Android Inappbilling Simple Usage

MainActivity

```
public class MainActivity extends AppCompatActivity implements
IabBroadcastReceiver.IabBroadcastListener {
     private InAppBillingHelper inappbillingHelper;
     private static final int IN_APP_BILLING_REQUEST_CODE = 10001;
     private static final String SKU_TEST = "android.test.purchased"
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         initInappBillingHelper();
         btnClick.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {
                 inappbillingHelper.launchPurchase(SKU_TEST);
             }
         });
     }

     private void initInappBillingHelper() {
         inappbillingHelper= new InAppBillingHelper(this,IN_APP_BILLING_REQUEST_CODE, new
InAppBillingHelperResponse() {
             @Override
             public void success(Purchase purchase) {
                 appBillingSuccess(purchase);
             }

             @Override
             public void fail() {
                 appBillingFail();
             }

             @Override
             public void onActivityResult(int requestCode, int resultCode, Intent data) {
                 MainActivity.super.onActivityResult(requestCode,resultCode,data);
             }
         });
     }

     private void appBillingSuccess(Purchase purchase) {
         Toast.makeText(getApplicationContext(), "Satın alma başarılı",
Toast.LENGTH_LONG).show();
     }
     private void appBillingFail() {
         Toast.makeText(getApplicationContext(), "Satın alma başarısız",
Toast.LENGTH_LONG).show();
     }
     public void onResume() {
         super.onResume();
         inAppBillingHelper.registerReceiver();
     }
     @Override
     public void onPause() {
         super.onPause();
         inAppBillingHelper.unregisterReceiver();
     }
```

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == IN_APP_BILLING_REQUEST_CODE)
        inappbillingHelper.onActivityResult(requestCode, resultCode, data);
        else super.onActivityResult(requestCode,resultCode,data);
    }
    @Override
    public void receivedBroadcast() {
        inappbillingHelper.receivedBroadcast();
    }
}
```

InAppBillingHelper

```
package com.dinibilgiyarismasiapp.Utils;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import com.dinibilgiyarismasiapp.MainActivity;
import com.dinibilgiyarismasiapp.R;
import com.dinibilgiyarismasiapp.inappbilling.IabBroadcastReceiver;
import com.dinibilgiyarismasiapp.inappbilling.IabHelper;
import com.dinibilgiyarismasiapp.inappbilling.IabResult;
import com.dinibilgiyarismasiapp.inappbilling.Inventory;
import com.dinibilgiyarismasiapp.inappbilling.Purchase;

import java.util.Date;
import java.util.UUID;

public class InAppBillingHelper {
    private static final String TAG = "InAppBilling"
    private static final String SKU_TEST = "android.test.purchased"
    private IabBroadcastReceiver mBroadcastReceiver;
    private IabHelper mHelper;
    private AppCompatActivity appCompatActivity;
    private InAppBillingHelperResponse inappbillingHelperResponse;
    private int requestCode;
    public InAppBillingHelper(AppCompatActivity appCompatActivity, int requestCode,
InAppBillingHelperResponse inappbillingHelperResponse) {
        this.appCompatActivity= appCompatActivity;
        this.inappbillingHelperResponse= inappbillingHelperResponse;
        this.requestCode=requestCode;
        initIabHelper();
    }

    public void launchPurchase(String skuType) {
        try {
            mHelper.launchPurchaseFlow(this.appCompatActivity, skuType, this.requestCode,
                    mPurchaseFinishedListener, UUID.randomUUID().toString());
        } catch (Exception e) {
            complain(e.getMessage());
        }
    }

    private IabHelper.OnIabPurchaseFinishedListener mPurchaseFinishedListener = new IabHelper
            .OnIabPurchaseFinishedListener() {
        public void onIabPurchaseFinished(IabResult result, Purchase purchase) {
            Log.d(TAG, "Purchase finished: " + result + ", purchase: " + purchase);
```

```
50
51              // if we were disposed of in the meantime, quit.
52              if (mHelper == null) return;
53
54              if (result.isFailure()) {
55
56                  complain("Satın alma hatası: " + result);
57                  setWaitScreen(false);
58                  return;
59              }
60              if (!verifyDeveloperPayload(purchase)) {
61                  complain("Satın alma hatası. Authenticity verification failed.");
62                  setWaitScreen(false);
63                  return;
64              }
65
66              Log.d(TAG, "Purchase successful.");
67              try {
68                  mHelper.consumeAsync(purchase, mConsumeFinishedListener);
69              } catch (IabHelper.IabAsyncInProgressException e) {
70                  complain("Another async operation in progress.");
71                  setWaitScreen(false);
72              }
73          }
74      };
75      private IabHelper.OnConsumeFinishedListener mConsumeFinishedListener = new IabHelper
76              .OnConsumeFinishedListener() {
77          public void onConsumeFinished(Purchase purchase, IabResult result) {
78              Log.d(TAG, "Consumption finished. Purchase: " + purchase + ", result: " + result);
79
80              // if we were disposed of in the meantime, quit.
81              if (mHelper == null) return;
82
83              // We know this is the "gas" sku because it's the only one we consume,
84              // so we don't check which sku was consumed. If you have more than one
85              // sku, you probably should check...
86              if (result.isSuccess()) {
87                  // successfully consumed, so we apply the effects of the item in our
88                  // game world's logic, which in our case means filling the gas tank a bit
89                  Log.d(TAG, "Consumption successful. Provisioning.");
90                  paymentSuccess(purchase);
91                  //mTank = mTank == TANK_MAX ? TANK_MAX : mTank + 1;
92                  // saveData();
93                  // alert("You filled 1/4 tank. Your tank is now " + String.valueOf(mTank) +
94  "/4
95                  // full!");
96              } else {
97                  complain("Error while consuming: " + result);
98                  paymentFail();
99              }
100             //updateUi();
101             setWaitScreen(false);
102             Log.d(TAG, "End consumption flow.");
103         }
104     };
105
106     private void paymentSuccess(Purchase purchase) {
107         this.inappbillingHelperResponse.success(purchase);
108         alert("Payment success. " + purchase.getOrderId());
109         if (SKU_TEST.equals(purchase.getSku())) {
110             SharedPreferencesUtil.writeLong(appCompatActivity, "REKLAM_KADIRMA_ZAMANI",
111                     new Date().getTime());
112         }
113     }
114
115     private void paymentFail() {
116         alert("Payment fail");
```

```
117             this.inappbillingHelperResponse.fail();
118         }
119 
120         private void initIabHelper() {
121             mHelper = new IabHelper(appCompatActivity,
122 this.appCompatActivity.getString(R.string.api_key));
123             mHelper.startSetup(new IabHelper.OnIabSetupFinishedListener() {
124                 public void onIabSetupFinished(IabResult result) {
125                     Log.d(TAG, "Setup finished.");
126 
127                     if (!result.isSuccess()) {
128                         // Oh noes, there was a problem.
129                         complain("Problem setting up in-app billing: " + result);
130                         return;
131                     }
132 
133                     // Have we been disposed of in the meantime? If so, quit.
134                     if (mHelper == null) return;
135 
136                     // IAB is fully set up. Now, let's get an inventory of stuff we own.
137                     Log.d(TAG, "Setup successful. Querying inventory.");
138                     try {
139                         mHelper.queryInventoryAsync(mGotInventoryListener);
140                     } catch (IabHelper.IabAsyncInProgressException e) {
141                         complain("Error querying inventory. Another async operation in
142 progress.");
143                     }
144                 }
145             });
146         }
147 
148         private IabHelper.QueryInventoryFinishedListener mGotInventoryListener = new IabHelper
149                 .QueryInventoryFinishedListener() {
150             public void onQueryInventoryFinished(IabResult result, Inventory inventory) {
151                 Log.d(TAG, "Query inventory finished.");
152 
153                 // Have we been disposed of in the meantime? If so, quit.
154                 if (mHelper == null) return;
155 
156                 // Is it a failure?
157                 if (result.isFailure()) {
158                     complain("Failed to query inventory: " + result);
159                     return; //todo comment out return
160                 }
161 
162                 Log.d(TAG, "Query inventory was successful.");
163 
164                 /*
165                  * Check for items we own. Notice that for each purchase, we check
166                  * the developer payload to see if it's correct! See
167                  * verifyDeveloperPayload().
168                  */
169 
170                 // Do we have the premium upgrade?
171                 Purchase testPurchase = inventory.getPurchase(SKU_TEST);
172                 boolean isTest = (testPurchase != null && verifyDeveloperPayload(testPurchase));
173 
174                 // Check for gas delivery -- if we own gas, we should fill up the tank immediately
175 
176                 if (isTest) {
177                     Log.d(TAG, "We have gas. Consuming it.");
178                     try {
179                         mHelper.consumeAsync(inventory.getPurchase(SKU_TEST),
180 mConsumeFinishedListener);
181                     } catch (IabHelper.IabAsyncInProgressException e) {
182                         complain("Error consuming gas. Another async operation in progress.");
183                     }
```

```
                return;
            }

            setWaitScreen(false);
            Log.d(TAG, "Initial inventory query finished; enabling main UI.");
        }
    };

    boolean verifyDeveloperPayload(Purchase p) {
        String payload = p.getDeveloperPayload();

        /*
         * TODO: verify that the developer payload of the purchase is correct. It will be
         * the same one that you sent when initiating the purchase.
         *
         * WARNING: Locally generating a random string when starting a purchase and
         * verifying it here might seem like a good approach, but this will fail in the
         * case where the user purchases an item on one device and then uses your app on
         * a different device, because on the other device you will not have access to the
         * random string you originally generated.
         *
         * So a good developer payload has these characteristics:
         *
         * 1. If two different users purchase an item, the payload is different between them,
         * so that one user's purchase can't be replayed to another user.
         *
         * 2. The payload must be such that you can verify it even when the app wasn't the
         * one who initiated the purchase flow (so that items purchased by the user on
         * one device work on other devices owned by the user).
         *
         * Using your own server to store and verify developer payloads across app
         * installations is recommended.
         */

        return true;
    }

    void setWaitScreen(boolean set) {
        // findViewById(R.id.screen_main).setVisibility(set ? View.GONE : View.VISIBLE);
        //findViewById(R.id.screen_wait).setVisibility(set ? View.VISIBLE : View.GONE);
    }

    void complain(String message) {
        Log.e(TAG, "**** TrivialDrive Error: " + message);
        alert("Error: " + message);
    }

    void alert(String message) {
        AlertDialog.Builder bld = new AlertDialog.Builder(appCompatActivity);
        bld.setMessage(message);
        bld.setNeutralButton("Tamam", null);
        Log.d(TAG, "Showing alert dialog: " + message);
        bld.create().show();
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        Log.d(TAG, "onActivityResult(" + requestCode + "," + resultCode + "," + data);
        if (mHelper == null) return;

        // Pass on the activity result to the helper for handling
        if (!mHelper.handleActivityResult(requestCode, resultCode, data)) {
            // not handled, so handle it ourselves (here's where you'd
            // perform any handling of activity results not related to in-app
            // billing...
            inappbillingHelperResponse.onActivityResult(requestCode, resultCode, data);
        } else {
            Log.d(TAG, "onActivityResult handled by IABUtil.");
```

```
            }
        }

251     public void receivedBroadcast() {
252         Log.d(TAG, "Received broadcast notification. Querying inventory.");
253         try {
254             mHelper.queryInventoryAsync(mGotInventoryListener);
255         } catch (IabHelper.IabAsyncInProgressException e) {
256             complain("Error querying inventory. Another async operation in progress.");
257         }
258     }
259     public void registerReceiver(){
260         mBroadcastReceiver = new
261 IabBroadcastReceiver((IabBroadcastReceiver.IabBroadcastListener) appCompatActivity);
262         IntentFilter broadcastFilter = new IntentFilter(IabBroadcastReceiver.ACTION);
263         appCompatActivity.registerReceiver(mBroadcastReceiver,broadcastFilter);
264     }
265     public void unregisterReceiver(){
            appCompatActivity.unregisterReceiver(mBroadcastReceiver);
        }
    }
```

InAppBilllingHelperResponse

```
1   package com.dinibilgiyarismasiapp.Utils;
2
3   import android.content.Intent;
4
5   import com.dinibilgiyarismasiapp.inappbilling.Purchase;
6
7   public interface InAppBillingHelperResponse {
8       void success(Purchase purchase);
9       void fail();
10      void onActivityResult(int requestCode, int resultCode, Intent data);
11  }
```