;
89.
90.     // ------------------- Location ----------------------------------
91.
92.     private EditText editCity = null;
93.     private AutoCompleteTextView keyWorldsView = null;
94.     private ArrayAdapter<String> sugAdapter = null;
95.     private int loadIndex = 0;
96.     LatLng center = new LatLng(32.0616667, 118.7777778);
97.     int radius = 2000;
98.     LatLng southwest = new LatLng(32.0616667, 118.7777778);
99.     LatLng northeast = new LatLng(32.0616667, 118.7777778);
100.    LatLngBounds searchbound = new LatLngBounds.Builder().include(so
      uthwest).include(northeast).build();
101.    int searchType = 0;
102.    private BDLocation mLocation = null;
103.
104.    @Override
105.    protected void onCreate(Bundle savedInstanceState) {
106.        // TODO Auto-generated method stub
107.        super.onCreate(savedInstanceState);
108.
109.        activityList.add(this);
110.        Intent t = this.getIntent();
111.        final int typeA = t.getIntExtra("type", 0);
112.        if (typeA == 0) {
113.            setContentView(R.layout.restaurant_layout);
114.
115.        } else {
116.            setContentView(R.layout.mosque_layout);
117.        }
118.
119.        img = (ImageView) this.findViewById(R.id.imageView1);
120.        img.setOnClickListener(new OnClickListener() {
121.
122.            @Override
123.            public void onClick(View v) {
124.                gotoMyLocation();
125.            }
```

```
126.
127.        });
128.
129.        Button b1 = (Button) this.findViewById(R.id.buttonList);
130.        b1.setOnClickListener(new View.OnClickListener() {
131.            @Override
132.            public void onClick(View v) {
133.                Intent in = new Intent(MainActivity.this, Recommend_List.clas
    s);
134.                in.putExtra("type", typeA);
135.                startActivity(in);
136.            }
137.        });
138.
139.        Handler h = new Handler();
140.        h.postDelayed(new Runnable() {
141.
142.            @Override
143.            public void run() {
144.                String name = Thread.currentThread().getName();
145.                Log.i("crug", name);
146.            }
147.        }, 500);
148.
149.        BNOuterLogUtil.setLogSwitcher(true);
150.
151.        if (initDirs()) {
152.            initNavi();
153.        }
154.
155.        mPoiSearch = PoiSearch.newInstance();
156.        mPoiSearch.setOnGetPoiSearchResultListener(this);
157.
158.        mSuggestionSearch = SuggestionSearch.newInstance();
159.        mSuggestionSearch.setOnGetSuggestionResultListener(this);
160.
161.        editCity = (EditText) findViewById(R.id.city);
162.        keyWorldsView = (AutoCompleteTextView) findViewById(R.id.sea
    rchkey);
163.        sugAdapter = new ArrayAdapter<String>(this, android.R.layout.sim
    ple_dropdown_item_1line);
164.        keyWorldsView.setAdapter(sugAdapter);
165.        keyWorldsView.setThreshold(1);
```

```
166.        mBaiduMap = ((SupportMapFragment) (getSupportFragmentManage
    r().findFragmentById(R.id.map))).getBaiduMap();
167.
168.        keyWorldsView.addTextChangedListener(new TextWatcher() {
169.
170.            @Override
171.            public void afterTextChanged(Editable arg0) {
172.
173.            }
174.
175.            @Override
176.            public void beforeTextChanged(CharSequence arg0, int arg1, int
    arg2, int arg3) {
177.
178.            }
179.
180.            @Override
181.            public void onTextChanged(CharSequence cs, int arg1, int arg2, i
    nt arg3) {
182.                if (cs.length() <= 0) {
183.                    return;
184.                }
185.
186.                mSuggestionSearch.requestSuggestion(
187.                        (new SuggestionSearchOption()).keyword(cs.toString()).cit
    y(editCity.getText().toString()));
188.            }
189.        });
190.
191.        // Location=================================================
    ========================================
192.        locationService = ((LocationApplication) getApplication()).locationS
    ervice;
193.        locationService.registerListener(mListener);
194.        int type = getIntent().getIntExtra("from", 0);
195.        if (type == 0) {
196.            locationService.setLocationOption(locationService.getDefaultLoca
    tionClientOption());
197.        } else if (type == 1) {
198.            locationService.setLocationOption(locationService.getOption());
199.        }
200.
201.        locationService.start();
```

```
202.     }
203.     // Location=========================================================
204. 
205.     private BDLocationListener mListener = new BDLocationListener() {
206. 
207.         @Override
208.         public void onReceiveLocation(BDLocation location) {
209.             // TODO Auto-generated method stub
210.             if (null != location && location.getLocType() != BDLocation.TypeServerError) {
211.                 mLocation = location;
212.                 myLocation();
213.                 locationService.stop();
214.                 locationService.unregisterListener(this);
215.             }
216.         }
217.     };
218. 
219.     // method draw and move location to center
220.     // screen---------------------------------------------------------------------
221.     private void myLocation() {
222.         if (mLocation != null) {
223.             LatLng point = new LatLng(mLocation.getLatitude(), mLocation.getLongitude());
224.             BitmapDescriptor bitmap = BitmapDescriptorFactory.fromResource(R.drawable.your_location);
225.             OverlayOptions option = new MarkerOptions().position(point).icon(bitmap);
226.             mBaiduMap.addOverlay(option);
227.             gotoMyLocation();
228.         }
229. 
230.     }
231. 
232.     private void gotoMyLocation() {
233. 
234.         if (mLocation != null) {
235.             LatLng point = new LatLng(mLocation.getLatitude(), mLocation.getLongitude());
236.             MapStatusUpdate status = MapStatusUpdateFactory.newLatLng(point);
237.             mBaiduMap.animateMapStatus(status);
```

```
238.         } else {
239.             Log.d(TAG, "location");
240.         }
241.     }
242.
243.     // ----------------------------------------------------------------------------
244.     protected void onDestroy() {
245.         mBaiduMap.setMyLocationEnabled(false);
246.         mBaiduMap = null;
247.         super.onDestroy();
248.     }
249.
250.     // -------------------Search City----------------------
251.     public void searchButtonProcess(View v) {
252.         searchType = 1;
253.         String citystr = editCity.getText().toString();
254.         String keystr = keyWorldsView.getText().toString();
255.         mPoiSearch.searchInCity((new PoiCitySearchOption()).city(citystr).k
    eyword(keystr).pageNum(loadIndex));
256.     }
257.
258.     public void goToNextPage(View v) {
259.         loadIndex++;
260.         searchButtonProcess(null);
261.     }
262.
263.     // ------------------------Search Nearby------------------
264.     public void searchNearbyProcess(View v) {
265.         searchType = 2;
266.         PoiNearbySearchOption nearbySearchOption = new PoiNearbySearc
    hOption()
267.                 .keyword(keyWorldsView.getText().toString()).sortType(PoiSor
    tType.distance_from_near_to_far)
268.                 .location(new LatLng(mLocation.getLatitude(), mLocation.getL
    ongitude())).radius(radius)
269.                 .pageNum(loadIndex);
270.         mPoiSearch.searchNearby(nearbySearchOption);
271.     }
272.
273.     // response from baidu get the place
274.     // respose====================================
275.     public void onGetPoiResult(PoiResult result) {
```

```
276.        if (result == null || result.error == SearchResult.ERRORNO.RESULT
    _NOT_FOUND) {
277.            Toast.makeText(MainActivity.this, "No results found", Toast.LEN
    GTH_LONG).show();
278.            return;
279.        }
280.        if (result.error == SearchResult.ERRORNO.NO_ERROR) {
281.            PoiOverlay overlay = new MyPoiOverlay(mBaiduMap);
282.            mBaiduMap.setOnMarkerClickListener(overlay);
283.            overlay.setData(result);
284.            overlay.addToMap();
285.            overlay.zoomToSpan();
286.
287.        }
288.        if (result.error == SearchResult.ERRORNO.AMBIGUOUS_KEYWO
    RD) {
289.            String strInfo = "在";
290.            for (CityInfo cityInfo : result.getSuggestCityList()) {
291.                strInfo += cityInfo.city;
292.                strInfo += ",";
293.            }
294.            strInfo += "找到结果";
295.            Toast.makeText(MainActivity.this, strInfo, Toast.LENGTH_LON
    G).show();
296.        }
297.    }
298.
299.    // get detail name of place
300.    public void onGetPoiDetailResult(PoiDetailResult result) {
301.        if (result.error != SearchResult.ERRORNO.NO_ERROR) {
302.            Toast.makeText(MainActivity.this, "Sorry, no results found", Toas
    t.LENGTH_SHORT).show();
303.        } else {
304.
305.            if (BaiduNaviManager.isNaviInited()) {
306.                routeplanToNavi(CoordinateType.WGS84,
307.                        new LatLng(result.getLocation().latitude, result.getLocatio
    n().longitude));
308.
309.            }
310.            Toast.makeText(MainActivity.this, result.getName() + ": Location
    : " + result.getAddress(),
```

```
311.                 Toast.LENGTH_SHORT).show();
312.             Toast.makeText(MainActivity.this, "Please Wait...", Toast.LENGT
    H_LONG).show();
313.             // ----------------------------------
314.         }
315.     }
316.
317.     // show red icon after search
318.     private class MyPoiOverlay extends PoiOverlay {
319.
320.         public MyPoiOverlay(BaiduMap baiduMap) {
321.             super(baiduMap);
322.         }
323.
324.         @Override
325.         public boolean onPoiClick(int index) {
326.             super.onPoiClick(index);
327.             PoiInfo poi = getPoiResult().getAllPoi().get(index);
328.             mPoiSearch.searchPoiDetail((new PoiDetailSearchOption()).poiUi
    d(poi.uid));
329.             return true;
330.         }
331.     }
332.
333.     @Override
334.     public void onGetSuggestionResult(SuggestionResult res) {
335.         if (res == null || res.getAllSuggestions() == null) {
336.             return;
337.         }
338.         suggest = new ArrayList<String>();
339.         for (SuggestionResult.SuggestionInfo info : res.getAllSuggestions())
    {
340.             if (info.key != null) {
341.                 suggest.add(info.key);
342.             }
343.         }
344.         sugAdapter = new ArrayAdapter<String>(MainActivity.this, android
    .R.layout.simple_dropdown_item_1line, suggest);
345.         keyWorldsView.setAdapter(sugAdapter);
346.         sugAdapter.notifyDataSetChanged();
347.     }
348.
349.     private boolean initDirs() {
```

```
350.        mSDCardPath = getSdcardDir();
351.        if (mSDCardPath == null) {
352.            return false;
353.        }
354.        File f = new File(mSDCardPath, APP_FOLDER_NAME);
355.        if (!f.exists()) {
356.            try {
357.                f.mkdir();
358.            } catch (Exception e) {
359.                e.printStackTrace();
360.                return false;
361.            }
362.        }
363.        return true;
364.    }
365.
366.    String authinfo = null;
367.
368.    private Handler ttsHandler = new Handler() {
369.        public void handleMessage(Message msg) {
370.            int type = msg.what;
371.            switch (type) {
372.            case BaiduNaviManager.TTSPlayMsgType.PLAY_START_MSG:
        {
373.                break;
374.            }
375.            case BaiduNaviManager.TTSPlayMsgType.PLAY_END_MSG: {

376.                break;
377.            }
378.            default:
379.                break;
380.            }
381.        }
382.    };
383.
384.    private BaiduNaviManager.TTSPlayStateListener ttsPlayStateListener
        = new BaiduNaviManager.TTSPlayStateListener() {
385.
386.        @Override
387.        public void playEnd() {
388.        }
389.
```

```
390.        @Override
391.        public void playStart() {
392.        }
393.    };
394.
395.    public void showToastMsg(final String msg) {
396.        MainActivity.this.runOnUiThread(new Runnable() {
397.
398.            @Override
399.            public void run() {
400.                Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHO
    RT).show();
401.            }
402.        });
403.    }
404.
405.    @TargetApi(23)
406.    private void initNavi() {
407.
408.        BNOuterTTSPlayerCallback ttsCallback = null;
409.
410.        BaiduNaviManager.getInstance().init(this, mSDCardPath, APP_FOL
    DER_NAME, new NaviInitListener() {
411.
412.            public void initSuccess() {
413.                Toast.makeText(MainActivity.this, "Init Success", Toast.LENG
    TH_SHORT).show();
414.                hasInitSuccess = true;
415.                initSetting();
416.            }
417.
418.            public void initStart() {
419.                Toast.makeText(MainActivity.this, "Init Start", Toast.LENGTH
    _SHORT).show();
420.            }
421.
422.            public void initFailed() {
423.                Toast.makeText(MainActivity.this, "Init Failed", Toast.LENGT
    H_SHORT).show();
424.            }
425.
426.            @Override
427.            public void onAuthResult(int arg0, String arg1) {
```

```
428.
429.             }
430.
431.         }, null, ttsHandler, ttsPlayStateListener);
432.
433.     }
434.
435.     private String getSdcardDir() {
436.         if (Environment.getExternalStorageState().equalsIgnoreCase(Environ
    ment.MEDIA_MOUNTED)) {
437.             return Environment.getExternalStorageDirectory().toString();
438.         }
439.         return null;
440.     }
441.
442.     private CoordinateType mCoordinateType = null;
443.
444.     // Navigation function
445.     @TargetApi(23)
446.     private void routeplanToNavi(CoordinateType coType, LatLng endNo
    de) {
447.         mCoordinateType = coType;
448.         if (!hasInitSuccess) {
449.             Toast.makeText(MainActivity.this, "Success", Toast.LENGTH_S
    HORT).show();
450.         }
451.
452.         BNRoutePlanNode sNode = null;
453.         BNRoutePlanNode eNode = null;
454.
455.         // -----------navigation place----------------------
456.         sNode = new BNRoutePlanNode(116.30142, 40.05087, "", null, coT
    ype);
457.         eNode = new BNRoutePlanNode(endNode.longitude, endNode.latitu
    de, "", null, coType);
458.
459.         if (sNode != null && eNode != null) {
460.             List<BNRoutePlanNode> list = new ArrayList<BNRoutePlanNod
    e>();
461.             list.add(sNode);
462.             list.add(eNode);
463.
```

```
464.            BaiduNaviManager.getInstance().launchNavigator(this, list, 1, tru
    e, new DemoRoutePlanListener(sNode),
465.                    eventListerner);
466.        }
467.    }
468.
469.    BaiduNaviManager.NavEventListener eventListerner = new BaiduNavi
    Manager.NavEventListener() {
470.
471.        @Override
472.        public void onCommonEventCall(int what, int arg1, int arg2, Bundl
    e bundle) {
473.            BNEventHandler.getInstance().handleNaviEvent(what, arg1, arg2,
    bundle);
474.        }
475.    };
476.
477.    public class DemoRoutePlanListener implements RoutePlanListener {

478.
479.        private BNRoutePlanNode mBNRoutePlanNode = null;
480.
481.        public DemoRoutePlanListener(BNRoutePlanNode node) {
482.            mBNRoutePlanNode = node;
483.        }
484.
485.        @Override
486.        public void onJumpToNavigator() {
487.            for (Activity ac : activityList) {
488.                if (ac.getClass().getName().endsWith("BNDemoGuideActivity")
    ) {
489.                    return;
490.                }
491.            }
492.            Intent intent = new Intent(MainActivity.this, BNaviGuideActivity.
    class);
493.            Bundle bundle = new Bundle();
494.            bundle.putSerializable(ROUTE_PLAN_NODE, (BNRoutePlanNo
    de) mBNRoutePlanNode);
495.            intent.putExtras(bundle);
496.            startActivity(intent);
497.
498.        }
```

```
499.
500.       @Override
501.       public void onRoutePlanFailed() {
502.           Toast.makeText(MainActivity.this, "Route Plan Failed", Toast.LE
    NGTH_SHORT).show();
503.       }
504.   }
505.
506.   private void initSetting() {
507.       BNaviSettingManager
508.               .setShowTotalRoadConditionBar(BNaviSettingManager.PreVie
    wRoadCondition.ROAD_CONDITION_BAR_SHOW_ON);
509.       BNaviSettingManager.setVoiceMode(BNaviSettingManager.VoiceM
    ode.Veteran);
510.       BNaviSettingManager.setRealRoadCondition(BNaviSettingManager.
    RealRoadCondition.NAVI_ITS_ON);
511.       BNaviSettingManager.setIsAutoQuitWhenArrived(true);
512.       Bundle bundle = new Bundle();
513.
514.       bundle.putString(BNCommonSettingParam.TTS_APP_ID, "9354030
    ");
515.       BNaviSettingManager.setNaviSdkParam(bundle);
516.   }
517.
518.   @TargetApi(23)
519.   @Override
520.   public void onRequestPermissionsResult(int requestCode, String[] per
    missions, int[] grantResults) {
521.       super.onRequestPermissionsResult(requestCode, permissions, grant
    Results);
522.       if (requestCode == authBaseRequestCode) {
523.           for (int ret : grantResults) {
524.               if (ret == 0) {
525.                   continue;
526.               } else {
527.                   Toast.makeText(MainActivity.this, "", Toast.LENGTH_SHO
    RT).show();
528.                   return;
529.               }
530.           }
531.           initNavi();
532.       } else if (requestCode == authComRequestCode) {
533.           for (int ret : grantResults) {
```

```
534.                 if (ret == 0) {
535.                     continue;
536.                 }
537.             }
538.             routeplanToNavi(mCoordinateType, null);
539.         }
540.
541.     }
542.
543.     @Override
544.     public void onGetPoiIndoorResult(PoiIndoorResult arg0) {
545.     }
546. }
```

Recommend_List.java

```
1.  package com.finderapp;
2.
3.  import java.util.ArrayList;
4.  import java.util.HashMap;
5.  import com.baidu.navi.sdkdemo.R;
6.  import android.os.Bundle;
7.  import android.view.View;
8.  import android.widget.AdapterView;
9.  import android.widget.AdapterView.OnItemClickListener;
10. import android.widget.ListView;
11. import android.widget.SimpleAdapter;
12. import android.widget.TextView;
13. import android.widget.Toast;
14. import android.app.Activity;
15. import android.content.Context;
16. import android.content.Intent;
17.
18. public class Recommend_List extends Activity {
19.
20.     ListView lv;
21.     String [] result;
22.     Context context;
23.
24.     protected void onCreate(Bundle savedInstanceState) {
25.         super.onCreate(savedInstanceState);
26.
```

```
27.         setContentView(R.layout.list_view_activity);
28.
29.         final int type=this.getIntent().getIntExtra("type", 0);
30.
31.         lv = (ListView) findViewById(R.id.listView1);
32.
33.         ArrayList<HashMap<String, Comparable>> list = new ArrayList<HashMap<String, Comparable>>();
34.
35.
36.         if(type==0){
37.             HashMap<String, Comparable> hm = new HashMap<String, Comparable>();
38.             hm.put("photo", R.drawable.saishan);
39.             hm.put("name", "塞上人家 \nSaishang Renjia");
40.             list.add(hm);
41.
42.             HashMap<String, Comparable> hm2 = new HashMap<String, Comparable>();
43.             hm2.put("photo", R.drawable.aladin);
44.             hm2.put("name", "阿拉丁风味餐
    厅 \nAlading Fengwei Canting");
45.             list.add(hm2);
46.
47.             HashMap<String, Comparable> hm3 = new HashMap<String, Comparable>();
48.             hm3.put("photo", R.drawable.anlenyuan2);
49.             hm3.put("name", "清真安乐园菜
    馆 \nQingzhen Anleyuan Caiguan");
50.             list.add(hm3);
51.
52.             HashMap<String, Comparable> hm4= new HashMap<String, Comparable>();
53.             hm4.put("photo", R.drawable.lishi);
54.             hm4.put("name", "李氏清真馆 \nLishi Qingzhenguan");
55.             list.add(hm4);
56.
57.             HashMap<String, Comparable> hm5 = new HashMap<String, Comparable>();
58.             hm5.put("photo", R.drawable.liuju);
```

```
59.             hm5.put("name", "绿柳居 \nLüv Liu Ju");
60.             list.add(hm5);
61.         }else{
62.
63.             HashMap<String, Comparable> hm = new HashMap<String, Com
    parable>();
64.             hm.put("photo", R.drawable.jingjue);
65.             hm.put("name", "Nanjing Jingjue Mosque");
66.             list.add(hm);
67.
68.             HashMap<String, Comparable> hm2 = new HashMap<String, Co
    mparable>();
69.             hm2.put("photo", R.drawable.caoqiao);
70.             hm2.put("name", "Nanjing Caoqiao Mosque");
71.             list.add(hm2);
72.
73.             HashMap<String, Comparable> hm3 = new HashMap<String, Co
    mparable>();
74.             hm3.put("photo", R.drawable.hanximen);
75.             hm3.put("name", "Nanjing Hanximen Mosque");
76.             list.add(hm3);
77.
78.             HashMap<String, Comparable> hm4= new HashMap<String, Com
    parable>();
79.             hm4.put("photo", R.drawable.jizhaoying);
80.             hm4.put("name", "Nanjing Jizhaoying Mosque");
81.             list.add(hm4);
82.         }
83.
84.
85.         SimpleAdapter adapter = new SimpleAdapter(
86.                 this,
87.                 list,
88.                 R.layout.recommend_list,
89.                 new String[]{"photo", "name", "street"},
90.                 new int[]{R.id.imageView1, R.id.textView1}
91.                 );
92.
93.         lv.setAdapter(adapter);
94.
95.         lv.setOnItemClickListener(new OnItemClickListener(){
96.
```

```
97.             @Override
98.             public void onItemClick(AdapterView<?> parent, View view, int
    position, long id) {
99.                 Intent i=new Intent(Recommend_List.this,Recommend_Explain
    .class);
100.                i.putExtra("position", position);
101.                i.putExtra("type", type);
102.                Recommend_List.this.startActivity(i);
103.            }
104.
105.        });
106.
107.    }
108. }
```

Recommend_Explain.java

```
1.  package com.finderapp;
2.
3.  import java.io.File;
4.  import java.util.ArrayList;
5.  import java.util.LinkedList;
6.  import java.util.List;
7.
8.  import com.baidu.mapapi.model.LatLng;
9.  import com.baidu.navi.sdkdemo.R;
10. import com.baidu.navisdk.adapter.BNCommonSettingParam;
11. import com.baidu.navisdk.adapter.BNOuterTTSPlayerCallback;
12. import com.baidu.navisdk.adapter.BNRoutePlanNode;
13. import com.baidu.navisdk.adapter.BNaviSettingManager;
14. import com.baidu.navisdk.adapter.BaiduNaviManager;
15. import com.baidu.navisdk.adapter.BNRoutePlanNode.CoordinateType;
16. import com.baidu.navisdk.adapter.BaiduNaviManager.NaviInitListener;
17. import com.baidu.navisdk.adapter.BaiduNaviManager.RoutePlanListener;

18.
19. import android.Manifest;
20. import android.annotation.TargetApi;
21. import android.app.Activity;
22. import android.content.Intent;
```

```
23. import android.content.pm.PackageManager;
24. import android.os.Bundle;
25. import android.os.Environment;
26. import android.os.Handler;
27. import android.os.Message;
28. import android.util.Log;
29. import android.view.View;
30. import android.view.View.OnClickListener;
31. import android.widget.Button;
32. import android.widget.ImageView;
33. import android.widget.TextView;
34. import android.widget.Toast;
35.
36. public class Recommend_Explain extends Activity {
37.     private Button btn;
38.     private TextView v;
39.     private TextView v2;
40.     private ImageView img;
41.
42.     private List<String> list1, list2, button;
43.     private List<LatLng> location_list;
44.     private List<Integer> imglist;
45.
46.     private Button btn_nav;
47.
48.     // -----------
49.     public static List<Activity> activityList = new LinkedList<Activity>();

50.     private static final String APP_FOLDER_NAME = "BNSDKSimpleDemo";
51.     private String mSDCardPath = null;
52.     public static final String ROUTE_PLAN_NODE = "routePlanNode";
53.     public static final String SHOW_CUSTOM_ITEM = "showCustomItem";
54.     public static final String RESET_END_NODE = "resetEndNode";
55.     public static final String VOID_MODE = "voidMode";
56.     private final static String authComArr[] = { Manifest.permission.READ_PHONE_STATE };
57.     private final static int authBaseRequestCode = 1;
58.     private final static int authComRequestCode = 2;
59.     private boolean hasInitSuccess = false;
60.     private boolean hasRequestComAuth = false;
61.
```

```
62.     protected void onCreate(Bundle savedInstanceState) {
63.         super.onCreate(savedInstanceState);
64.         setContentView(R.layout.recommend_explain);
65.
66.         imglist = new ArrayList<Integer>();
67.         list1 = new ArrayList<String>();
68.         list2 = new ArrayList<String>();
69.         button = new ArrayList();
70.
71.         location_list = new ArrayList<LatLng>();
72.
73.         final Intent t = this.getIntent();
74.
75.         int type = t.getIntExtra("type", 0);
76.         if (type == 0) {
77.             location_list.add(new LatLng(32.047072, 118.79148));
78.             location_list.add(new LatLng(32.041066, 118.771169));
79.             location_list.add(new LatLng(32.034929, 118.776866));
80.             location_list.add(new LatLng(32.029589, 118.780208));
81.             location_list.add(new LatLng(32.034342, 118.792285));
82.
83.             imglist.add(R.drawable.saishan);
84.             list1.add("\n 塞上人家 Saishang Renjia\n");
85.             list2.add(
86.                     "Recommended dishes: \nXinjiang-style braised chicken (大盘鸡), mutton kebab (羊肉串), braised noodles pieces with vegetables (烩面), fried lamb with cumin (孜然羊肉), beef in mini hotpot (牛肉锅仔), sheet of starch jelly in spicy sauce (大拉皮), sautéed mutton with pickled cabbage (酸菜牛肉), and sautéed shredded potatoes with chili and vinegar sauce (酸辣土豆丝).\n\n"
87.                             + "Average price per person: \n43 yuan\n\n"
88.                             + "Opening hours: \n11:00-14:00, 17:00-21:00\n\n"
89.                             + "Address: \n55 Weigang, Zhongshanmen Avenue, Xuanwu District (玄武区中山门大街卫岗 55 号)\n\n"
90.                             + "Transportation: \nTake Subway line 2 and get off at Mu Xu Yuan,, the restaurant is approximately 5-
```

```
mintue minute walk from there; or take bus 5, 36, 49, 142, 163, 805, or 814, and get off at Weigang (卫岗).\n\n");
            button.add("Navigate!");

            imglist.add(R.drawable.aladin);
            list1.add("\n 阿拉丁风味餐厅 Alading Fengwei Canting\n");
            list2.add(
                    "Recommended dishes: \nyogurt (酸奶), mutton kebab (羊肉串), special-flavored sautéed lamb with naan -bread (风味馕炒肉), Xinjiang-style braised chicken (大盘鸡), roast lamb chops (烤羊排), roasted leg of lamb (烤羊腿), noodles in spicy sauce (拌面), mashed potato with blueberry sauce (蓝莓土豆泥), grenadine juice (石榴汁), grilled buns (烤包子), stir-fried noodles pieces (丁丁炒面), and mutton soup (羊肉汤).\n\n"
                            + "Average price per person: \n43 yuan\n\n"
                            + "Opening hours: \n11:00-14:00, 17:00-21:00\n\n"
                            + "Address: \n43 Luolang Lane, Hanzhong Road, Baixia District (白下区汉中路罗廊巷 43 号)\n\n"
                            + "Transportation: \nTake Subway Lline 2, get off at Exit 3 at Shanghai stop, and walk to the first crossroad. The restaurant is approximately 20 meters from there on left.\n\n");
            button.add("Navigate!");

            imglist.add(R.drawable.anlenyuan2);
            list1.add("\n 清真安乐园菜馆 Qingzhen Anleyuan Caiguan\n");
            list2.add(
                    "Recommended dishes: \nBuns stuffed with beef (牛肉包子), buns stuffed with vegetables (菜包), beef meat balls (牛肉丸子), buns stuffed with red beanm paste (豆沙包), roast lamb chops (烤羊排), buns stuffed filled with beef and juicy soup (牛肉汤包), streamed dumplings (蒸饺), steamed dumplings stuffed with mustard (芥菜蒸饺), roasted leg of lamb (烤羊腿), and beef wontons stuffed with beef (牛肉馄饨).\n\n"
```

```
107.                    + "Average price per person: \n22 yuan\n\n"
108.                    + "Address: \n138 Wangfu Avenue, Biaxia District ((白
    下区王府大街 138 号)\n\n"
109.                    + "Transportation: \nTake bus 43 and get off at Wangfu
    Dajie (Wangfu Avenue/王府大
    街), or take bus 43 or 306, and get off at Chaotiangong Dong (east of Chao
    tian Palace/朝天宫东).\n\n");
110.            button.add("Navigate!");
111.
112.            imglist.add(R.drawable.lishi);
113.            list1.add("\n 李氏清真馆 Lishi Qingzhenguan\n");
114.            list2.add(
115.                    "Recommended dishes: \npot-
    stickerFried dumplings stuffed with beef (牛肉锅
    贴), wonton wontons stuffed with beef (牛肉馄饨), beef soup (牛肉
    汤), fried buns stuffed with beef (牛肉煎包), ox sweetbread soup (牛杂
    汤), marinated beef in spiced sauce (卤牛肉), dried spicy roast beef (牛肉
    干), plain noodles (阳春面), noodles with beef (牛肉
    面), and spicy diced bean curd (兰花干).\n\n"
116.                    + "Average price per person: \n11 yuan\n\n" + "Opening
    hours: \n05:30-19:30\n\n"
117.                    + "Address: \n1 Dading Lane, Pingshi Street, Biaxia Dist
    rict (白下区评事街打钉巷 1 号)\n\n"
118.                    + "Transportation: \nTake bus 4, 7, 23, 35, 37, or 62, and
     get off at Pingshi Jie (Pingshi Street/评事
    街), or take bus 43, 128, or 313, and get off at Dingxin Lu (Dingxin Road).
    \n\n");
119.            button.add("Navigate!");
120.
121.            imglist.add(R.drawable.liuju);
122.            list1.add("\n 绿柳居 Lüv Liu Ju\n");
123.            list2.add(
124.                    "Recommended dishes: \nbuns stuffed with vegetables (素菜
    包), pot-stickerfried beef dumplings stuffed with beef (牛肉锅
```

```
贴), sliced lotus roots in honey sauce (蜜汁糖藕), deep-
fried dried bean curd stuffed with thin mushrooms and carrots (素烧
鹅), buns with diced chicken, beef, and green asparagus (三丁
包), wontons stuffed with beef (牛肉馄饨), deep-
fried dried bean curd stuffed with dried black mushrooms and black fungu
s (素烧鸭), fried buns stuffed with beef (牛肉煎包), stir-
fried vegetables (罗汉观
斋), and marinated dried bean curd in spiced sauce (回卤干).\n\n"
                + "Average price per person: \n24 yuan\n\n" + "Opening
hours: \n07:00 — 20:00\n\n"
                + "Address: \n248 Taiping South Road, Biaxia District (
白下区太平南路 248 号)\n\n"
                + "Transportation: \nTake bus 1, Travel 2, 15, 31, 80, 31
3, 801, or 802, and get off at Yanggong Jing (Bei) (Yanggong Well (North
)/杨公井(北)).\n\n");
        button.add("Navigate!");

    } else {

        location_list.add(new LatLng(32.023994, 118.78324));
        location_list.add(new LatLng(32.029376, 118.778304));
        location_list.add(new LatLng(32.029477, 118.777898));
        location_list.add(new LatLng(32.052116, 118.785385));

        imglist.add(R.drawable.jingjue);
        list1.add("\nNanjing Jingjue Mosque\n");
        list2.add(
            "The Nanjing Jingjue mosque, also known as Sanshan Street
mosque, dates back to Hongwu Ming Dynasty. It is the most time-
honored mosque in Nanjing, which get rebuilted in the early of 4th century
 under the advise of a famous Chinese voyager, Zhenghe./n/n"
                + "Covering an area of 4000 square meters, the mosque i
ncludes a brick gate, a balcony, a prayer room, a reception room and a gue
st room etc. The gate is a brick architecture with rare carvings, looking div
ine. While the most famous remains in Jingjue mosque is the epigraphs fro
m emperors of the Ming and Qing Dynasty, which is a symbol of great ho
nor.\n\n"
```

```
142.                     + "Address :\n" + "SanShan Street, Baixia District, Nanji
    ng, Jiangsu, P.R. China\n\n"
143.                     + "Transportation :\n" + "Bus : 37, 7, 166.\n"
144.                     + "Metro : Line 1 ShanShan Jie Station Exit 3\n\n");
145.             button.add("Navigate!");
146.
147.             imglist.add(R.drawable.caoqiao);
148.             list1.add("\nNanjing Caoqiao Mosque\n");
149.             list2.add("Nanjing Caoqiao mosque is named after the place where
    it located. \n\n"
150.                     + "There were 33 mosques in Nanjing in 1958, and Caoqiao
    mosque was the important one of them. It was built in Qing Dynasty(1736
    -
    795), then unfortunately get destroyed when Taiping Heavenly Kingdom c
    ame in power. The mosque was rebuilt in Tongzhi Qing Dynasty.\n\n"
151.                     + "The main buildings of the mosque are in traditional Chines
    e architectural style, including 5 main halls, 6 Lecture halls relatively locat
    ed in north and south of the yard, 3 Water halls and other facilities. A large
     number of muslim living in this area, where Caoqiao mosque is the prime
    place for muslims to pray.\n\n"
152.                     + "Address : \n" + "No. 20 Dading Lane, Qinhuai District, Na
    njing, Jiangsu, P.R. China\n\n"
153.                     + "Transportation : \n" + "Bus : 28, 16, 302, 33, 100, 35\n"
154.                     + "Metro : Line 1 ZhangFuYuan Station Exit 3\n\n");
155.             button.add("Navigate!");
156.
157.             imglist.add(R.drawable.hanximen);
158.             list1.add("\nNanjing Hanximen Mosque\n");
159.             list2.add(
160.                     "Hanximen Mosque dates back to Tianqi Ming Dynasty featur
    ing Chinese classical architectural style with walls and doors in Arabic styl
    e. The main hall covers an area of 320 square meters.\n\n"
161.                     + "It was once destroyed when Taiping Heavenly Kingdo
    m came in power, and get reconstructed in the Tongzhi Qing Dynasty.\n\n
    "
162.                     + "In 1915, Mr Mayanshu raised funds to rebuild the pra
    yer hall, also built memorial archway, moon floor, Jingshui hall, lecture ha
    ll, dormitories and other buildings.\n\n"
163.                     + "Address : \n" + "No. 43 Jizhaoying (Fish Street Alley)
    , Nanjing, Jiangsu, P.R. China\n\n"
164.                     + "Transportation : \n" + "Bus : 100, 25 \n"
165.                     + "Metro : Line 1 ZhuJiangLu Station, Exit 3\n\n");
166.             button.add("Navigate!");
```

```
167.
168.         imglist.add(R.drawable.jizhaoying);
169.         list1.add("\nNanjing Jizhaoying Mosque\n");
170.         list2.add(
171.                 "Nanjing Jizhaoying Mosque covers an area of 815.1 square
    meters. Although there are no written record about the exactly time when t
    he mosque built, it is believed that the mosque was constructed in the midd
    le of the Qing Dynasty.\n\n"
172.                         + "The ancient mosque was once burned when Taiping H
    eavenly Kingdom ( 1853-
    1864) came in power. However, it still strived functioning effectively. The
     religious activities last for decades under very difficult circumstances.\n\n
    "
173.                         + "In the beginning of the Republic of China, some celeb
    rities raised money and rebuilt the the prayer hall in 1912. In 1987, the Urb
    an Islamic Association in Nanning invested 30,000 yuan to repair the mos
    que in total. It is with efforts of oodles of kind people, the old mosque can
     still functions well. \n\n"
174.                         + "Address : \n" + "No. 43, Jizhaoying, Xuanwu District,
     Nanjing, Jiangsu, P.R. China\n\n");
175.         button.add("Navigate!");
176.
177.     }
178.
179.     v = (TextView) findViewById(R.id.textView1);
180.     v2 = (TextView) findViewById(R.id.textView2);
181.     img = (ImageView) findViewById(R.id.imageView1);
182.     btn = (Button) findViewById(R.id.button1);
183.
184.     img.setImageResource(imglist.get(t.getIntExtra("position", 0)));
185.     v.setText(list1.get(t.getIntExtra("position", 0)));
186.     v2.setText(list2.get(t.getIntExtra("position", 0)));
187.     btn.setText(button.get(t.getIntExtra("position", 0)));
188.
189.     btn_nav = (Button) findViewById(R.id.button1);
190.     btn_nav.setOnClickListener(new OnClickListener() {
191.
192.         @Override
193.         public void onClick(View v) {
194.             if (BaiduNaviManager.isNaviInited()) {
195.                 routeplanToNavi(CoordinateType.WGS84, location_list.get(t.
    getIntExtra("position", 0)));
196.
```

```
197.             }
198.
199.         }
200.
201.     });
202.
203. }
204.
205. private boolean initDirs() {
206.     mSDCardPath = getSdcardDir();
207.     if (mSDCardPath == null) {
208.         return false;
209.     }
210.     File f = new File(mSDCardPath, APP_FOLDER_NAME);
211.     if (!f.exists()) {
212.         try {
213.             f.mkdir();
214.         } catch (Exception e) {
215.             e.printStackTrace();
216.             return false;
217.         }
218.     }
219.     return true;
220. }
221.
222. String authinfo = null;
223.
224. private Handler ttsHandler = new Handler() {
225.     public void handleMessage(Message msg) {
226.         int type = msg.what;
227.         switch (type) {
228.         case BaiduNaviManager.TTSPlayMsgType.PLAY_START_MSG:
    {
229.             break;
230.         }
231.         case BaiduNaviManager.TTSPlayMsgType.PLAY_END_MSG: {

232.             break;
233.         }
234.         default:
235.             break;
236.         }
237.     }
```

```
238.     };
239.
240.     private BaiduNaviManager.TTSPlayStateListener ttsPlayStateListener
     = new BaiduNaviManager.TTSPlayStateListener() {
241.
242.         @Override
243.         public void playEnd() {
244.         }
245.
246.         @Override
247.         public void playStart() {
248.         }
249.     };
250.
251.     public void showToastMsg(final String msg) {
252.         Recommend_Explain.this.runOnUiThread(new Runnable() {
253.
254.             @Override
255.             public void run() {
256.                 Toast.makeText(Recommend_Explain.this, msg, Toast.LENGT
     H_SHORT).show();
257.             }
258.         });
259.     }
260.
261.     private boolean hasCompletePhoneAuth() {
262.         PackageManager pm = this.getPackageManager();
263.         for (String auth : authComArr) {
264.             if (pm.checkPermission(auth, this.getPackageName()) != Package
     Manager.PERMISSION_GRANTED) {
265.                 return false;
266.             }
267.         }
268.         return true;
269.     }
270.
271.     @TargetApi(23)
272.     private void initNavi() {
273.
274.         BNOuterTTSPlayerCallback ttsCallback = null;
275.
276.         BaiduNaviManager.getInstance().init(this, mSDCardPath, APP_FOL
     DER_NAME, new NaviInitListener() {
```

```
277.
278.            public void initSuccess() {
279.                Toast.makeText(Recommend_Explain.this, "Init Success", Toas
    t.LENGTH_SHORT).show();
280.                hasInitSuccess = true;
281.                initSetting();
282.            }
283.
284.            public void initStart() {
285.                Toast.makeText(Recommend_Explain.this, "Init Start", Toast.L
    ENGTH_SHORT).show();
286.            }
287.
288.            public void initFailed() {
289.                Toast.makeText(Recommend_Explain.this, "Init Failed", Toast.
    LENGTH_SHORT).show();
290.            }
291.
292.            @Override
293.            public void onAuthResult(int arg0, String arg1) {
294.
295.            }
296.
297.        }, null, ttsHandler, ttsPlayStateListener);
298.
299.    }
300.
301.    private String getSdcardDir() {
302.        if (Environment.getExternalStorageState().equalsIgnoreCase(Environ
    ment.MEDIA_MOUNTED)) {
303.            return Environment.getExternalStorageDirectory().toString();
304.        }
305.        return null;
306.    }
307.
308.    private CoordinateType mCoordinateType = null;
309.
310.    @TargetApi(23)
311.    private void routeplanToNavi(CoordinateType coType, LatLng endNo
    de) {
312.        mCoordinateType = coType;
313.        if (!hasInitSuccess) {
```

```
314.            Toast.makeText(Recommend_Explain.this, "Success", Toast.LENGTH_SHORT).show();
315.        }
316.
317.        BNRoutePlanNode sNode = null;
318.        BNRoutePlanNode eNode = null;
319.
320.        // -----------navigation place-----------------------
321.        sNode = new BNRoutePlanNode(116.30142, 40.05087, "", null, coType);
322.        eNode = new BNRoutePlanNode(endNode.longitude, endNode.latitude, "", null, coType);
323.
324.        if (sNode != null && eNode != null) {
325.            List<BNRoutePlanNode> list = new ArrayList<BNRoutePlanNode>();
326.            list.add(sNode);
327.            list.add(eNode);
328.
329.            BaiduNaviManager.getInstance().launchNavigator(this, list, 1, true, new DemoRoutePlanListener(sNode),
330.                    eventListerner);
331.        }
332.
333.    }
334.
335.    BaiduNaviManager.NavEventListener eventListerner = new BaiduNaviManager.NavEventListener() {
336.
337.        @Override
338.        public void onCommonEventCall(int what, int arg1, int arg2, Bundle bundle) {
339.            BNEventHandler.getInstance().handleNaviEvent(what, arg1, arg2, bundle);
340.        }
341.    };
342.
343.    public class DemoRoutePlanListener implements RoutePlanListener {

344.
345.        private BNRoutePlanNode mBNRoutePlanNode = null;
346.
347.        public DemoRoutePlanListener(BNRoutePlanNode node) {
```

```
348.            mBNRoutePlanNode = node;
349.        }
350.
351.        @Override
352.        public void onJumpToNavigator() {
353.            for (Activity ac : activityList) {
354.                if (ac.getClass().getName().endsWith("BNDemoGuideActivity")
    ) {
355.                    return;
356.                }
357.            }
358.            Intent intent = new Intent(Recommend_Explain.this, BNaviGuide
    Activity.class);
359.            Bundle bundle = new Bundle();
360.            bundle.putSerializable(ROUTE_PLAN_NODE, (BNRoutePlanNo
    de) mBNRoutePlanNode);
361.            intent.putExtras(bundle);
362.            startActivity(intent);
363.
364.        }
365.
366.        @Override
367.        public void onRoutePlanFailed() {
368.            Toast.makeText(Recommend_Explain.this, "Route Plan Failed", T
    oast.LENGTH_SHORT).show();
369.        }
370.    }
371.
372.    private void initSetting() {
373.        BNaviSettingManager
374.                .setShowTotalRoadConditionBar(BNaviSettingManager.PreVie
    wRoadCondition.ROAD_CONDITION_BAR_SHOW_ON);
375.        BNaviSettingManager.setVoiceMode(BNaviSettingManager.VoiceM
    ode.Veteran);
376.        BNaviSettingManager.setRealRoadCondition(BNaviSettingManager.
    RealRoadCondition.NAVI_ITS_ON);
377.        BNaviSettingManager.setIsAutoQuitWhenArrived(true);
378.        Bundle bundle = new Bundle();
379.
380.        bundle.putString(BNCommonSettingParam.TTS_APP_ID, "9354030
    ");
381.        BNaviSettingManager.setNaviSdkParam(bundle);
382.    }
```

```
383.
384.     private BNOuterTTSPlayerCallback mTTSCallback = new BNOuterT
         TSPlayerCallback() {
385.
386.         @Override
387.         public void stopTTS() {
388.             Log.e("test_TTS", "stopTTS");
389.         }
390.
391.         @Override
392.         public void resumeTTS() {
393.             Log.e("test_TTS", "resumeTTS");
394.         }
395.
396.         @Override
397.         public void releaseTTSPlayer() {
398.             Log.e("test_TTS", "releaseTTSPlayer");
399.         }
400.
401.         @Override
402.         public int playTTSText(String speech, int bPreempt) {
403.             Log.e("test_TTS", "playTTSText" + "_" + speech + "_" + bPreemp
         t);
404.
405.             return 1;
406.         }
407.
408.         @Override
409.         public void phoneHangUp() {
410.             Log.e("test_TTS", "phoneHangUp");
411.         }
412.
413.         @Override
414.         public void phoneCalling() {
415.             Log.e("test_TTS", "phoneCalling");
416.         }
417.
418.         @Override
419.         public void pauseTTS() {
420.             Log.e("test_TTS", "pauseTTS");
421.         }
422.
423.         @Override
```

```
424.        public void initTTSPlayer() {
425.            Log.e("test_TTS", "initTTSPlayer");
426.        }
427.
428.        @Override
429.        public int getTTSState() {
430.            Log.e("test_TTS", "getTTSState");
431.            return 1;
432.        }
433.    };
434.
435.    @TargetApi(23)
436.    @Override
437.    public void onRequestPermissionsResult(int requestCode, String[] per
    missions, int[] grantResults) {
438.        super.onRequestPermissionsResult(requestCode, permissions, grant
    Results);
439.        if (requestCode == authBaseRequestCode) {
440.            for (int ret : grantResults) {
441.                if (ret == 0) {
442.                    continue;
443.                } else {
444.                    Toast.makeText(Recommend_Explain.this, "", Toast.LENGT
    H_SHORT).show();
445.                    return;
446.                }
447.            }
448.            initNavi();
449.        } else if (requestCode == authComRequestCode) {
450.            for (int ret : grantResults) {
451.                if (ret == 0) {
452.                    continue;
453.                }
454.            }
455.            routeplanToNavi(mCoordinateType, null);
456.        }
457.
458.    }
459. }
```