Simple game

Easy to be addicted

UI eye-catching

More features

Play with friends

2048 PETS

FINAL PROJECT DSA

HomiesHCMIU

The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048; however, you can keep playing the game, creating tiles with larger numbers. The game is won when a tile with a value of 2048 appears on the board, hence the name of the game. After reaching the 2048 tile, players can continue to play (beyond the 2048 tile) to reach higher scores. When the player has no legal moves.

GAME DEVELOPER

Vu Minh Tu Anh

Luu Tuan Hung

Bui Ngoc Thanh Hien

Ho Hai Nguyen

ADVISOR

Prof. Tran Thanh Tung

Contents

CHAP 1: INTRODUCTION

1.1 Gaming in the Field:

In the fast growing field of software engineering and development, video game development is unique yet similar to other software endeavors. It is unique in that it combines the work of teams covering multiple disciplines (art, music, acting, programming, etc.), and that engaging game play is sought after through the use of prototypes and iterations.

We are working with 2048 game as our game project for the “Data Structure and Algorithms” subject is a four - credit course and as part of our degree. We choose this type of work for doing better with development cycle, development period, graphics. We have chance to improve java skills and practice theory we have learn on class (***object oriented programming, singleton, stack***…)

1.2 About the game project:

There are such similar games as 2048 in CH play and the web games.

However, they runs almost the same operation: players start playing game to achieve the highest scores, which may cause boredom for players thereafter.

Recognizing that problem, our team has remade the game 2048 with a fresher, up-to-date interface, adding more features such as saving scores by logging in Facebook, stores to buy more features, personal profile attaching with avatar bought from stores.

To put it in a nutshell, with an aim to create a more appealing game to users so that we can earn the user loyalty, we create our game 2048

1.3 Our 2048 Pets game:

As mentioned that we will add more features to our 2048 game to make the User get more excited while enjoying the game.

So firstly we have the basic rule is to slide numbered tiles (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096,..) on a grid to combine them to create a tile with the number 2048; however, you can keep playing the game, creating tiles with larger numbers.

Then, we add some features to the game:

- Login Facebook to save your score.
- Go shopping with the store:
  - "Undo" and "Hammer" with the score you get.
  - "Avatar" for your profile
  - "Theme" for game
- Add more sounds and animation to make the game more attractive.
- Apply Stack, singleton pattern, adapter pattern and some other OOP skills.

1.4 References:

- Images from https://www.google.com.vn/
- https://developer.android.com/reference/android/app
- https://stackoverflow.com
- Tutorial Android App Development for Beginner
  https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl
  https://www.youtube.com/playlist?list=PLzrVYRai0riSRJ3M3bifVWWRq5eJMu6tv

- Facebook connect
  https://developers.facebook.com/docs/facebook-login/android

1.5 Developer team:

Homies team is a Data Structure Algorithm project team. We have 4 members come from International University, Computer Science major:

Name - Github username	**UID**	**Contribute**
Luu Tuan Hung - *Hiroyughi*	ITITIU15034	Write report Shopping activity Draw class diagram
Ho Hai Nguyen - *NguyenHoHai*	ITITIU15104	Write report Design Motion event Support main function of HandleGame
Bui Ngoc Thanh Hien - *thanhhien191097*	ITITIU1503	Write report Design UI and effect Write Menu Activity
Vu Minh Tu Anh - *suhymin97*	ITITIU15093	Write report Draw class diagram Support every part of the project Fix bugs

CHAP 2: SOFTWARE REQUIREMENTS

2.1 What we have:

1. User friendly efficient and lucrative system.
2. Minimum maintenance cost (graphics).
3. Availability of expected requirements within the PC/mobile configuration. (with Android Studio)
4. Easy to operate.
5. With measured coding and professional thinking.

2.2 What we want:

1. Develop system within limited cost.
2. Maximum high definition.
3. Design whole system with efficient manner.
4. Provide an international player rank list.
5. Easy to update.

2.3 Working tools, platform:

1. Android Studio (minimum API: 17, target API 26) with addition library

    a. Facebook sdk

```
implementation 'com.facebook.android:facebook-android-sdk:[4,5)'
```

    b. Circle Image View

```
implementation 'de.hdodenhof:circleimageview:2.2.0'
```

    c. Picasso

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

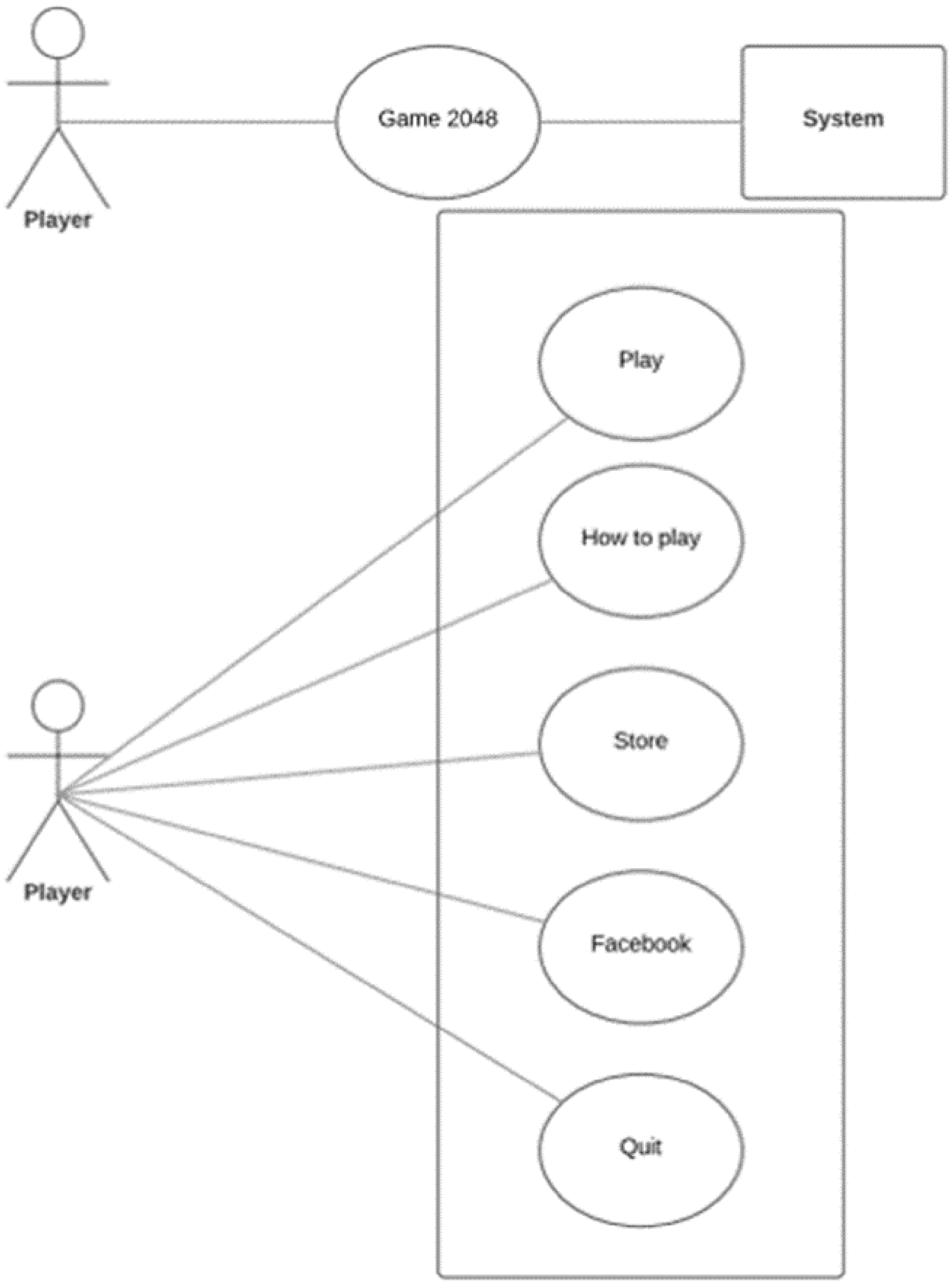
2. Photoshop

2.4 Use Case Scenario

We have created the use cases based on the UX view of the game.

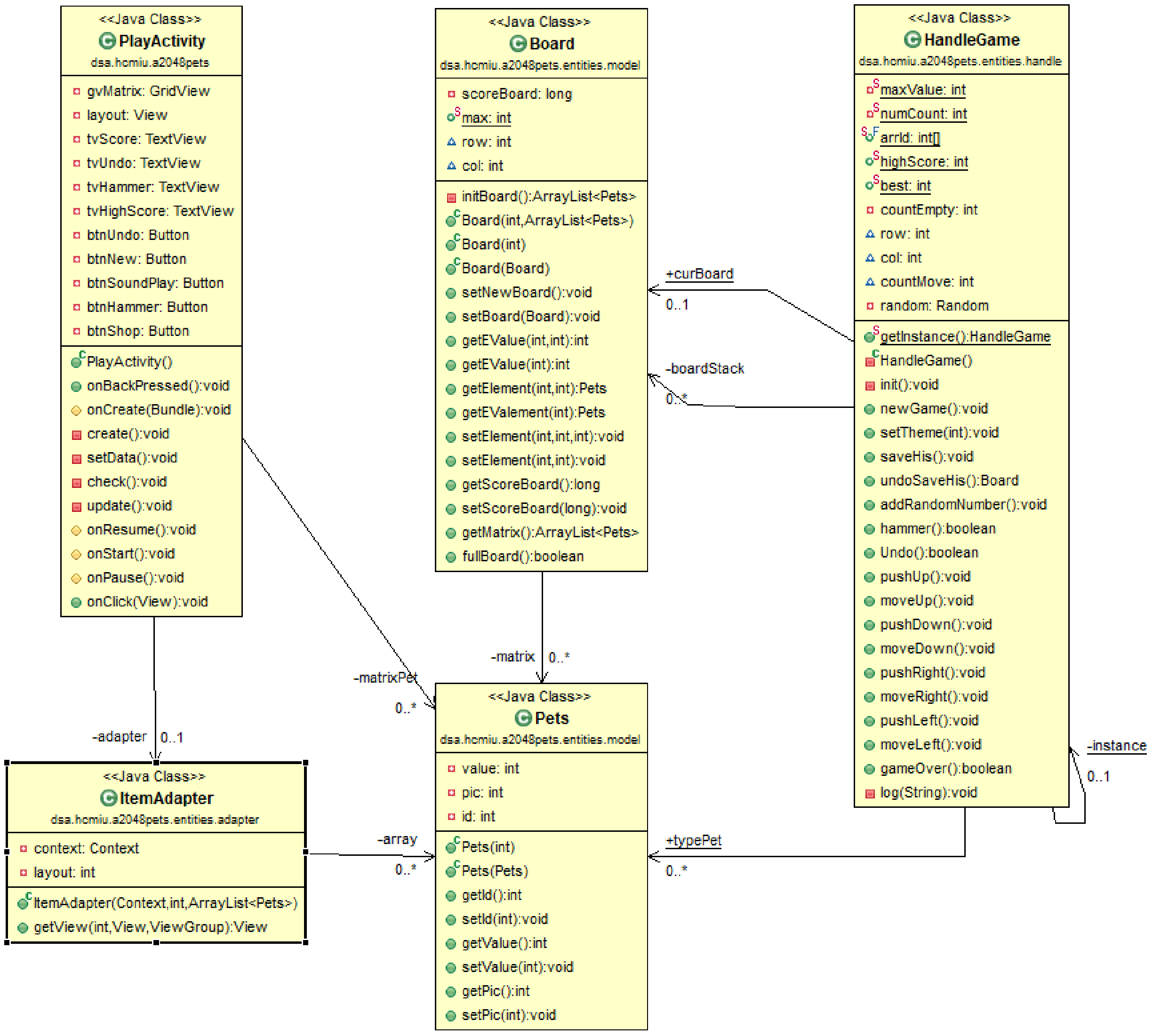
2048 Pets	PLAY	Play the game
		Resume the game
		Exit the game
	HOW TO PLAY	Show “How to play”
	STORE	Buy
		Equip
	FACEBOOK	Log in to save your score
	QUIT	Exit the game

2.5 Use Case diagram

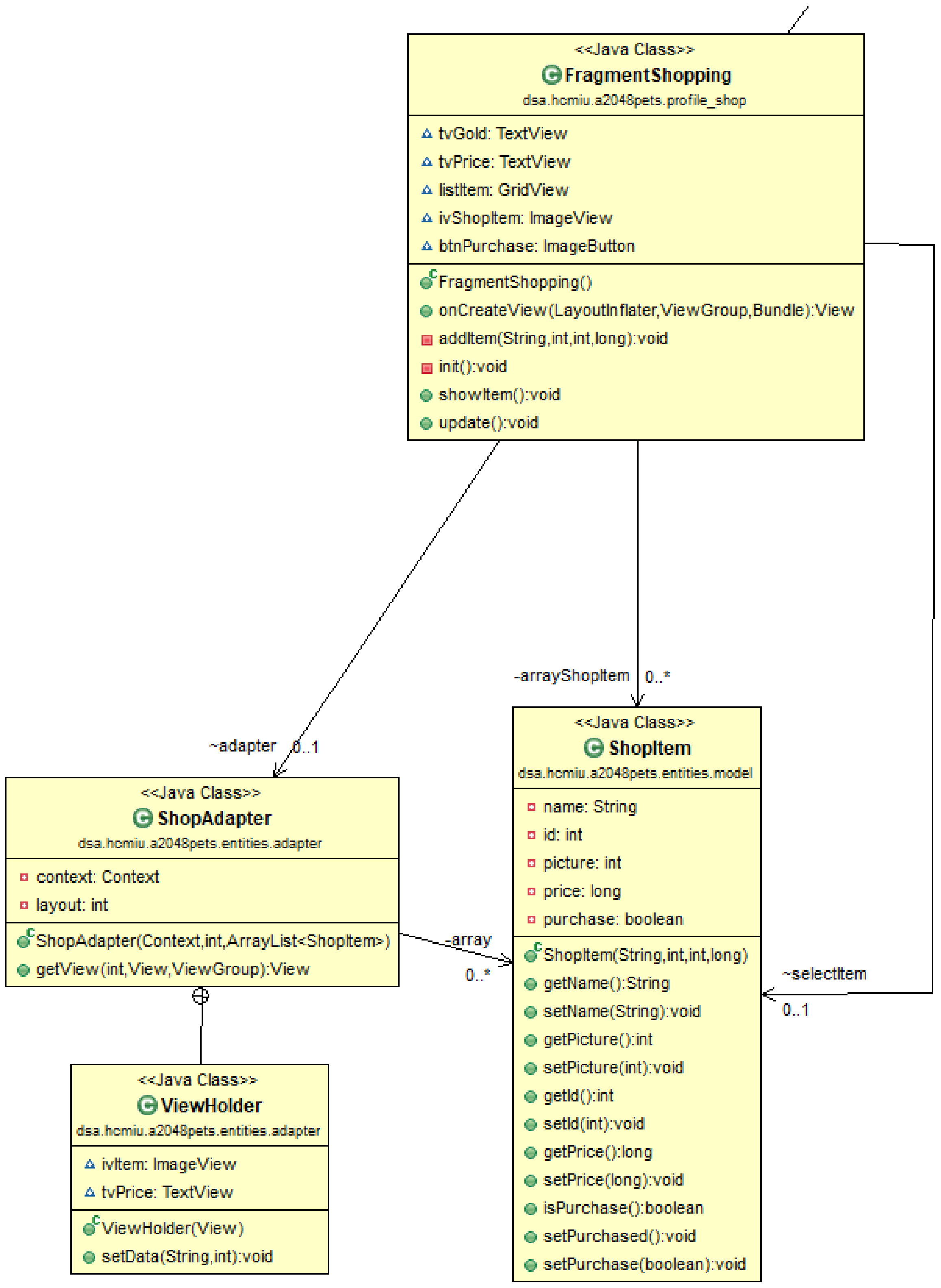


2.6 Class diagram

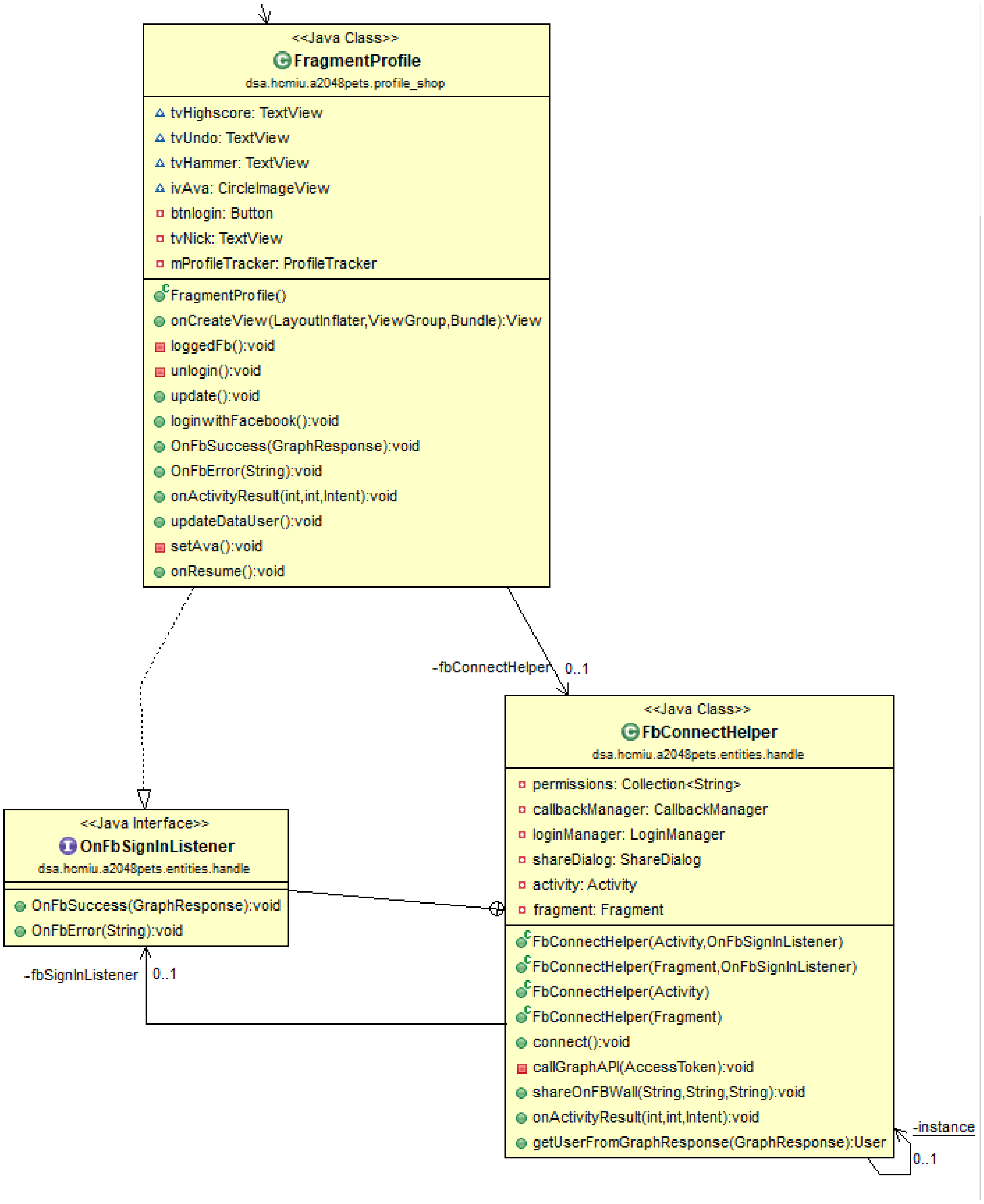
Play process diagram



Shop process diagram

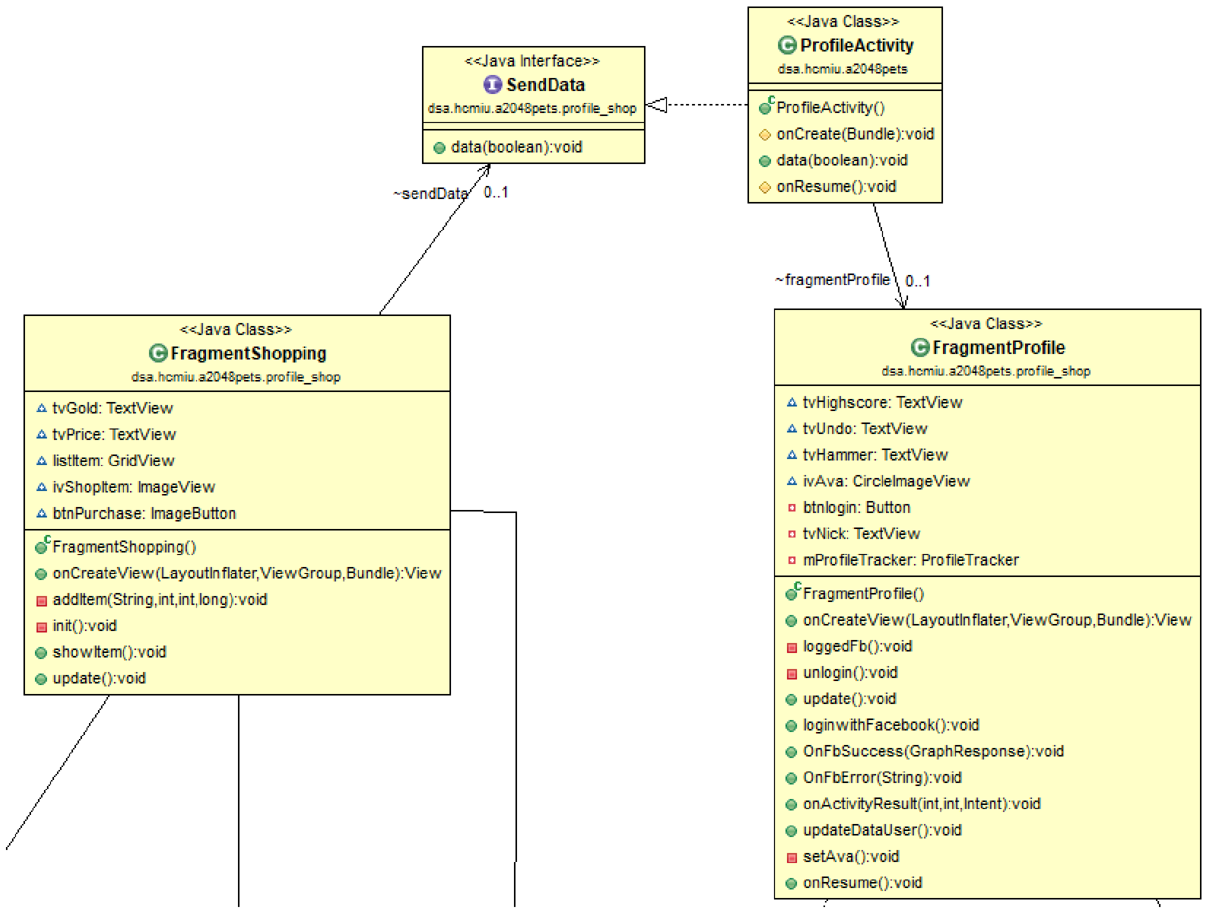


Profile diagram



<<Java Class>>
FragmentProfile
dsa.hcmiu.a2048pets.profile_shop
tvHighscore: TextView
tvUndo: TextView
tvHammer: TextView
ivAva: CircleImageView
btnlogin: Button
tvNick: TextView
mProfileTracker: ProfileTracker
FragmentProfile()
onCreateView(LayoutInflater,ViewGroup,Bundle):View
loggedFb():void
unlogin():void
update():void
loginwithFacebook():void
OnFbSuccess(GraphResponse):void
OnFbError(String):void
onActivityResult(int,int,Intent):void
updateDataUser():void
setAva():void
onResume():void
-fbConnectHelper
0..1
<<Java Class>>
FbConnectHelper
dsa.hcmiu.a2048pets.entities.handle
permissions: Collection<String>
callbackManager: CallbackManager
loginManager: LoginManager
shareDialog: ShareDialog
activity: Activity
fragment: Fragment
FbConnectHelper(Activity,OnFbSignInListener)
FbConnectHelper(Fragment,OnFbSignInListener)
FbConnectHelper(Activity)
FbConnectHelper(Fragment)
connect():void
callGraphAPI(AccessToken):void
shareOnFBWall(String,String,String):void
onActivityResult(int,int,Intent):void
getUserFromGraphResponse(GraphResponse):User
-instance
0..1
<<Java Interface>>
OnFbSignInListener
dsa.hcmiu.a2048pets.entities.handle
OnFbSuccess(GraphResponse):void
OnFbError(String):void
-fbSignInListener
0..1

User Profile Activity



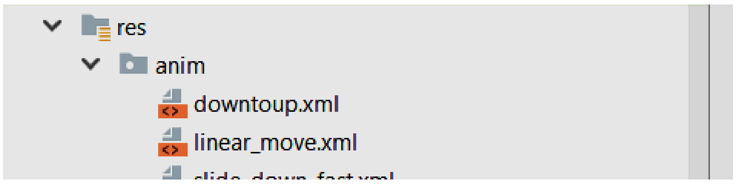
CHAP 3: DESIGN & IMPLEMENTATION

Package Diagram

UI Design

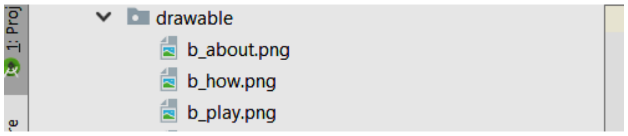
Resources files will be stored in res folder. We can put them in different directory with diffrent categories:

- **anim:** xml files of animation: slide down-up, zoom in-out,...



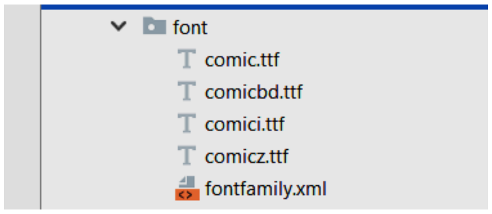
```
res
  anim
    downtoup.xml
    linear_move.xml
    slide_down_fast.xml
```

- **drawable:** background, button images,.... – most of the images for UI



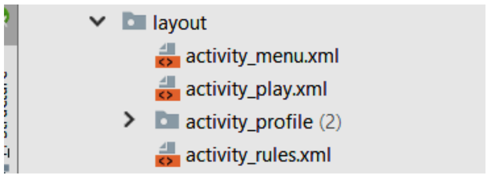
```
drawable
  b_about.png
  b_how.png
  b_play.png
```

- **font:** Some fonts and file xml *fontfamily* which declairs our addition fonts for later use.



```
font
  comic.ttf
  comicbd.ttf
  comici.ttf
  comicz.ttf
  fontfamily.xml
```

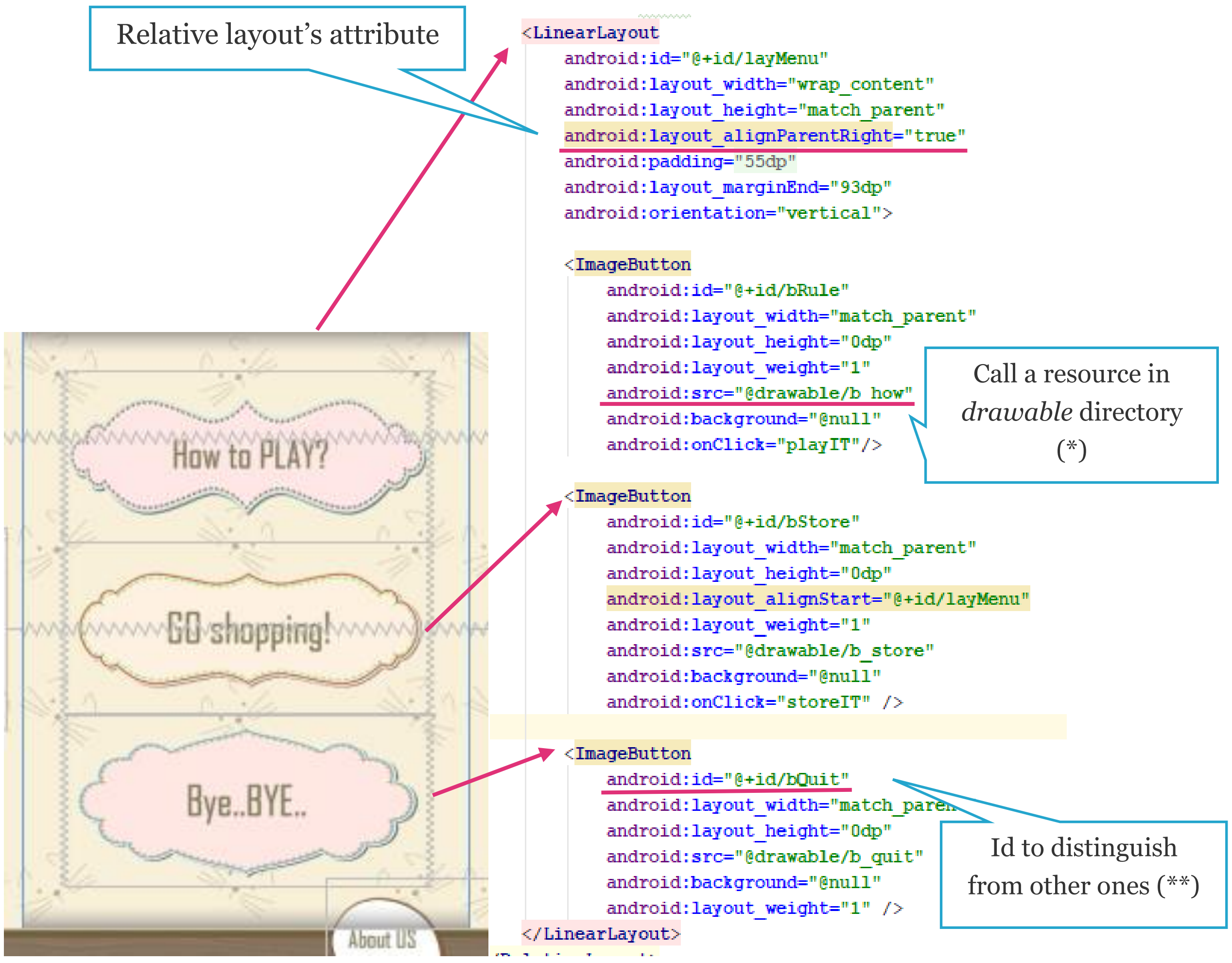
- **raw:** contains other raw files like sound...
- **layout:** A layout defines the structure for a user interface in your app, such as in an activity.



```
layout
  activity_menu.xml
  activity_play.xml
  activity_profile (2)
  activity_rules.xml
```

With UI design, xml file is main tool. So using Android's XML vocabulary, we can quickly design UI layouts and the screen elements they contain, in the same way creating web pages in HTML — with a series of nested elements.

Each layout file must contain exactly one root element, which must be a View or ViewGroup object. Once you've defined the root element, you can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout. For example, here's an XML layout that uses a vertical LinearLayout to hold a Button:


Relative layout's attribute
Call a resource in *drawable* directory (*)
Id to distinguish from other ones (**)
How to PLAY?
GO shopping!
Bye..BYE..
About US

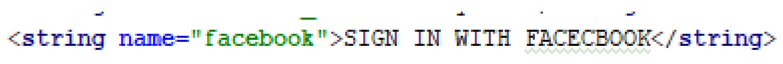
```
<LinearLayout
    android:id="@+id/layMenu"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_alignParentRight="true"
    android:padding="55dp"
    android:layout_marginEnd="93dp"
    android:orientation="vertical">

    <ImageButton
        android:id="@+id/bRule"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:src="@drawable/b_how"
        android:background="@null"
        android:onClick="playIT"/>

    <ImageButton
        android:id="@+id/bStore"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_alignStart="@+id/layMenu"
        android:layout_weight="1"
        android:src="@drawable/b_store"
        android:background="@null"
        android:onClick="storeIT" />

    <ImageButton
        android:id="@+id/bQuit"
        android:layout_width="match_paren
        android:layout_height="0dp"
        android:src="@drawable/b_quit"
        android:background="@null"
        android:layout_weight="1" />
</LinearLayout>
```

(*): we can call other resources in xml language by this method. In java language, we use: `R.drawable.b_how`

(**): By declaring id for each item, we can easily call them in java class: `R.id.bQuit`

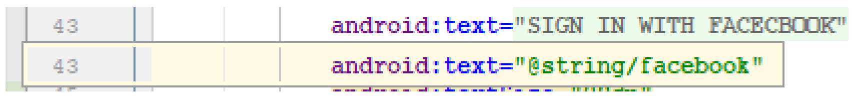
- **values:** Some default xml files for assigning color html code, dimension values, strings / string arrays, style layout into an id. For example with our strings.xml:

single string:



```
<string name="facebook">SIGN IN WITH FACECBOOK</string>
```

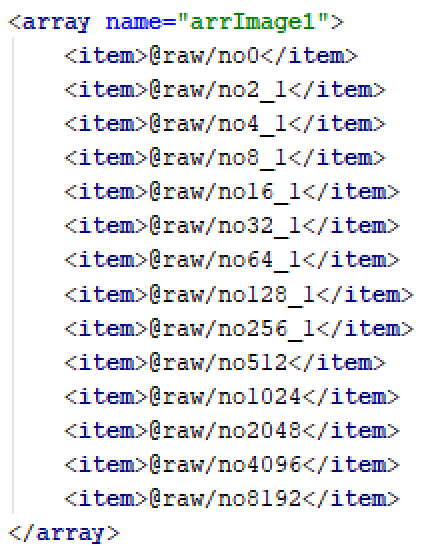
- Access item in xml language:



```
android:text="SIGN IN WITH FACECBOOK"
android:text="@string/facebook"
```

- Access item in java language: `R.string.facebook`

array string:



```
<array name="arrImage1">
    <item>@raw/no0</item>
    <item>@raw/no2_1</item>
    <item>@raw/no4_1</item>
    <item>@raw/no8_1</item>
    <item>@raw/no16_1</item>
    <item>@raw/no32_1</item>
    <item>@raw/no64_1</item>
    <item>@raw/no128_1</item>
    <item>@raw/no256_1</item>
    <item>@raw/no512</item>
    <item>@raw/no1024</item>
    <item>@raw/no2048</item>
    <item>@raw/no4096</item>
    <item>@raw/no8192</item>
</array>
```

- Access item in java language:

```
int[] no;

int resid= R.array.arrImage1;

TypedArray images = context.obtainTypedArray(resid);

for (int i = 1; i < 14; i++) {

    no[i]=setPic(images.getResourceId(i, -1));

}
```

If there isn't any picture at index i, method will return default value (-1)

Basic Android

1. Application

Exit App:

```
System.exit(0);
```

Open an url:

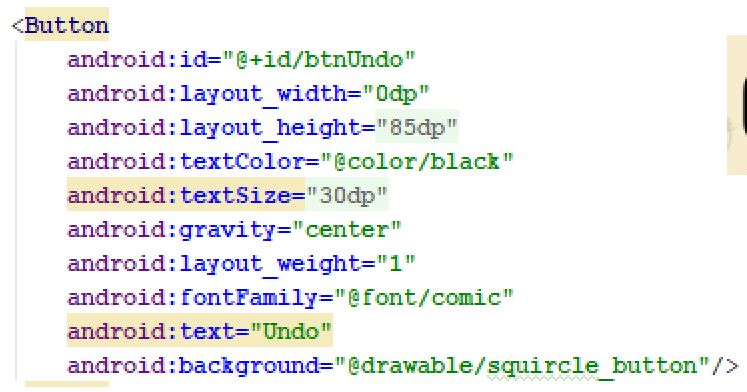
```
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(String url)));
```

2. Palette attributes:

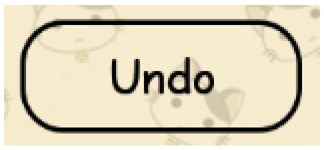
Android Studio gives us many kinds of palletes to use such as button, image button, text view,... similar with netbean, eclipse... but it supports more item properties, easier for developer to use

3. Map variables:

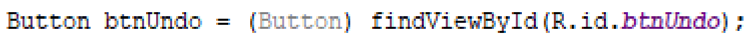
When we have a button on xml, and we want to manage it on java class, we need to map them. For example, our project has a button like this:



```xml
<Button
    android:id="@+id/btnUndo"
    android:layout_width="0dp"
    android:layout_height="85dp"
    android:textColor="@color/black"
    android:textSize="30dp"
    android:gravity="center"
    android:layout_weight="1"
    android:fontFamily="@font/comic"
    android:text="Undo"
    android:background="@drawable/squircle_button"/>
```

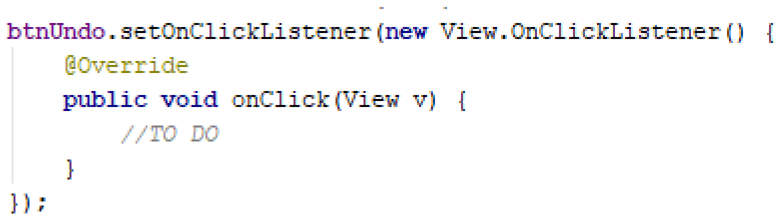


To map this one with a variable in Java class we just need to cast id:



```java
Button btnUndo = (Button) findViewById(R.id.btnUndo);
```

Then we can manage the button above such as: On Clicked:



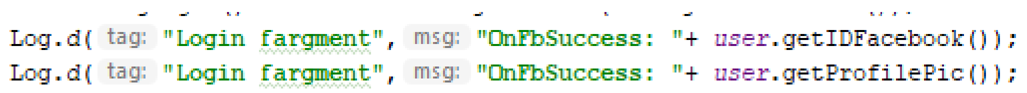
```java
btnUndo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //TO DO
    }
});
```

4. Debug

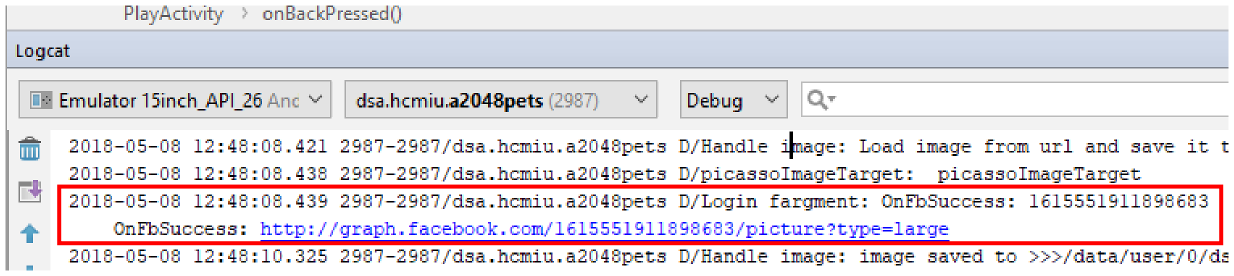
This method help us check and debug the code easier. These debug messages can be seen in logcat screen when the app is running by emulator.

```
log.d(String tag, String message);
```

For example:



```
Log.d( tag: "Login fargment", msg: "OnFbSuccess: "+ user.getIDFacebook());
Log.d( tag: "Login fargment", msg: "OnFbSuccess: "+ user.getProfilePic());
```



PlayActivity › onBackPressed()
Logcat
Emulator 15inch_API_26 Anc
dsa.hcmiu.a2048pets (2987)
Debug

```
2018-05-08 12:48:08.421 2987-2987/dsa.hcmiu.a2048pets D/Handle image: Load image from url and save it t
2018-05-08 12:48:08.438 2987-2987/dsa.hcmiu.a2048pets D/picassoImageTarget:  picassoImageTarget
2018-05-08 12:48:08.439 2987-2987/dsa.hcmiu.a2048pets D/Login fargment: OnFbSuccess: 1615551911898683
    OnFbSuccess: http://graph.facebook.com/1615551911898683/picture?type=large
2018-05-08 12:48:10.325 2987-2987/dsa.hcmiu.a2048pets D/Handle image: image saved to >>>/data/user/0/ds
```

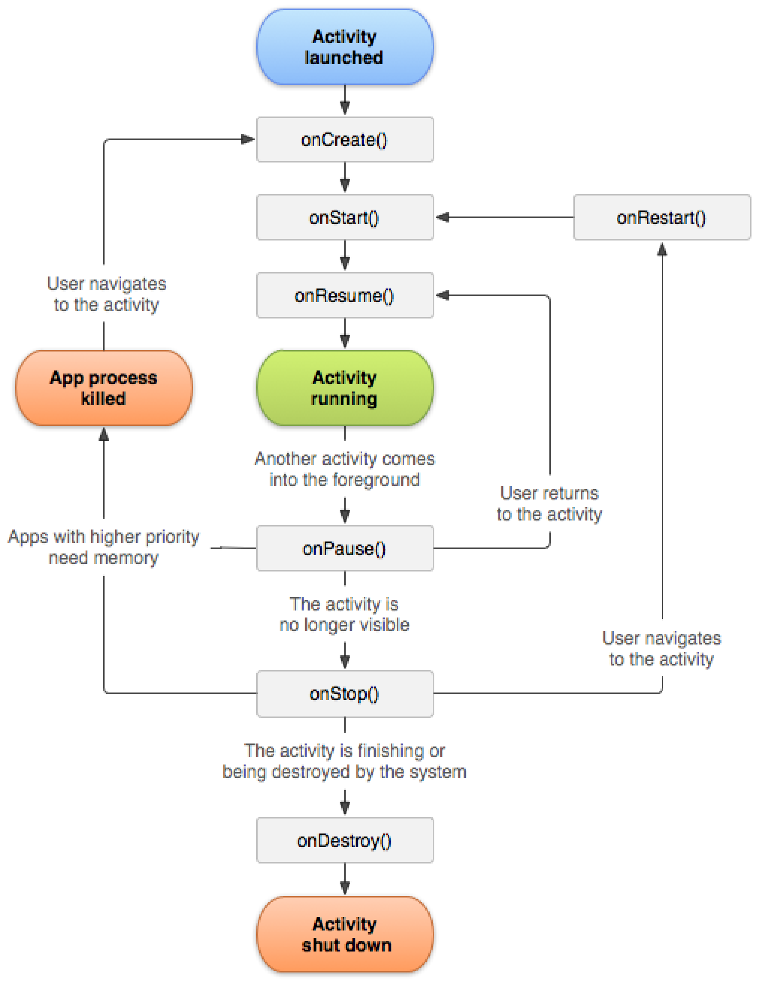
5. Activity

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

To declare your activity, open your manifest file and add an <activity> element as a child of the <application> element. For example:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

State diagram for an Android Activity Lifecycle.

There are three key loops you may be interested in monitoring within your activity:

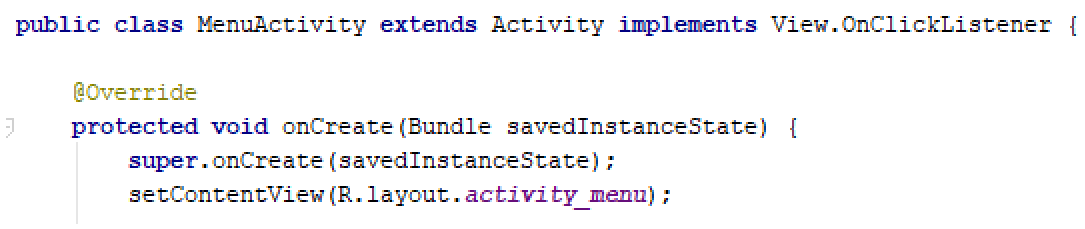
- The entire lifetime of an activity happens between the first call to onCreate(Bundle) through to a single final call to onDestroy(). An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy().

- The visible lifetime of an activity happens between a call to onStart() until a corresponding call to onStop(). During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user. Between these two methods you can maintain resources that are needed to show the activity to the user. The foreground lifetime of an activity happens between a call to onResume() until a corresponding call to onPause(). During this time the activity is in front of all other activities and interacting with the user. An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```

❖ To connect layout xml file with activity class we call method:

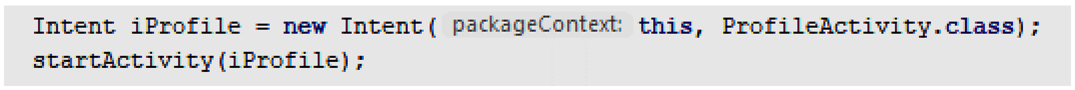
```
setContentView(int layout)
```



```
public class MenuActivity extends Activity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
```

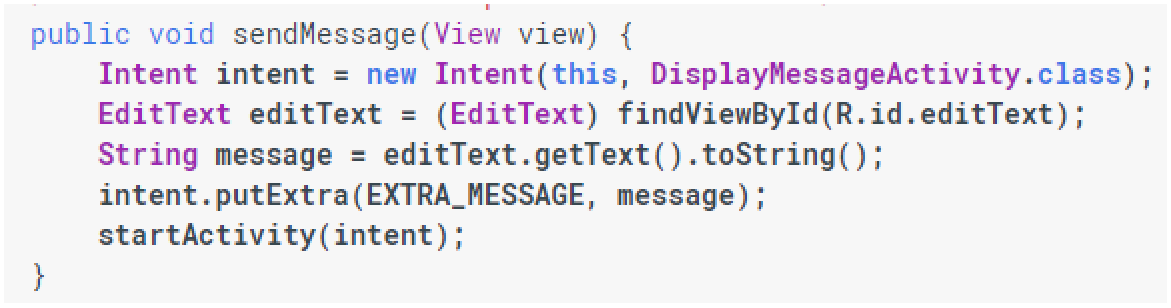
❖ Start another activity:

An Intent is an object that provides runtime binding between separate components, such as two activities. The Intent represents an app's "intent to do something." You can use intents for a wide variety of tasks, but in this lesson, your intent starts another activity.



```
Intent iProfile = new Intent( packageContext: this, ProfileActivity.class);
startActivity(iProfile);
```

❖ Transfer data between activities: add the EXTRA_MESSAGE constant and the sendMessage() code, as shown here:



```
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.editText);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

In *DisplayMessageActivity*, add the following code to the onCreate() method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);

    // Get the Intent that started this activity and extract the string
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    // Capture the layout's TextView and set the string as its text
    TextView textView = findViewById(R.id.textView);
    textView.setText(message);
}
```

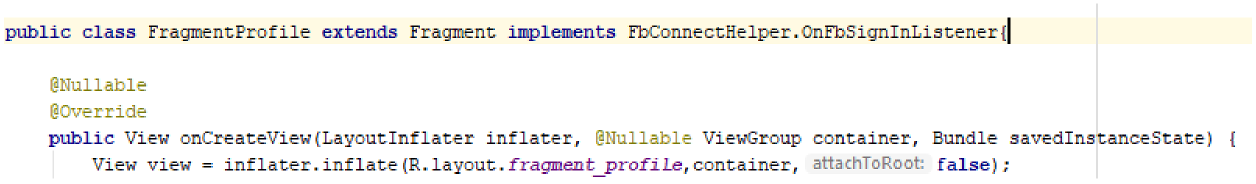
❖ Handle other button of your mobile/tablet

```
@Override
public void onBackPress() {
        //TO DO
}
```

6. Fragment

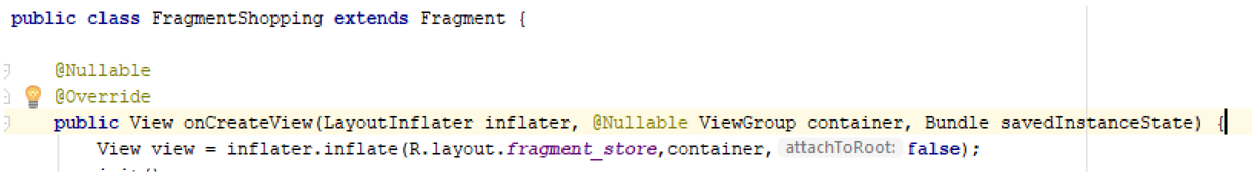
The FragmentActivity subclass can make use of the Fragment class to better modularize their code, build more sophisticated user interfaces for larger screens, and help scale their application between small and large screens.

- *FragmentProfile* connects with layout *fragment_profile*



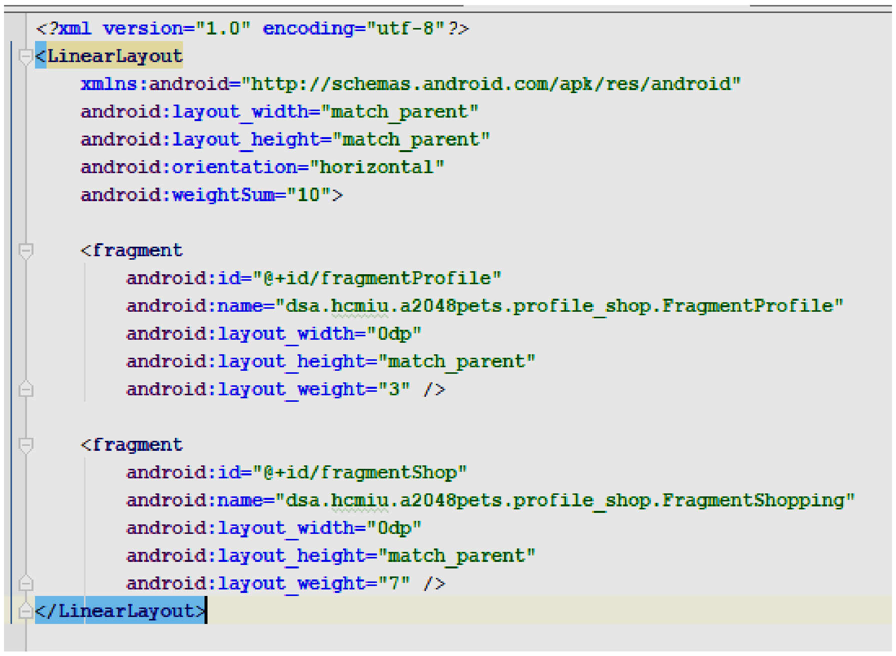
```
public class FragmentProfile extends Fragment implements FbConnectHelper.OnFbSignInListener{

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_profile,container, attachToRoot: false);
```

- *FragmentShopping* connects with layout *fragment_store*



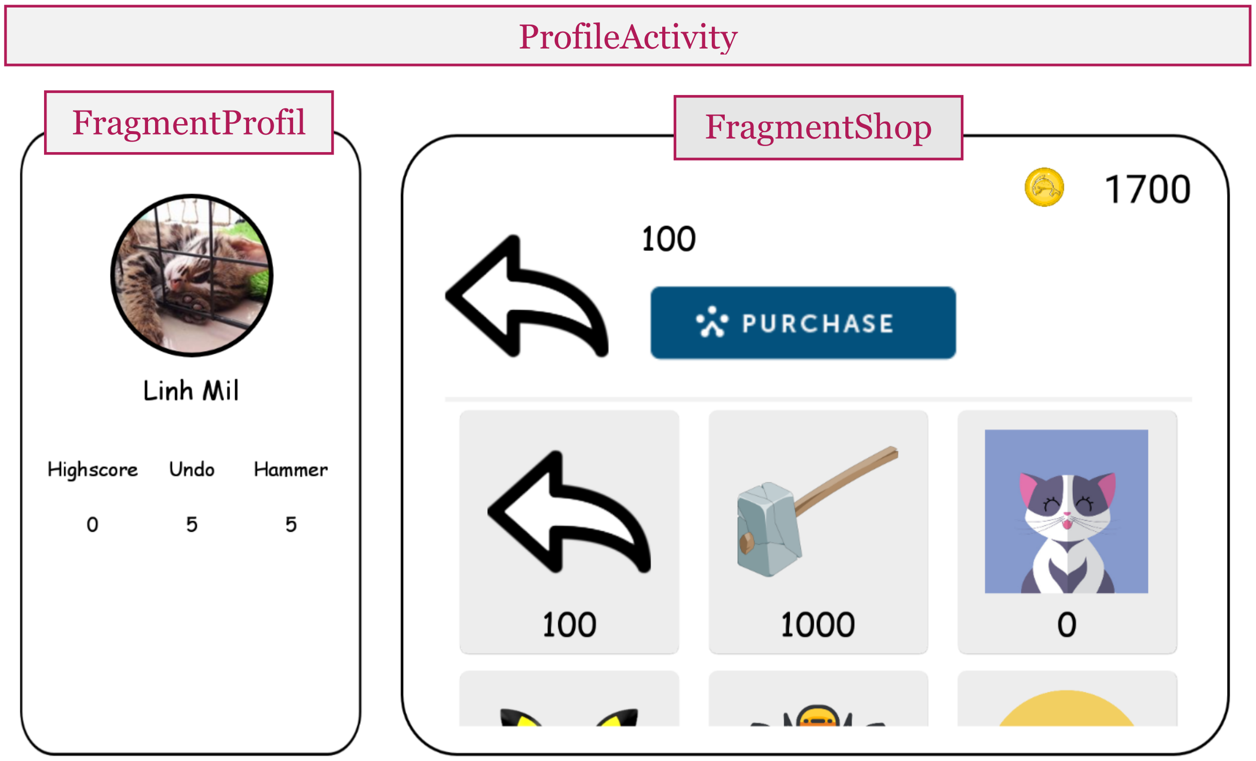
```
public class FragmentShopping extends Fragment {

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_store,container, attachToRoot: false);
```

- Then add those 2 fragments to file xml which is connected with main activity:



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="10">

    <fragment
        android:id="@+id/fragmentProfile"
        android:name="dsa.hcmiu.a2048pets.profile_shop.FragmentProfile"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

    <fragment
        android:id="@+id/fragmentShop"
        android:name="dsa.hcmiu.a2048pets.profile_shop.FragmentShopping"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="7" />
</LinearLayout>
```

So whenever that activity starts, 2 fragment will start together:

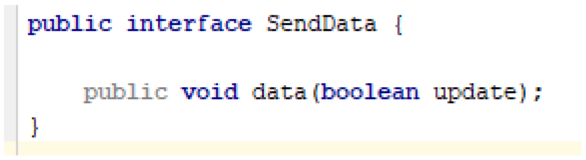


- Send data between 2 fragments:

In order to receive event callbacks from the fragment, the activity that hosts it must implement the interface defined in the fragment class.

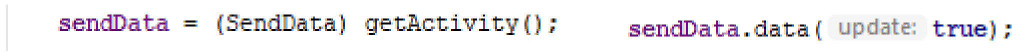
```
public static class MainActivity extends Activity
        implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // Do something here to display that article
    }
}
```

For example, we creat interface sendData:



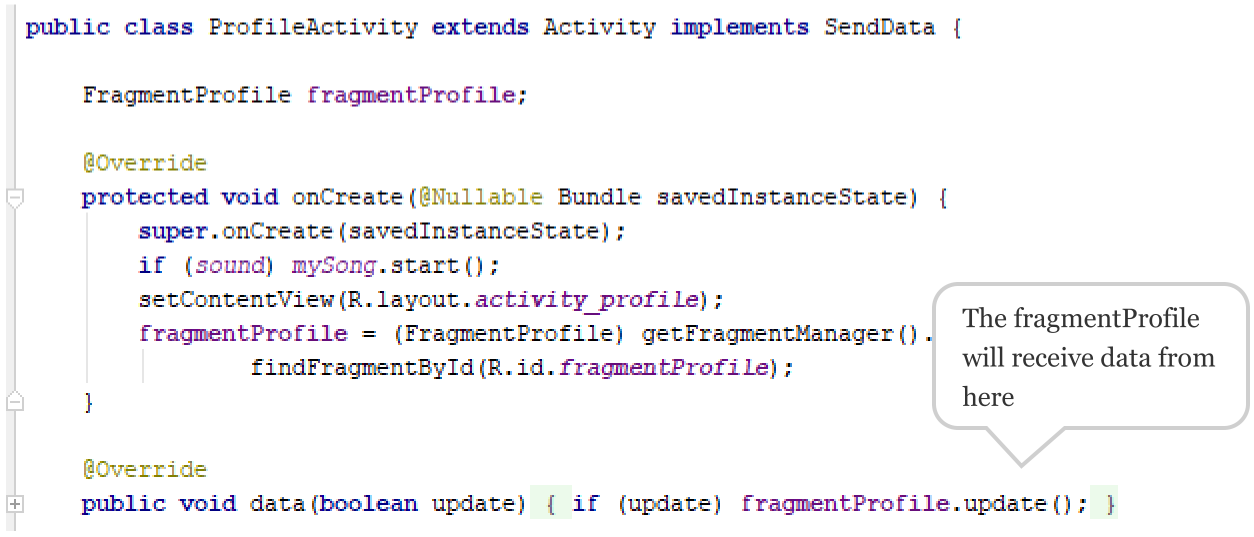
```
public interface SendData {

    public void data(boolean update);
}
```

Then in FragmentShopping, we map sendData object, and give it the data:



```
    sendData = (SendData) getActivity();       sendData.data( update: true);
```

In parent Activity, we implement sendData interface:



```
public class ProfileActivity extends Activity implements SendData {

    FragmentProfile fragmentProfile;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (sound) mySong.start();
        setContentView(R.layout.activity_profile);
        fragmentProfile = (FragmentProfile) getFragmentManager().
                findFragmentById(R.id.fragmentProfile);
    }

    @Override
    public void data(boolean update) { if (update) fragmentProfile.update(); }
```

The fragmentProfile will receive data from here

We can also apply this method on transferring data between 2 activity or dialog and activity.

7. Sound & Animation Effect

- **Animation**

In order to perform animation in android , we are going to call a static function loadAnimation() of the class AnimationUtils. We are going to receive the result in an instance of Animation Object. Its syntax is as follows:

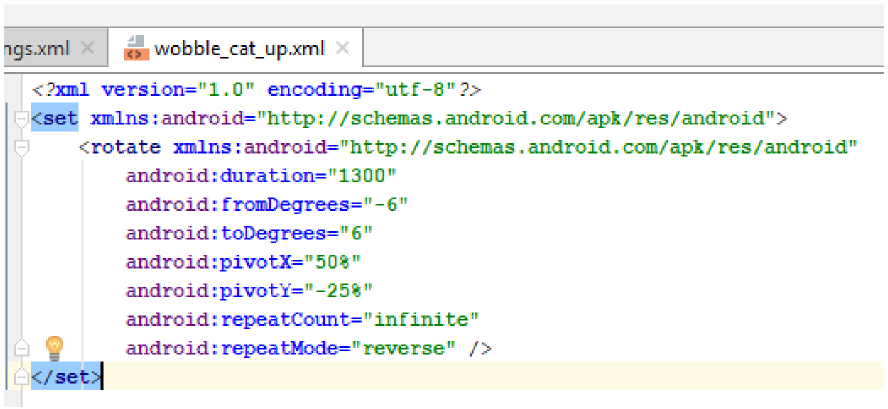
```
Animation animation = AnimationUtils.loadAnimation(getApplicationContext(),

   int idAnimationResources);
```

Then assign that animation to the attribute that we want:

```
       source.setAnimation(animation);
```

For example:

wobble_cat_up.xml is stored in anim directory



```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1300"
        android:fromDegrees="-6"
        android:toDegrees="6"
        android:pivotX="50%"
        android:pivotY="-25%"
        android:repeatCount="infinite"
        android:repeatMode="reverse" />
</set>
```

Set animation to item

```
Animation wobbleCup = AnimationUtils.loadAnimation( context: this, R.anim.wobble_cup_cats);
Animation wobbleCatUp = AnimationUtils.loadAnimation( context: this, R.anim.wobble_cat_up);
Animation linear = AnimationUtils.loadAnimation( context: this, R.anim.linear_move);
ivCupCat.setAnimation(wobbleCup);
ivShadow.setAnimation(linear);
btnSound.setAnimation(wobbleCatUp);
```

- **Sound**

Similar with animation effect, in this case, we have *MediaPlayer* (This class is the primary API for playing sound and video)

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
          //sound_file_1 is stored in raw diretory
mediaPlayer.start();
mediaPlayer.setLooping(true); // loop the music infinitly
```

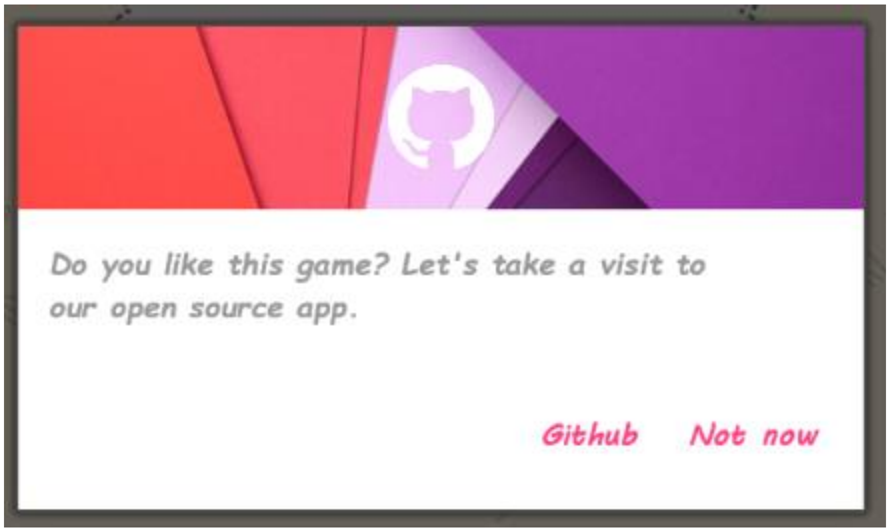
8. Dialog

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

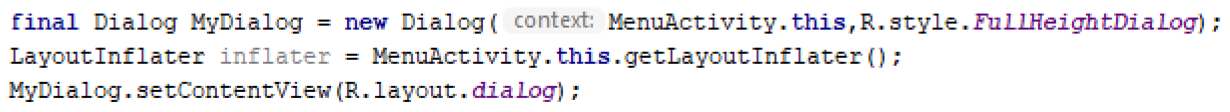
The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:

- **AlertDialog**: A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.
- **DatePickerDialog** or **TimePickerDialog**: A dialog with a pre-defined UI that allows the user to select a date or time.

But in this project we only use, our custom Dialog. This is one of them:

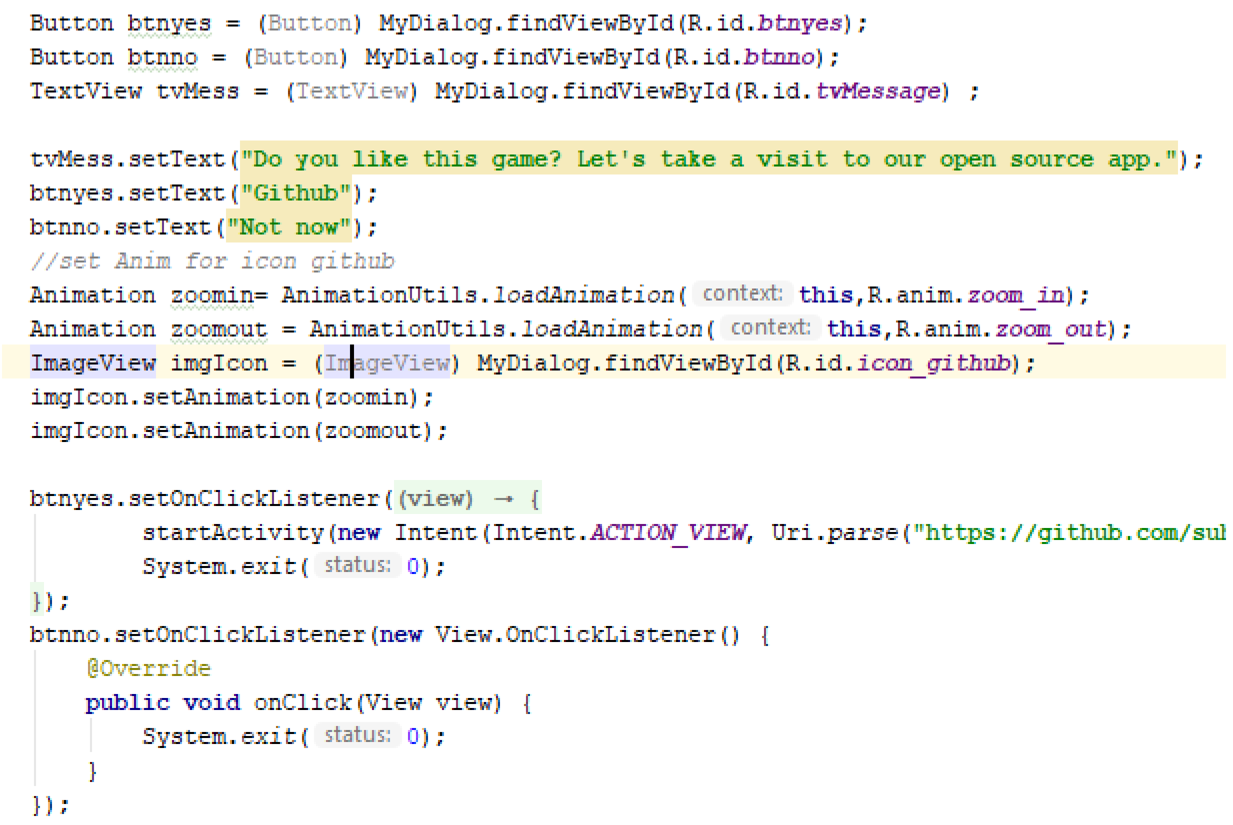

Do you like this game? Let's take a visit to our open source app.

The layout xml keeps being placed in layout directory and has 2 button, 1 TextView, 1 image effect. To make this dialog, we absolutely use *LayoutInflater* to set layout for it.



```
final Dialog MyDialog = new Dialog( context: MenuActivity.this,R.style.FullHeightDialog);
LayoutInflater inflater = MenuActivity.this.getLayoutInflater();
MyDialog.setContentView(R.layout.dialog);
```

Then do basic steps like mapping, assigning,….:



```
Button btnyes = (Button) MyDialog.findViewById(R.id.btnyes);
Button btnno = (Button) MyDialog.findViewById(R.id.btnno);
TextView tvMess = (TextView) MyDialog.findViewById(R.id.tvMessage) ;

tvMess.setText("Do you like this game? Let's take a visit to our open source app.");
btnyes.setText("Github");
btnno.setText("Not now");
//set Anim for icon github
Animation zoomin= AnimationUtils.loadAnimation( context: this,R.anim.zoom_in);
Animation zoomout = AnimationUtils.loadAnimation( context: this,R.anim.zoom_out);
ImageView imgIcon = (ImageView) MyDialog.findViewById(R.id.icon_github);
imgIcon.setAnimation(zoomin);
imgIcon.setAnimation(zoomout);

btnyes.setOnClickListener((view) → {
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://github.com/sul
        System.exit( status: 0);
});
btnno.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        System.exit( status: 0);
    }
});
```

Finally, to make this dialog appear on screen we use method: `MyDialog.show();`

Dialog gives us some more properties:

- Cancle dialog: `Cancle()`
- Unable touch outside the dialog: `setCanceledOnTouchOutside(false)`

Main methods

1. Handle file

The HandleFile is a singleton class. It helps our app remain the data for later running, against missing user data. We use JSON file to store those data from Feature class (which contains the static variables).

JSON stands for JavaScript Object Notation.It is an independent data exchange format and is the best alternative for XML. This chapter explains how to parse the JSON file and extract necessary information from it.

Android provides four different classes to manipulate JSON data. These classes are JSONArray, JSONObject, JSONStringer and JSONTokenizer.

An JSON file consist of many components. Here is the table defining the components of an JSON file and their description –

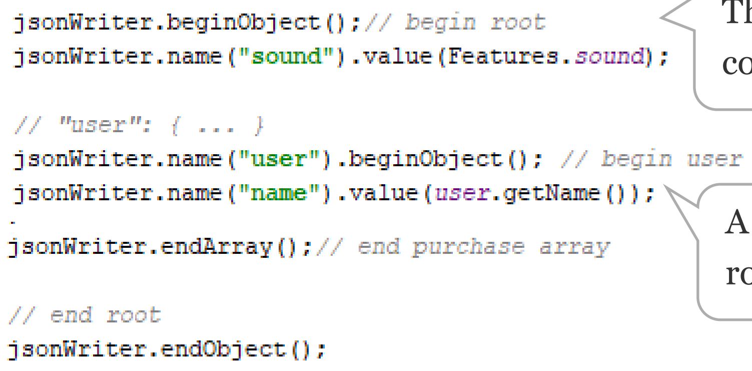
Sr.No	Component & description
1	**Array([)** In a JSON file , square bracket ([) represents a JSON array
2	**Objects({)** In a JSON file, curly bracket ({) represents a JSON object
3	**Key** A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object
4	**Value** Each key has a value that could be string , integer or double e.t.c

The method getJSONObject returns the JSON object. The method getString returns the string value of the specified key.

get(String name)	This method just Returns the value but in the form of Object type
getBoolean(String name)	This method returns the boolean value specified by the key
getDouble(String name)	This method returns the double value specified by the key
getInt(String name)	This method returns the integer value specified by the key

For example:

- Create JSON:

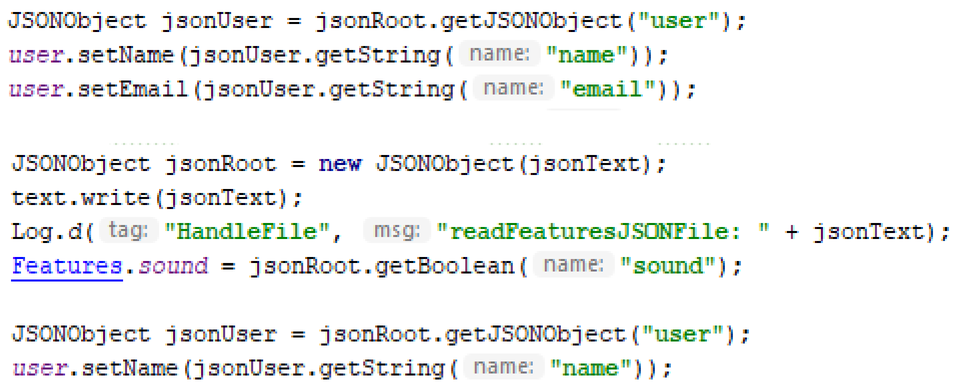


```
jsonWriter.beginObject();// begin root
jsonWriter.name("sound").value(Features.sound);

// "user": { ... }
jsonWriter.name("user").beginObject(); // begin user
jsonWriter.name("name").value(user.getName());

jsonWriter.endArray();// end purchase array

// end root
jsonWriter.endObject();
```

The root object which contains those variables

A child node of the above root

- Read JSON:



```
JSONObject jsonUser = jsonRoot.getJSONObject("user");
user.setName(jsonUser.getString( name: "name"));
user.setEmail(jsonUser.getString( name: "email"));

JSONObject jsonRoot = new JSONObject(jsonText);
text.write(jsonText);
Log.d( tag: "HandleFile",  msg: "readFeaturesJSONFile: " + jsonText);
Features.sound = jsonRoot.getBoolean( name: "sound");

JSONObject jsonUser = jsonRoot.getJSONObject("user");
user.setName(jsonUser.getString( name: "name"));
```

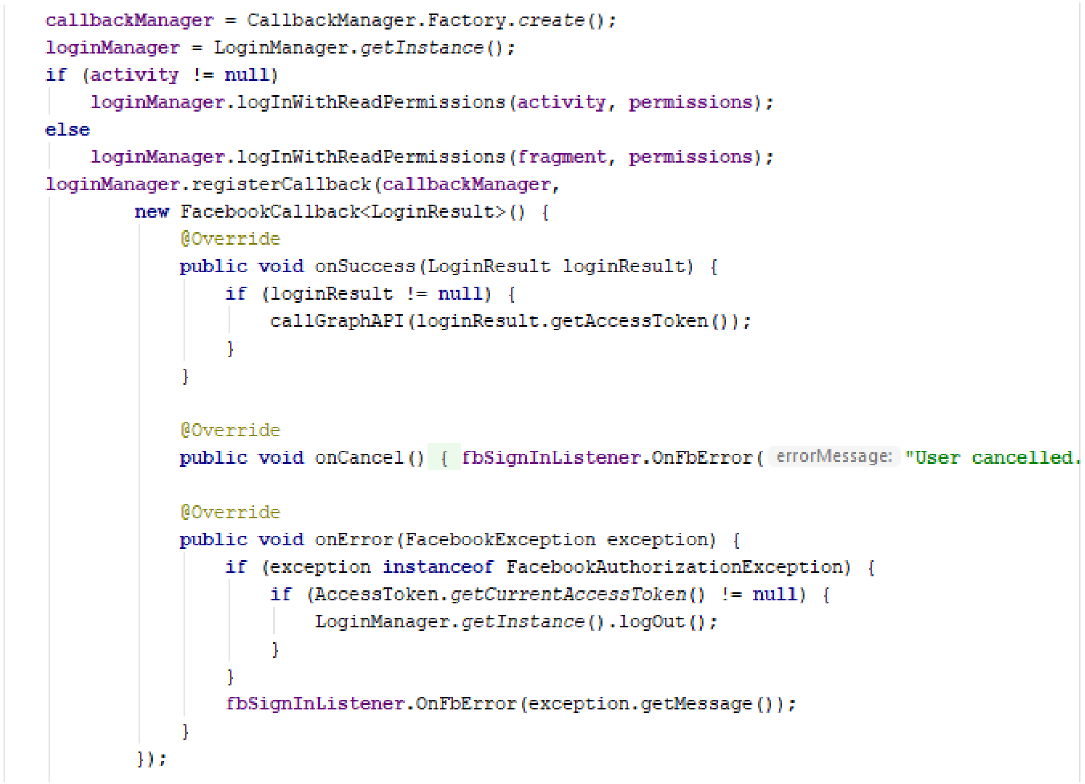
2. Facebook Connect:

FbConnectHelper class helps this app connect with facebook account, follow the instruction of Facebook developers (https://developers.facebook.com/docs/facebook-login/android)

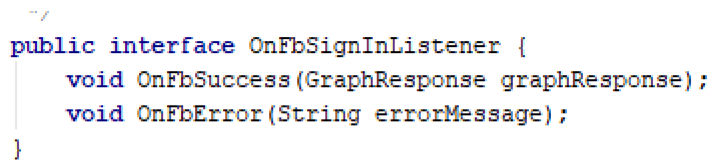
- Firstly, we give this app access Internet permission: in /app/manifest/AndroidManifest.xml file

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- a callbackManager to handle login responses by calling `CallbackManager.Factory.create.`
- To respond to a login result, you need to register a callback with `LoginManager`

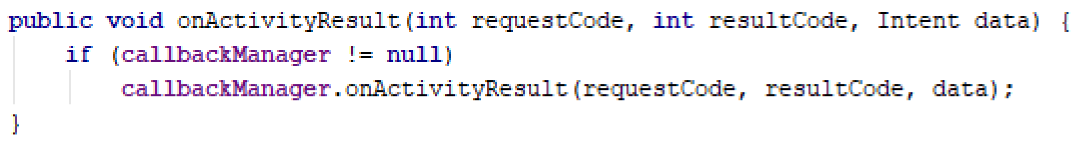


```
callbackManager = CallbackManager.Factory.create();
loginManager = LoginManager.getInstance();
if (activity != null)
    loginManager.logInWithReadPermissions(activity, permissions);
else
    loginManager.logInWithReadPermissions(fragment, permissions);
loginManager.registerCallback(callbackManager,
        new FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                if (loginResult != null) {
                    callGraphAPI(loginResult.getAccessToken());
                }
            }

            @Override
            public void onCancel() { fbSignInListener.OnFbError( errorMessage: "User cancelled."); }

            @Override
            public void onError(FacebookException exception) {
                if (exception instanceof FacebookAuthorizationException) {
                    if (AccessToken.getCurrentAccessToken() != null) {
                        LoginManager.getInstance().logOut();
                    }
                }
                fbSignInListener.OnFbError(exception.getMessage());
            }
        });
```



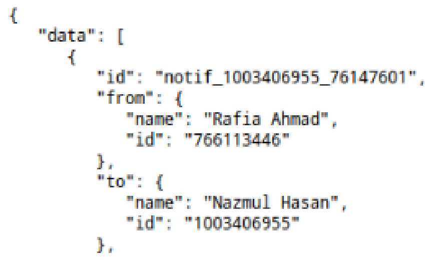
```
 */
public interface OnFbSignInListener {
    void OnFbSuccess(GraphResponse graphResponse);
    void OnFbError(String errorMessage);
}
```

- call `callbackManager.onActivityResult` to pass the login results to the `LoginManager` via `callbackManager`.



```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (callbackManager != null)
        callbackManager.onActivityResult(requestCode, resultCode, data);
}
```

- Then Facebook will response us by JSON file like this

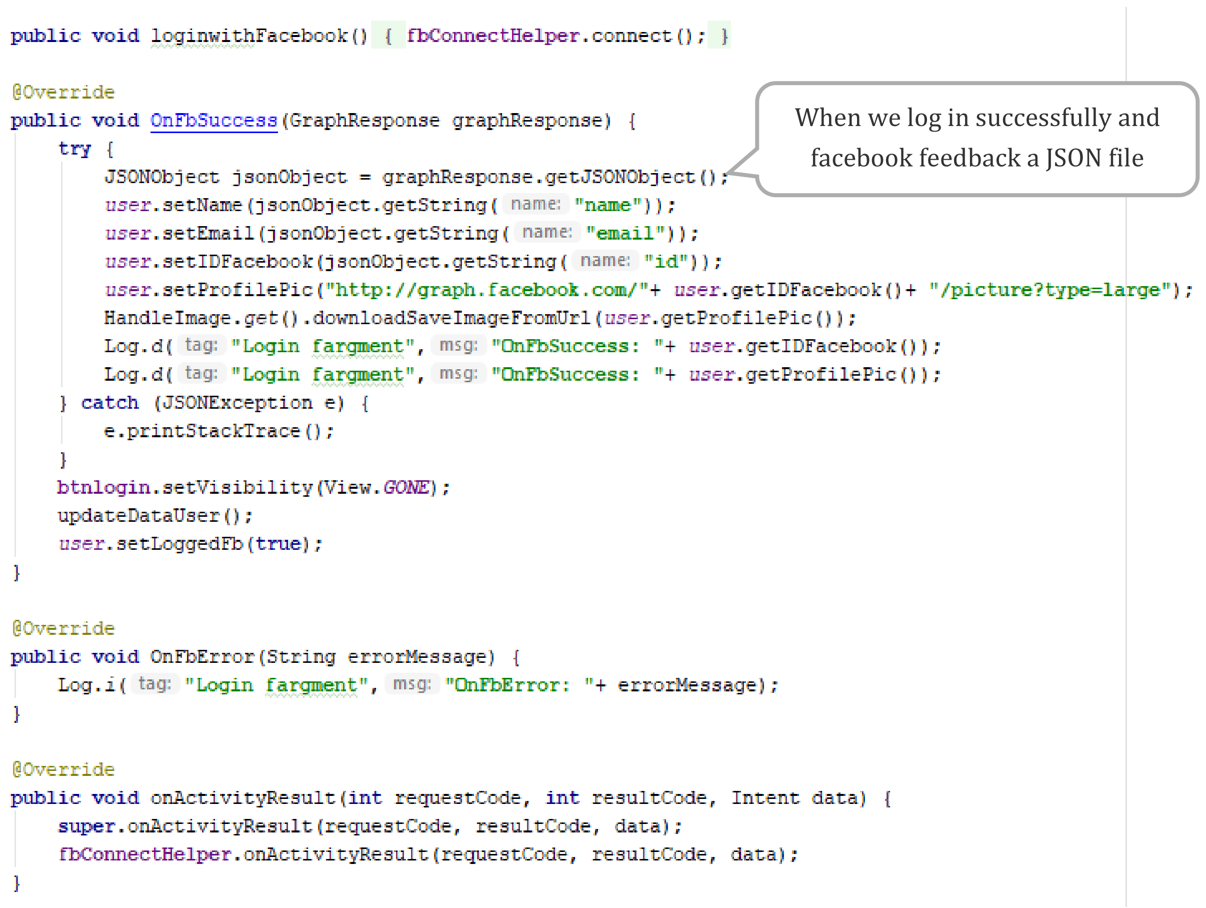


```
{
   "data": [
      {
         "id": "notif_1003406955_76147601",
         "from": {
            "name": "Rafia Ahmad",
            "id": "766113446"
         },
         "to": {
            "name": "Nazmul Hasan",
            "id": "1003406955"
         },
```

After that, we can analyze that file like Handle File

Use case of *FbConnectHelper*:

```
implements FbConnectHelper.OnFbSignInListener
```



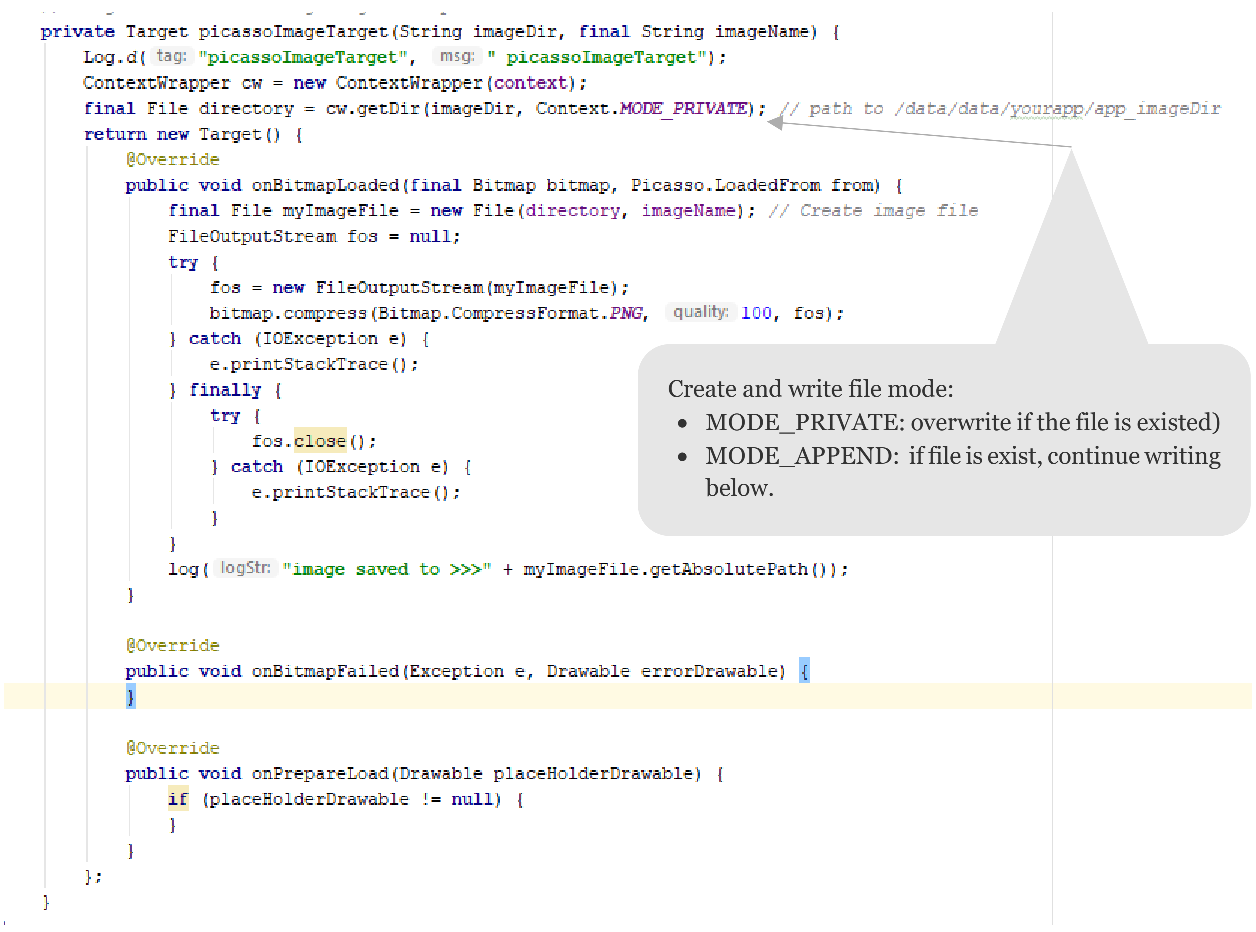
```
public void loginwithFacebook() { fbConnectHelper.connect(); }

@Override
public void OnFbSuccess(GraphResponse graphResponse) {
    try {
        JSONObject jsonObject = graphResponse.getJSONObject();
        user.setName(jsonObject.getString( name: "name"));
        user.setEmail(jsonObject.getString( name: "email"));
        user.setIDFacebook(jsonObject.getString( name: "id"));
        user.setProfilePic("http://graph.facebook.com/"+ user.getIDFacebook()+ "/picture?type=large");
        HandleImage.get().downloadSaveImageFromUrl(user.getProfilePic());
        Log.d( tag: "Login fargment", msg: "OnFbSuccess: "+ user.getIDFacebook());
        Log.d( tag: "Login fargment", msg: "OnFbSuccess: "+ user.getProfilePic());
    } catch (JSONException e) {
        e.printStackTrace();
    }
    btnlogin.setVisibility(View.GONE);
    updateDataUser();
    user.setLoggedFb(true);
}

@Override
public void OnFbError(String errorMessage) {
    Log.i( tag: "Login fargment", msg: "OnFbError: "+ errorMessage);
}

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

When we log in successfully and facebook feedback a JSON file

3. Handle image

We use Picasso library to support us handle the image which we will from the internet. We also apply singleton to this class.

- Target class for saving image bitmap returned from Picasso, so we just need to implement and use it to convert bitmap to jpg image.



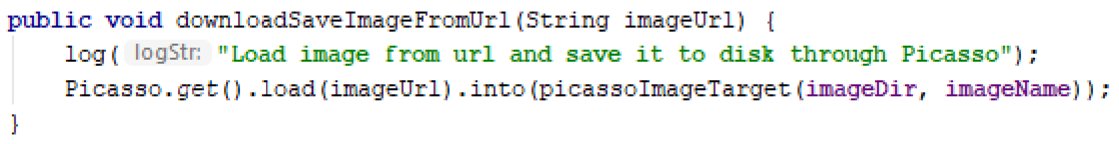
```
private Target picassoImageTarget(String imageDir, final String imageName) {
    Log.d( tag: "picassoImageTarget",  msg: " picassoImageTarget");
    ContextWrapper cw = new ContextWrapper(context);
    final File directory = cw.getDir(imageDir, Context.MODE_PRIVATE); // path to /data/data/yourapp/app_imageDir
    return new Target() {
        @Override
        public void onBitmapLoaded(final Bitmap bitmap, Picasso.LoadedFrom from) {
            final File myImageFile = new File(directory, imageName); // Create image file
            FileOutputStream fos = null;
            try {
                fos = new FileOutputStream(myImageFile);
                bitmap.compress(Bitmap.CompressFormat.PNG,  quality: 100, fos);
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            log( logStr: "image saved to >>>" + myImageFile.getAbsolutePath());
        }

        @Override
        public void onBitmapFailed(Exception e, Drawable errorDrawable) {
        }

        @Override
        public void onPrepareLoad(Drawable placeHolderDrawable) {
            if (placeHolderDrawable != null) {
            }
        }
    };
}
```

Create and write file mode:
- MODE_PRIVATE: overwrite if the file is existed)
- MODE_APPEND: if file is exist, continue writing below.

Then use load-image-file method of Picasso to assign to a directory:

```
Picasso.get().load(File file).into(Target target);

Picasso.get().load(File file).into(ImageView imageview);
```

This method also support load image from Internet url into target, for example:



```
public void downloadSaveImageFromUrl(String imageUrl) {
    log( logStr: "Load image from url and save it to disk through Picasso");
    Picasso.get().load(imageUrl).into(picassoImageTarget(imageDir, imageName));
}
```

4. Handle sound

It is a feature that when you click on the screen the sound will be on and become softer smoother in a moment of time

Starting up with 2 fade in, out and time duration

```
numberOfSteps = FADE_DURATION/FADE_INTERVAL

deltaVolume = MAX_VOLUME / (float)numberOfStep
```

(calculate number of fade step and volume change)

```
final Timer timer = new Timer(true);
```

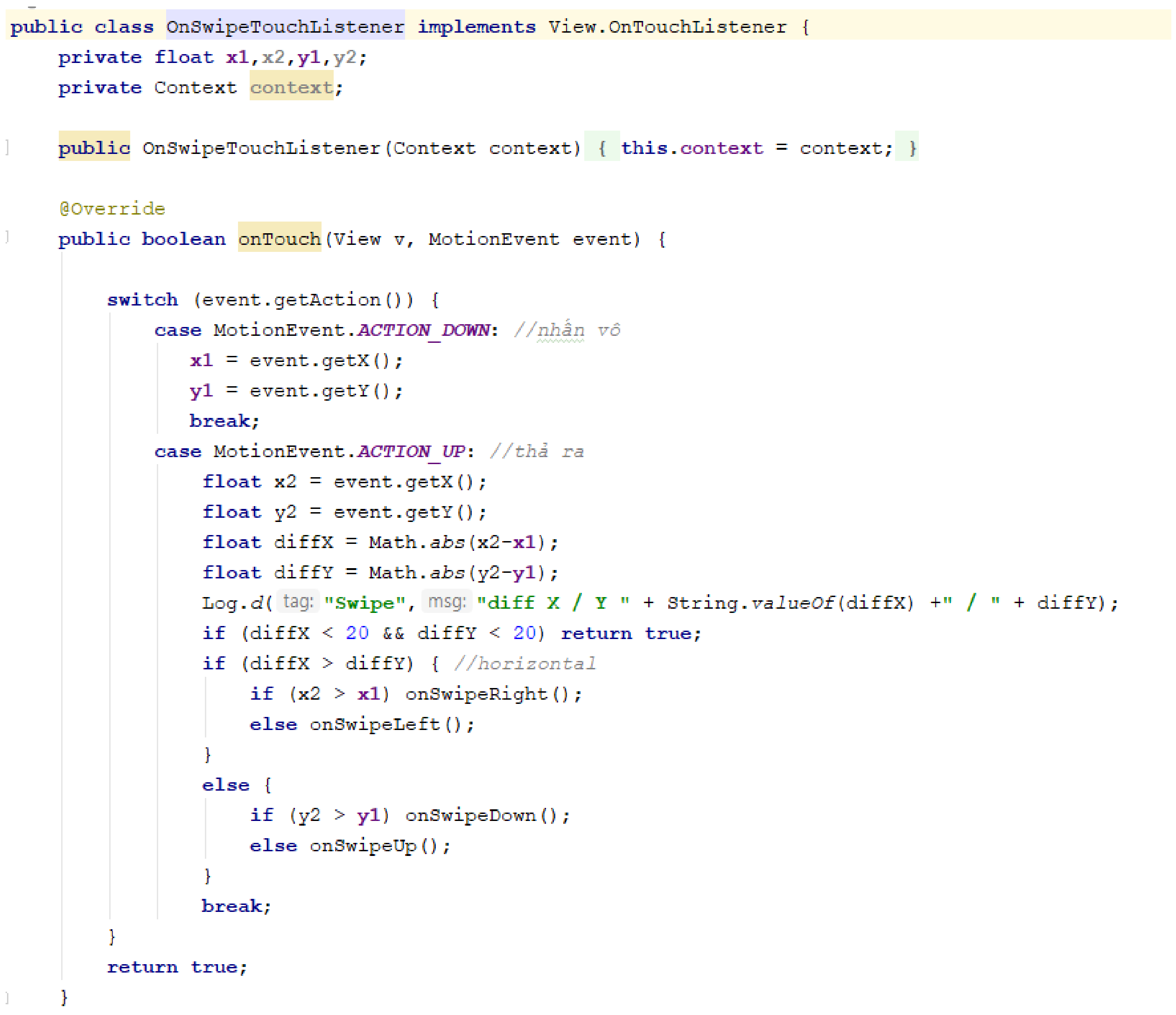
(create a new Timer task to run on UI thread)

```
public void run() {

      fadeInStep(deltaVolume);      //fade step

      if(volume_fadein>=1f){

                  timer.cancel();

                  timer.purge();

      }

}
```

(Cancel and purge the Timer)

5. OnSwipeTouchListener (implements OntouchListener)

Touch listener create a connection between the mouse and the application. By "listening" to the signal DOWN when you press the mouse button then UP, it could create a function correspond to those users action.

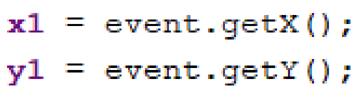


```
public class OnSwipeTouchListener implements View.OnTouchListener {
    private float x1,x2,y1,y2;
    private Context context;

    public OnSwipeTouchListener(Context context) { this.context = context; }

    @Override
    public boolean onTouch(View v, MotionEvent event) {

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN: //nhấn vô
               x1 = event.getX();
               y1 = event.getY();
               break;
            case MotionEvent.ACTION_UP: //thả ra
                float x2 = event.getX();
                float y2 = event.getY();
                float diffX = Math.abs(x2-x1);
                float diffY = Math.abs(y2-y1);
                Log.d( tag: "Swipe", msg: "diff X / Y " + String.valueOf(diffX) +" / " + diffY);
                if (diffX < 20 && diffY < 20) return true;
                if (diffX > diffY) { //horizontal
                    if (x2 > x1) onSwipeRight();
                    else onSwipeLeft();
                }
                else {
                    if (y2 > y1) onSwipeDown();
                    else onSwipeUp();
                }
                break;
        }
        return true;
    }
```

Attributes: x1, x2, y1, y2 (float) , context.

Variable with type “Context” represent the current state of the application. In another way, it could “tell” the newly- created object what has been going on the application.

x1 and y1 record the position of the mouse when we press the left mouse button by using the formula:



```
x1 = event.getX();
y1 = event.getY();
```

We define “event” as a variable of “Motion Event”

```
onTouch(View v, MotionEvent event)
```

"Motion events" describe movements in terms of an action code and a set of axis values. The action code specifies the state change that occurred such as a pointer going down or up. The axis values describe the position and other movement properties." (source: develop.android.com)

Therefore, we know that variable "event" have a function to describe the position as axis values, because we are using the 2nd dimension array so X-axis and Y-axis values are all we need.

Similarly, x2 and y2 record the position of the mouse cursor when un-hold the mouse.

```
float x2 = event.getX();
float y2 = event.getY();
```

```
float diffX = Math.abs(x2-x1);
float diffY = Math.abs(y2-y1);
```

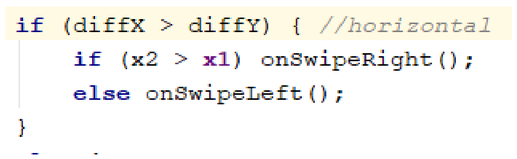
Next, there are 2 variables called "diffX" and "diffY". Those variable store results which are used for identify kind of motions (left, right, up, down).

```
if (diffX > diffY) { //horizontal
    if (x2 > x1) onSwipeRight();
    else onSwipeLeft();
}
else {
    if (y2 > y1) onSwipeDown();
    else onSwipeUp();
}
break;
```

When the mouse move horizontally, no matter left or right the absolute value of "x2-x1" will be greater than "y2-y1"(equals to zero). Then we have

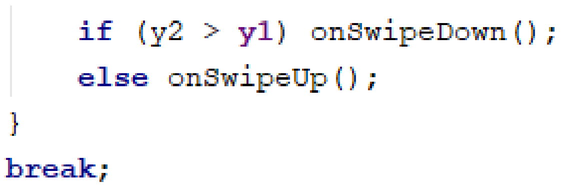
```
if (diffX > diffY) { //horizontal
```

After that we compare x1 and x2, as x1 is the first position and x2 is the last position. We know if x2 > x1 then move right, x2<x1 then move left.



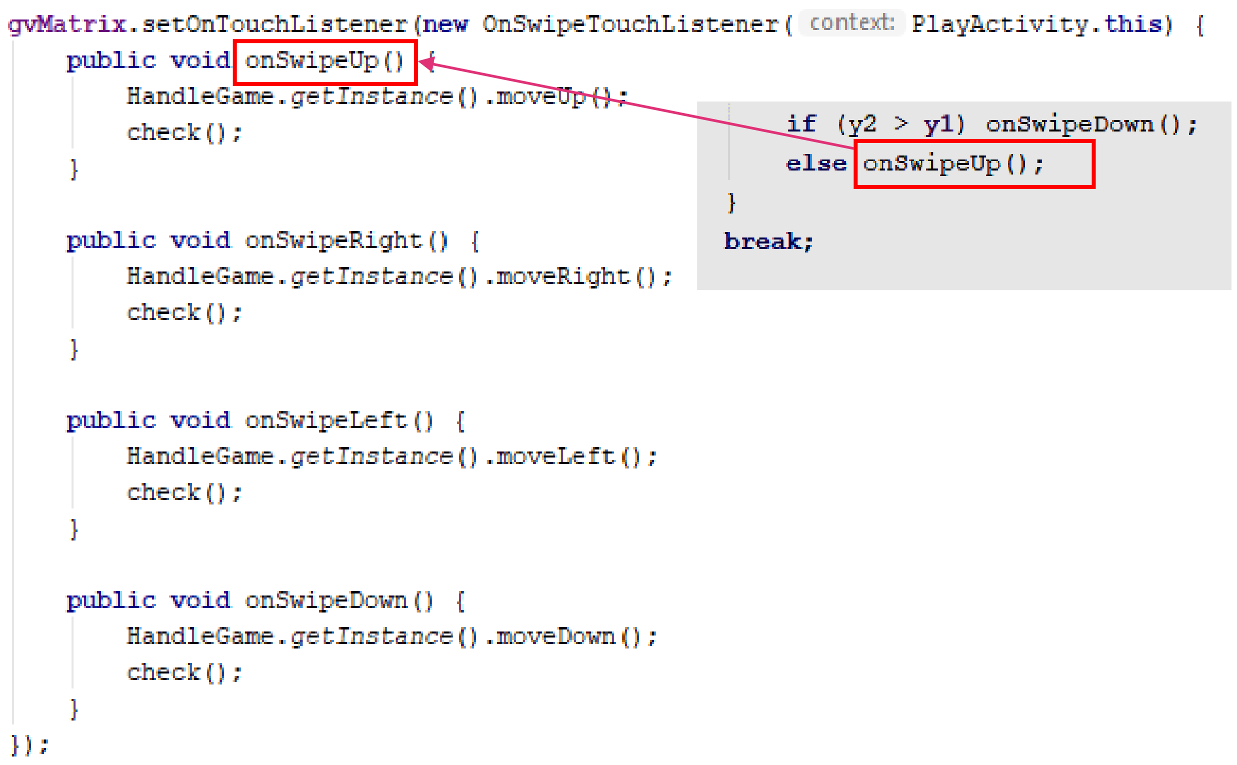
```
if (diffX > diffY) { //horizontal
    if (x2 > x1) onSwipeRight();
    else onSwipeLeft();
}
```

Similarly, when (y2 – y1)> (x2-x1) we know the mouse is moving vertically, then we compare y2 and y1 to confirm the motion is UP or Down



```
    if (y2 > y1) onSwipeDown();
    else onSwipeUp();
}
break;
```

Finally, we can check direction of event swipe by implement this class to any activity or implement as OnTouchListener. For example:



```
gvMatrix.setOnTouchListener(new OnSwipeTouchListener( context: PlayActivity.this) {
    public void onSwipeUp() {
        HandleGame.getInstance().moveUp();
        check();
    }

    public void onSwipeRight() {
        HandleGame.getInstance().moveRight();
        check();
    }

    public void onSwipeLeft() {
        HandleGame.getInstance().moveLeft();
        check();
    }

    public void onSwipeDown() {
        HandleGame.getInstance().moveDown();
        check();
    }
});
```

```
    if (y2 > y1) onSwipeDown();
    else onSwipeUp();
}
break;
```

6. Adapter

In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc. For more customization in Views we uses the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapter acts like a bridge between UI component and data source. Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.

There are the some commonly used Adapter in Android used to fill the data in the UI components.

- BaseAdapter – It is parent adapter for all other adapters
- ArrayAdapter – It is used whenever we have a list of single items which is backed by an array
- Custom ArrayAdapter – It is used whenever we need to display a custom list
- SimpleAdapter – It is an easy adapter to map static data to views defined in your XML file
- Custom SimpleAdapter – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

In this project, we use custom ArrayAdapter to display matrix in game and items in Shop to Gridview. Here is how our ItemAdapter looks :

```
ItemAdapter(Context context, int resource, ArrayList<Pets> array )
```

Lets discuss parameter in ArrayAdapter class:

- **context:**

The first parameter is used to pass the context means the reference of current class. Here this is a keyword used to show the current class reference. We can also use *getApplicationContext(), getActivity()* in the place of this keyword. *getApplicationContext()* is used in a Activity and *getActivity()* is used in a Fragment.

- **resource:**

The second parameter is resource id used to set the layout(xml file) for list items in which you want to show as an item.

- **objects:**

The fourth parameter is an array of objects, used to set the array of elements. We can set the object of array or array list here.

Below is the example code. *curBoard.getMatrix()* return an ArrayList in adapter to display the Item's list for matrix 4x4.

```
Gridview gvMatrix = (GridView) findViewById(R.id.gvMatrix);
adapter = new ItemAdapter(this, R.layout.item_pet,
                HandleGame.getInstance().curBoard.getMatrix());
gvMatrix.setAdapter(adapter);
```

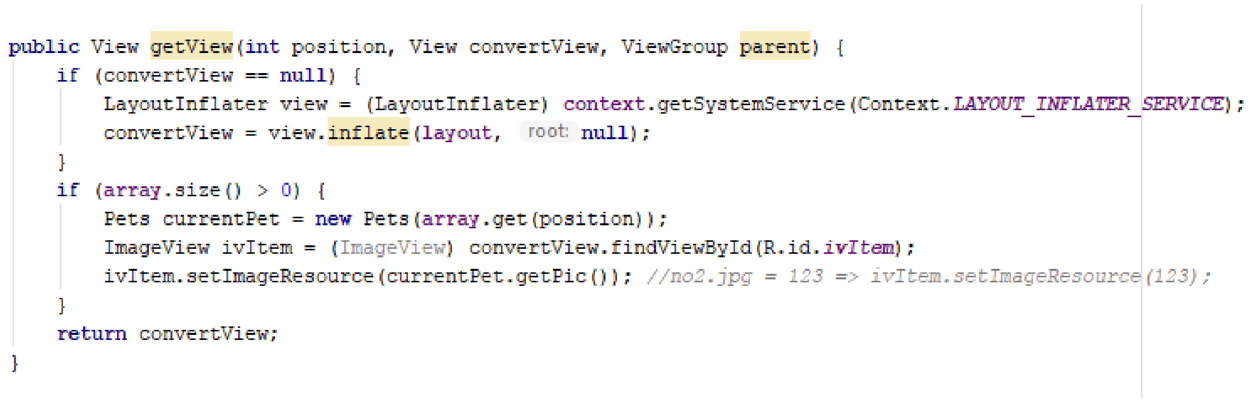
Coming to ShopAdapter.class:

```
    getView(int i, View view, ViewGroup viewGroup):
```

This function is automatically called when the list item view is ready to be displayed or about to be displayed. In this function we set the layout for list items using

LayoutInflater class and then add the data to the views like ImageView, TextView etc.

Below is the *getView* function's example code with explanation included in which we set the layout using *LayoutInflater* and then get the view's id and implement them.

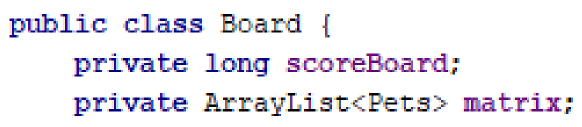


```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        LayoutInflater view = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = view.inflate(layout,  root: null);
    }
    if (array.size() > 0) {
        Pets currentPet = new Pets(array.get(position));
        ImageView ivItem = (ImageView) convertView.findViewById(R.id.ivItem);
        ivItem.setImageResource(currentPet.getPic()); //no2.jpg = 123 => ivItem.setImageResource(123);
    }
    return convertView;
}
```

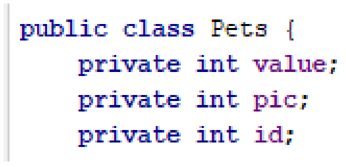
7. Handle Game

HandleGame is a singleton class doing all the action needed on the game such as movement, plus numbers, handle some features.

To support this class, we have more some models objects:



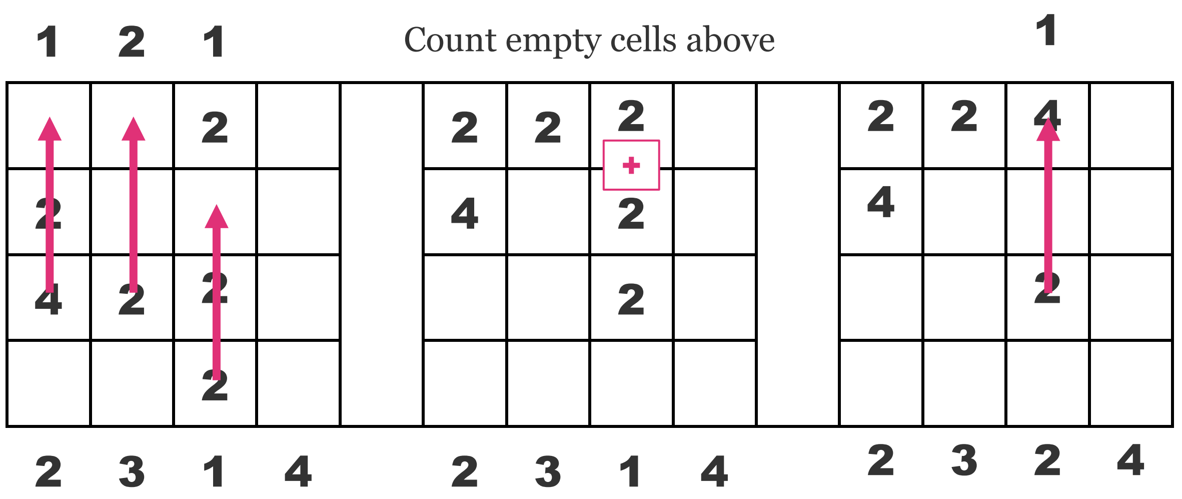
```
public class Board {
    private long scoreBoard;
    private ArrayList<Pets> matrix;
```



```
public class Pets {
    private int value;
    private int pic;
    private int id;
```

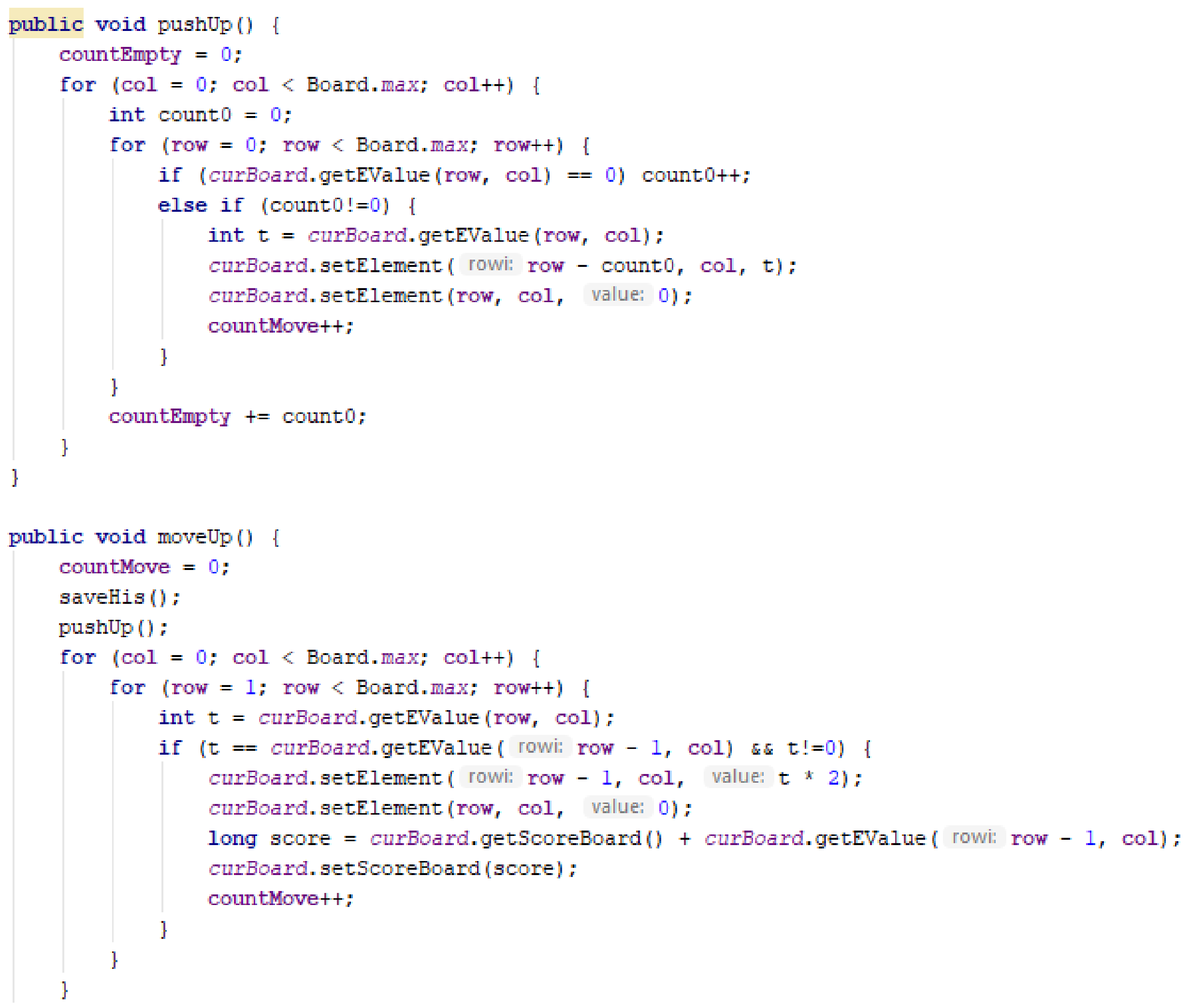
Board object contains ArrayList "matrix" with size 4*4. It represents for matrix in 2048 game. And "pets" is an item in that board.

In our idea, we give HandleGame a board to play (*Board curBoard*) and divide swipe action into two times push. It means that when the player swipe up, all the element will push up without calculating to fill in all the hole on top. Then HandleGame will check upper element whether it is same value to sum up. If it has some summation, it maybe exist some hole among them. Therefore, curBoard pushes one more time. The numbers are pushed up by counting the empty tile above them. And do the same as other directions.



=> sum all the empty cells to support adding random number.

Besides, we also check number of moved cells to know the non-move case.



```java
public void pushUp() {
    countEmpty = 0;
    for (col = 0; col < Board.max; col++) {
        int count0 = 0;
        for (row = 0; row < Board.max; row++) {
            if (curBoard.getEValue(row, col) == 0) count0++;
            else if (count0!=0) {
                int t = curBoard.getEValue(row, col);
                curBoard.setElement( row: row - count0, col, t);
                curBoard.setElement(row, col,  value: 0);
                countMove++;
            }
        }
        countEmpty += count0;
    }
}

public void moveUp() {
    countMove = 0;
    saveHis();
    pushUp();
    for (col = 0; col < Board.max; col++) {
        for (row = 1; row < Board.max; row++) {
            int t = curBoard.getEValue(row, col);
            if (t == curBoard.getEValue( row: row - 1, col) && t!=0) {
                curBoard.setElement( row: row - 1, col,  value: t * 2);
                curBoard.setElement(row, col,  value: 0);
                long score = curBoard.getScoreBoard() + curBoard.getEValue( row: row - 1, col);
                curBoard.setScoreBoard(score);
                countMove++;
            }
        }
    }
    if (countMove > 0) {
        countMove = 0;
        pushUp();
        addRandomNumber();
    } else {
        undoSaveHis();
    }
}
```

The GameOver case will be recognized when there is neither hole nor summation.

By applying ArrayList to list of numbers on board, we can use adapter and gridview (section 6) to show those ones on screen.

8. Some Features

- **Undo**

It is very easy for us to save history of movement by deep copy from *curBoard* into *Stack<Board> boardStack*:

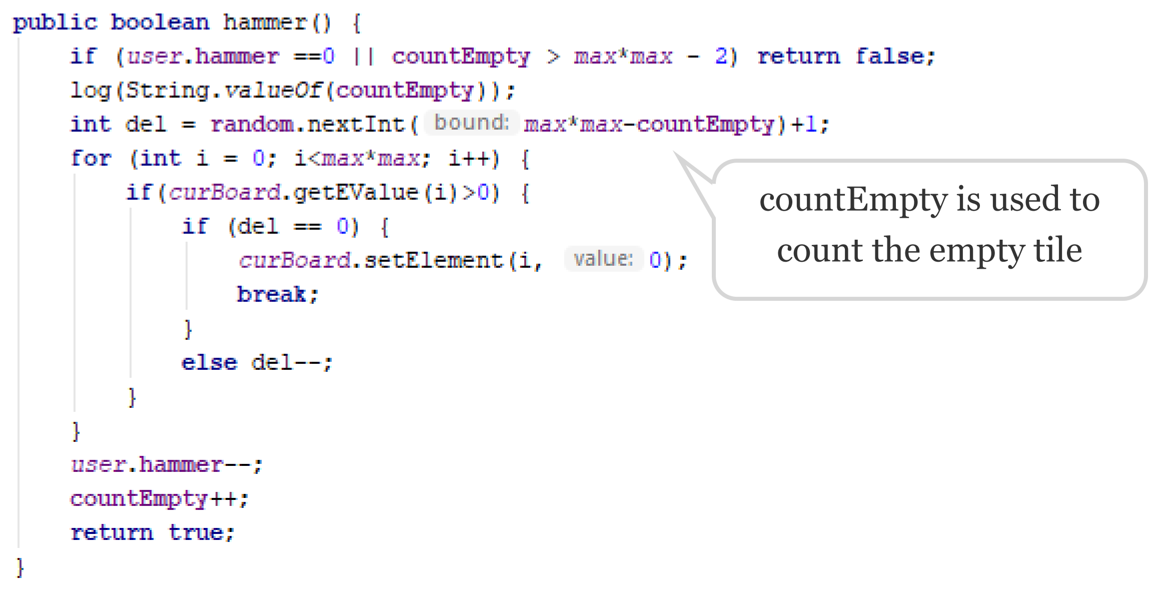
```
public void saveHis() {
    Board boardTemp = new Board(curBoard);
    boardStack.push(boardTemp);
}
```

```
public Board undoSaveHis() {
    return boardStack.pop();
}
```

So whenever the player swipe, this method will be called saveHis(), and undo by return the *Board* on the top.

- **Hammer**

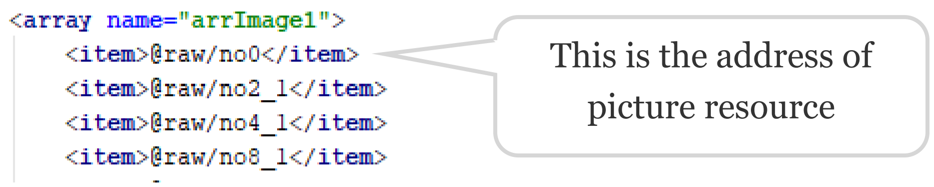
Hammer can break a tile randomly when there are more then 2 tiles on board.



```
public boolean hammer() {
    if (user.hammer ==0 || countEmpty > max*max - 2) return false;
    log(String.valueOf(countEmpty));
    int del = random.nextInt( bound: max*max-countEmpty)+1;
    for (int i = 0; i<max*max; i++) {
        if(curBoard.getEValue(i)>0) {
            if (del == 0) {
                curBoard.setElement(i,  value: 0);
                break;
            }
            else del--;
        }
    }
    user.hammer--;
    countEmpty++;
    return true;
}
```

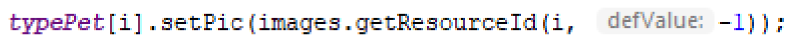
- **Set theme**

We give each value a picture to represent on screen. So we just need to change the picture that we assign for *Pets* object with the help of string array in string.xml file.

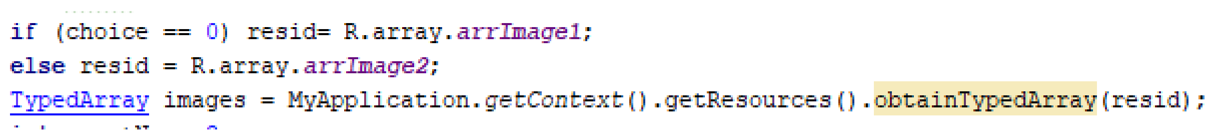


```
<array name="arrImage1">
    <item>@raw/no0</item>
    <item>@raw/no2_1</item>
    <item>@raw/no4_1</item>
    <item>@raw/no8_1</item>
```

Therefore, it is very easy for us to call and assign picture to each element (as I have shown in UI design section):



```
typePet[i].setPic(images.getResourceId(i,  defValue: -1));
```

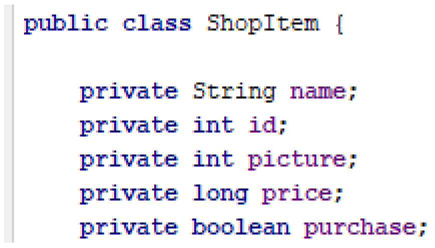


```
if (choice == 0) resid= R.array.arrImage1;
else resid = R.array.arrImage2;
TypedArray images = MyApplication.getContext().getResources().obtainTypedArray(resid);
```

This allow player buying and trying different theme for new experience.

9. Shop (ProfileActivity):

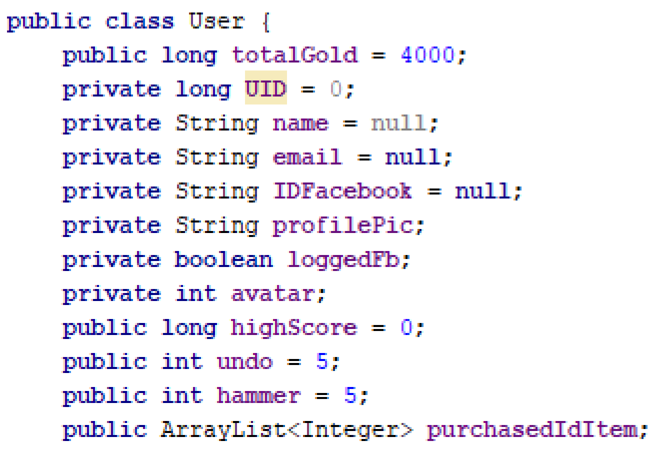
Just like *HandleGame*, we need *ShopItem.class* to store data of each item for sale, ArrayList to store items and use gridview to show those ones:



```
public class ShopItem {

    private String name;
    private int id;
    private int picture;
    private long price;
    private boolean purchase;
```

Our shop now has 5 types of items: undo, hammer, avatar, facebook avatar, theme. They are distinguish by first character in *id*.

10. User



```
public class User {
    public long totalGold = 4000;
    private long UID = 0;
    private String name = null;
    private String email = null;
    private String IDFacebook = null;
    private String profilePic;
    private boolean loggedFb;
    private int avatar;
    public long highScore = 0;
    public int undo = 5;
    public int hammer = 5;
    public ArrayList<Integer> purchasedIdItem;
```

This object contains information of the player such as: name, custom setting, facebook account, score, gold, purchased item…

We will update these attributes frequently throughout the app.

CHAP 4:
FINAL APP GAME

Source code (link github):

```
https://github.com/suhymin97/2048Pets
```

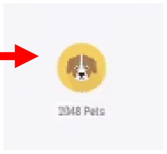
Demo video:

```
https://youtu.be/GBvmYQE_RLs
```

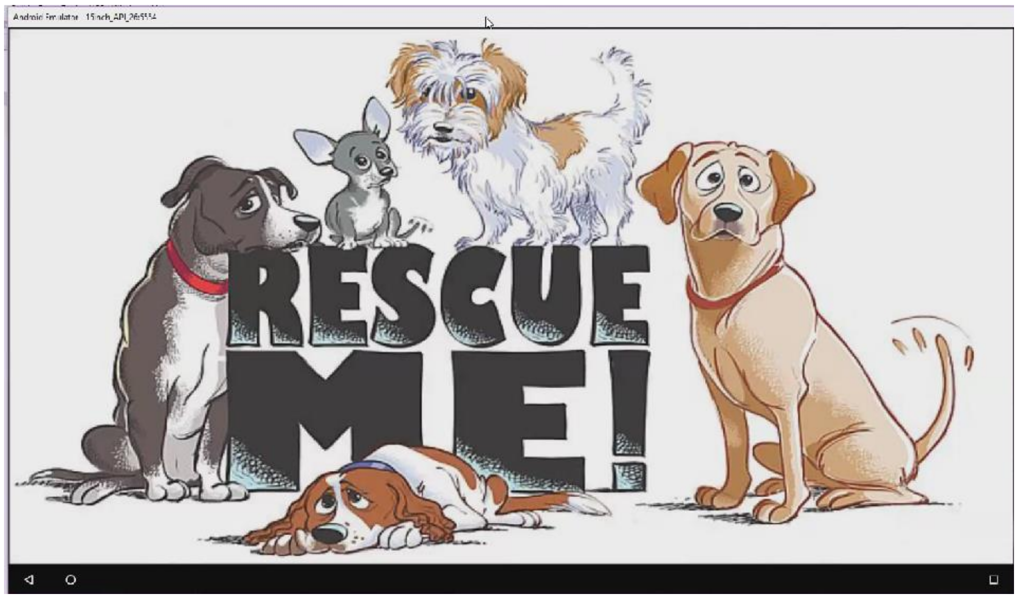
Instruction

1. Begin the game:

Click on the icon to open the game.

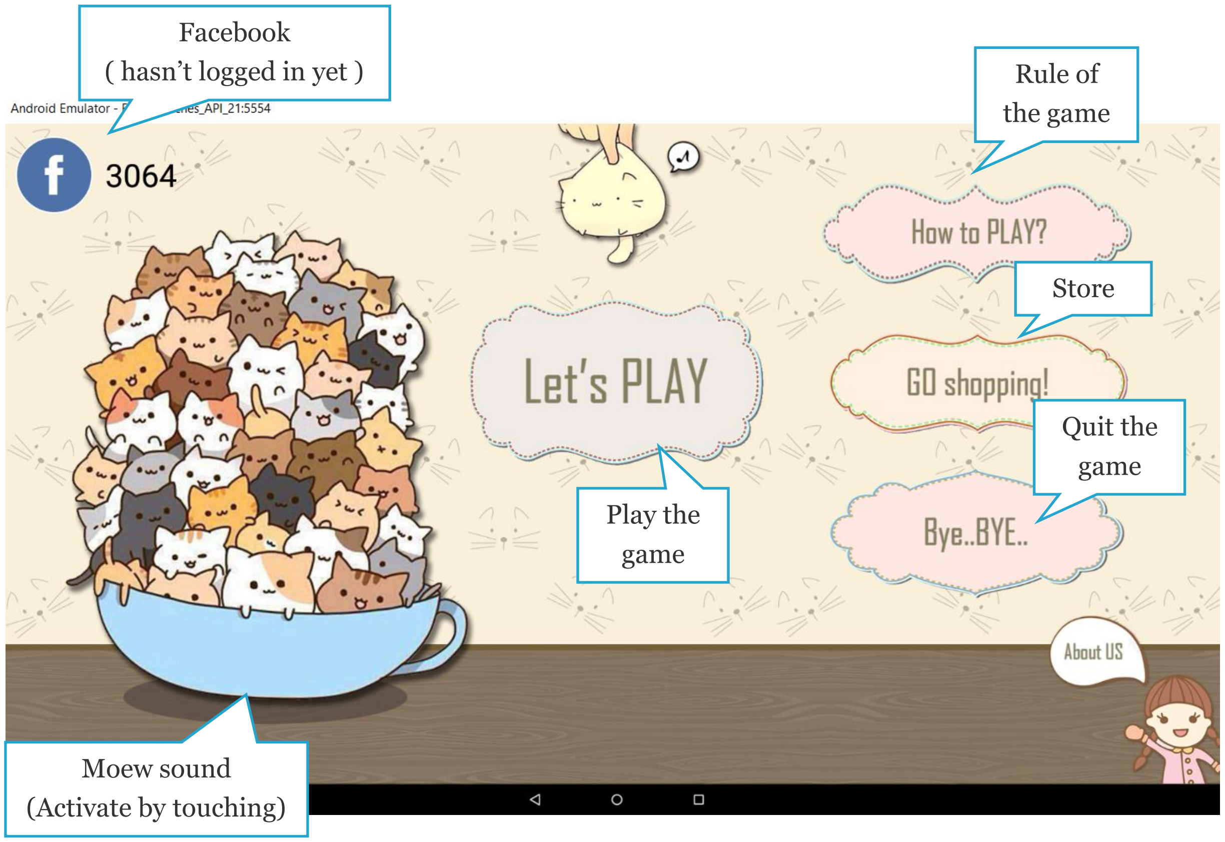


Loading screen will appear.



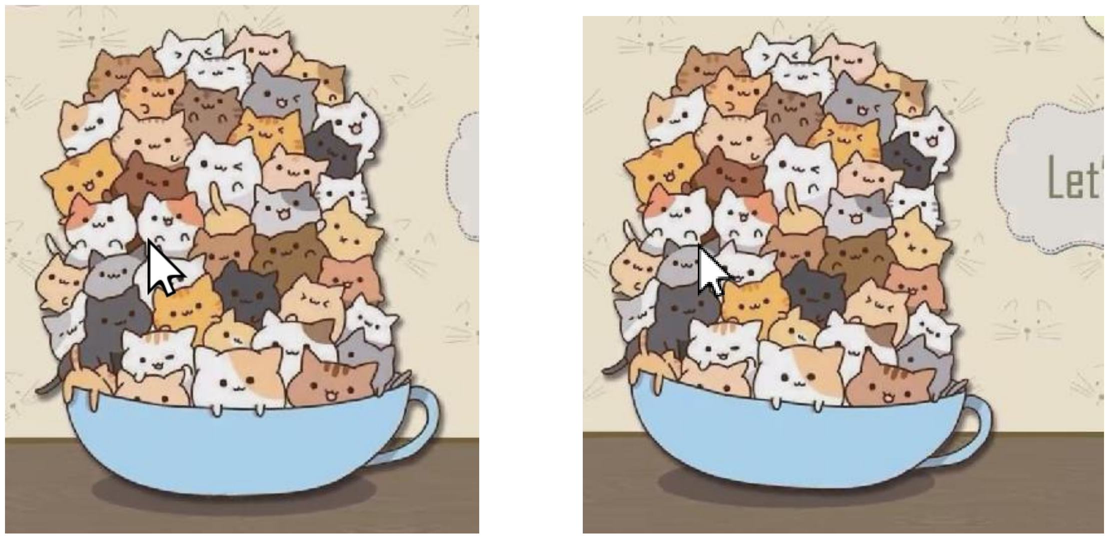
Click on the backspace to enter the “Main menu”

2. Main Menu:



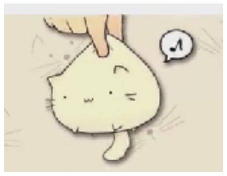
We add the animation and sounds for the "cup of cats" and the "sound cat" :

- The cup of cats moves "y-axis" both sides:



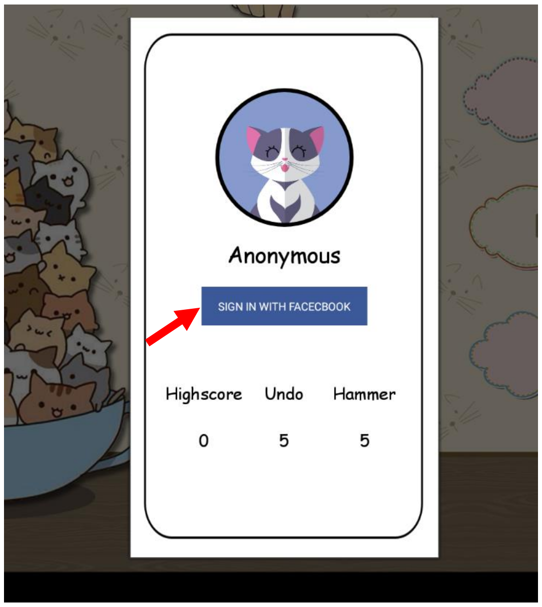
- When you move the mouse the "cup of cats", the sound "meow" is ON with the ***fade-in*** effect, and it is OFF with the ***fade-out*** effect when your mouse is out of that.

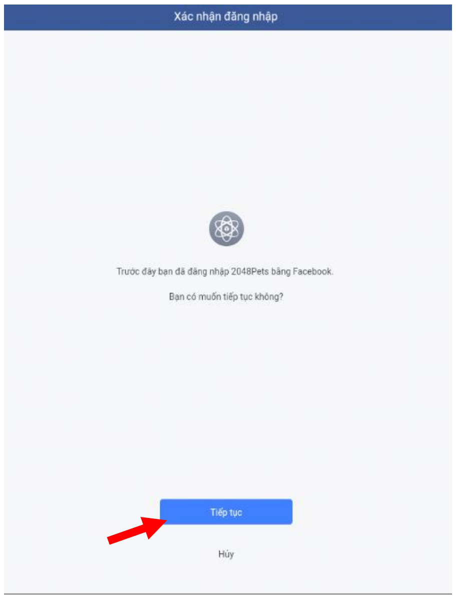
- The sound cat moves “y-axis” both sides:



3. Facebook:

Click on the Facebook button in the “Main menu” then log in your account.




Xác nhận đăng nhập

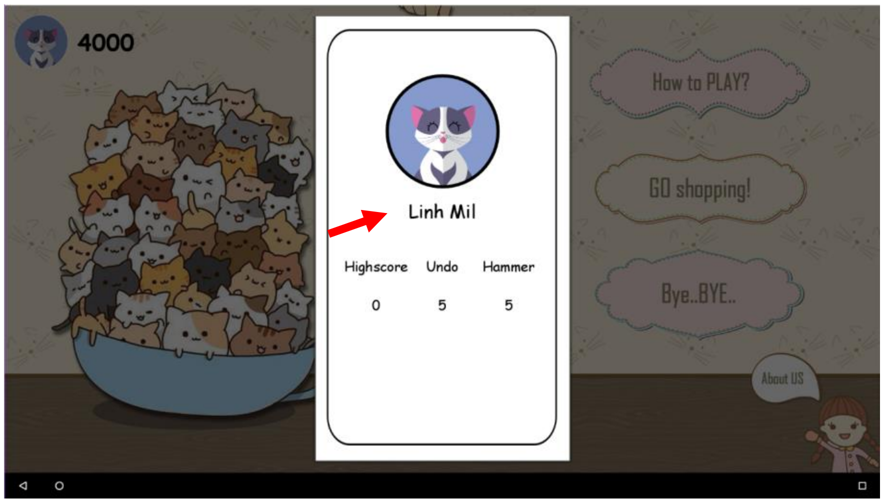
Trước đây bạn đã đăng nhập 2048Pets bằng Facebook.

Bạn có muốn tiếp tục không?

Tiếp tục

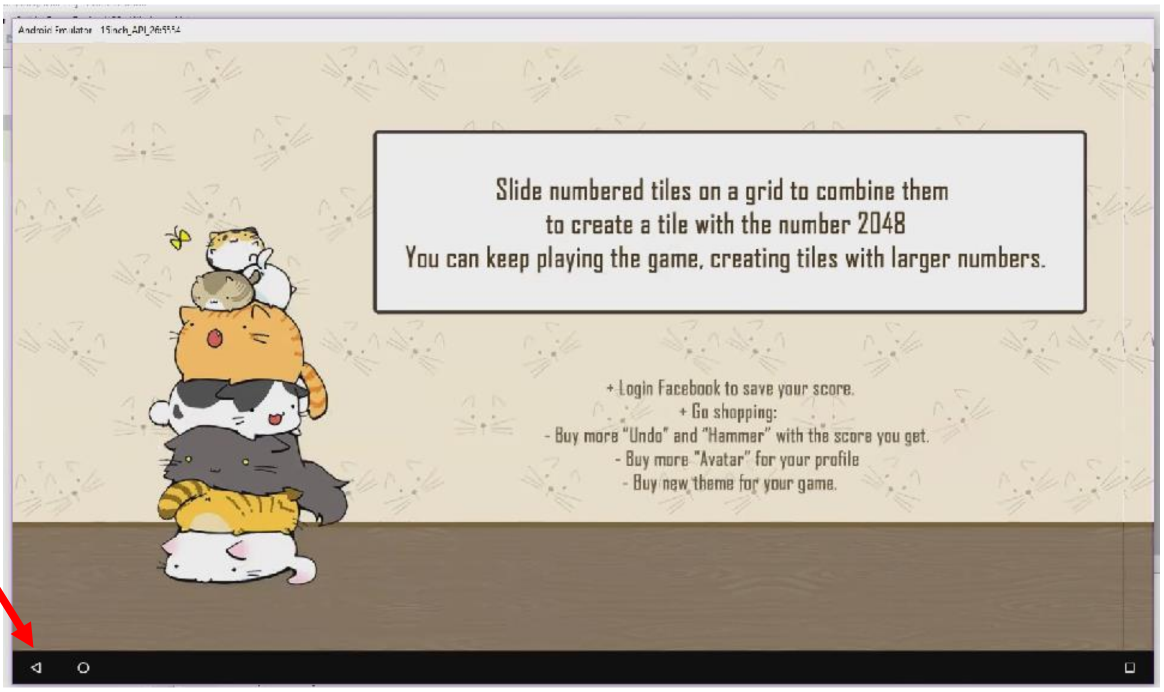
Hủy

Now, you’ve already logged in your Facebook account to save your data of the game.



4. How to play:

Show user the rule of the game:



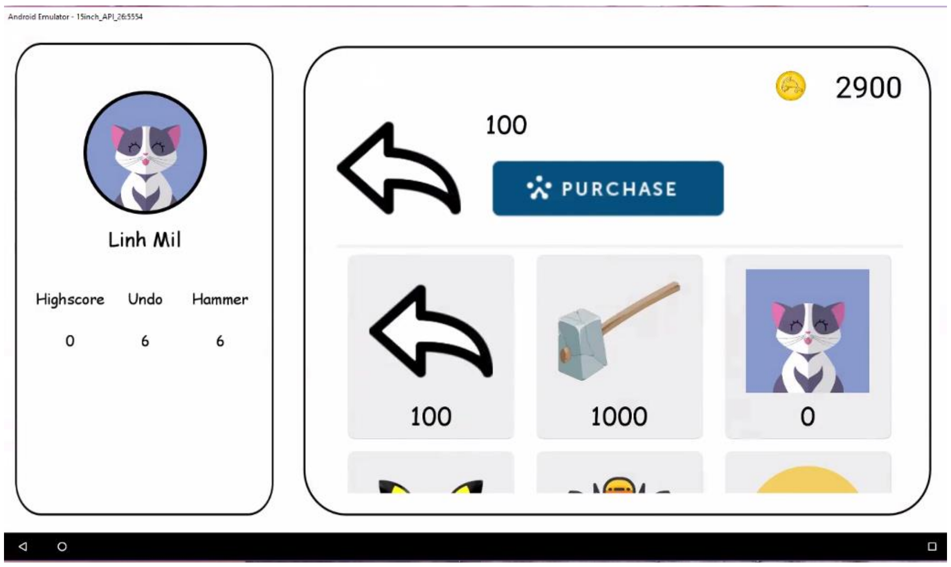
Slide numbered tiles on a grid to combine them
to create a tile with the number 2048
You can keep playing the game, creating tiles with larger numbers.

+ Login Facebook to save your score.
+ Go shopping:
- Buy more "Undo" and "Hammer" with the score you get.
- Buy more "Avatar" for your profile
- Buy new theme for your game.

Click on the backspace to get back to the "Main menu"

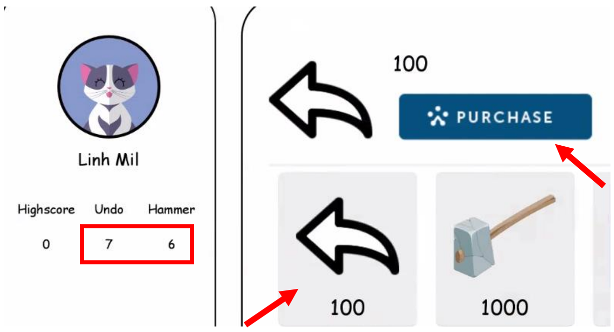
5. Go shopping:

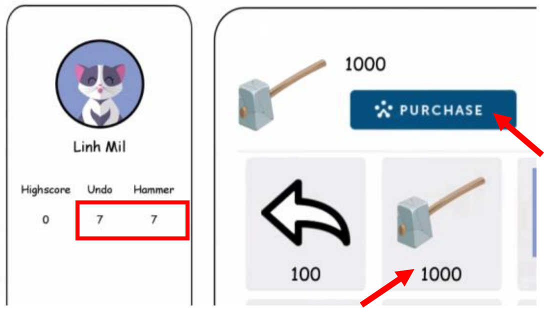
This is the store where User can buy some more stuffs for the game.



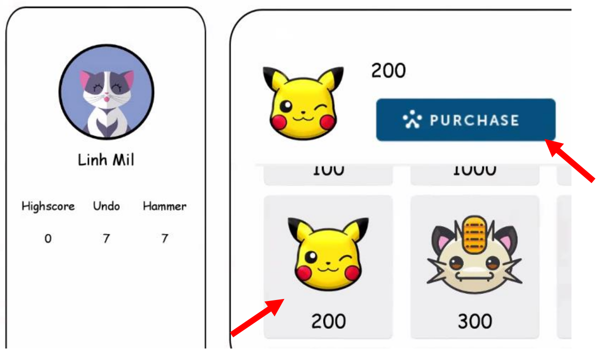
Use the money which is the score you gained from the game to buy the stuffs. Click on the stuff you want to buy and click PURCHASE. For example:

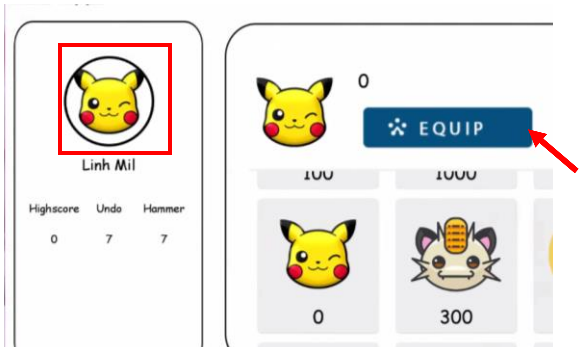
- Buy 1 Undo and 1 Hammer.



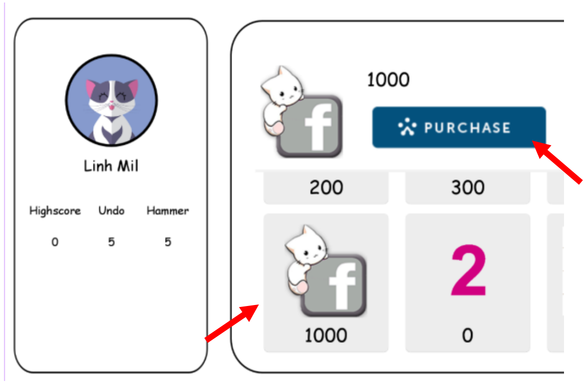


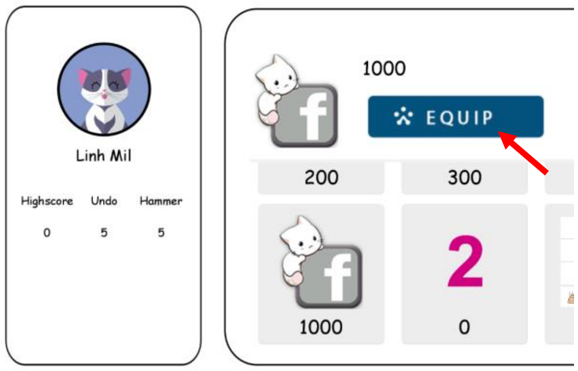
- Buy a new avatar for your profile and then apply it by clicking EQUIP

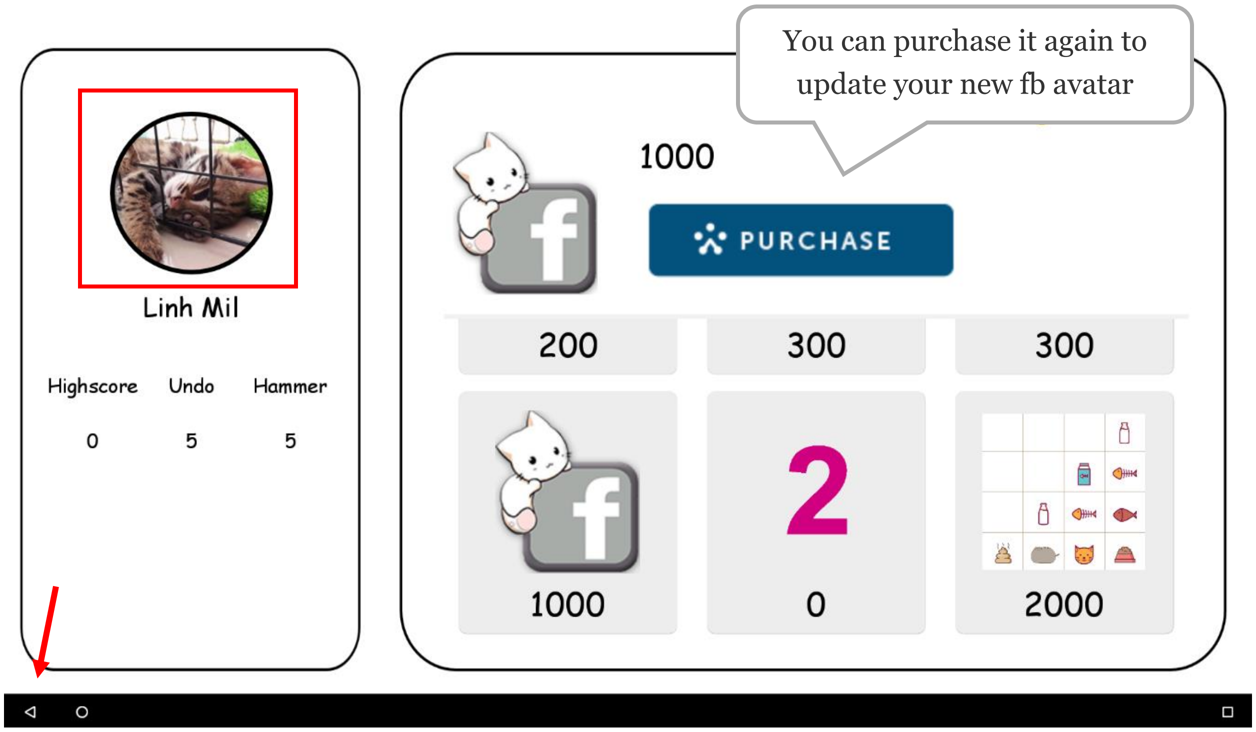




- Or change game avatar to facebook avatar:



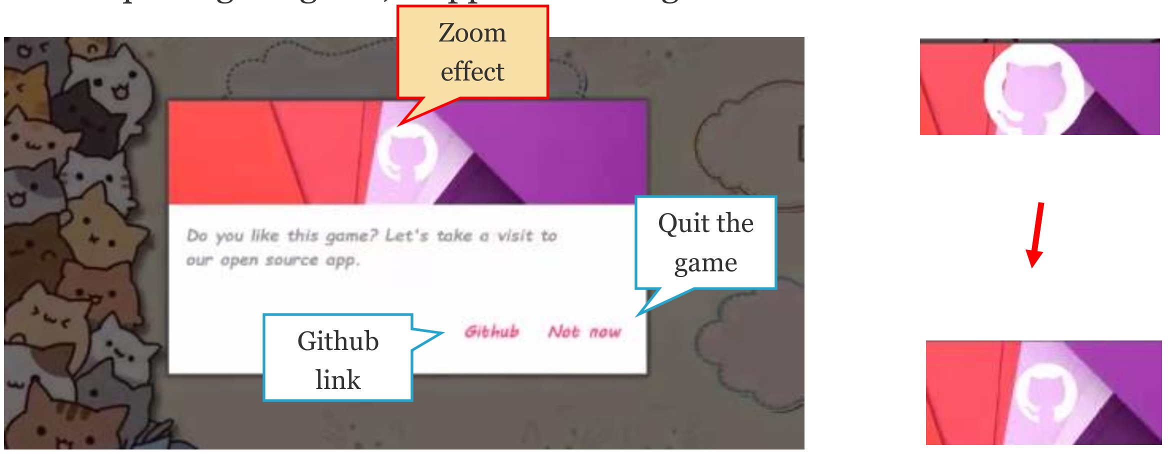




Click on the backspace to get back to the Previous screen.

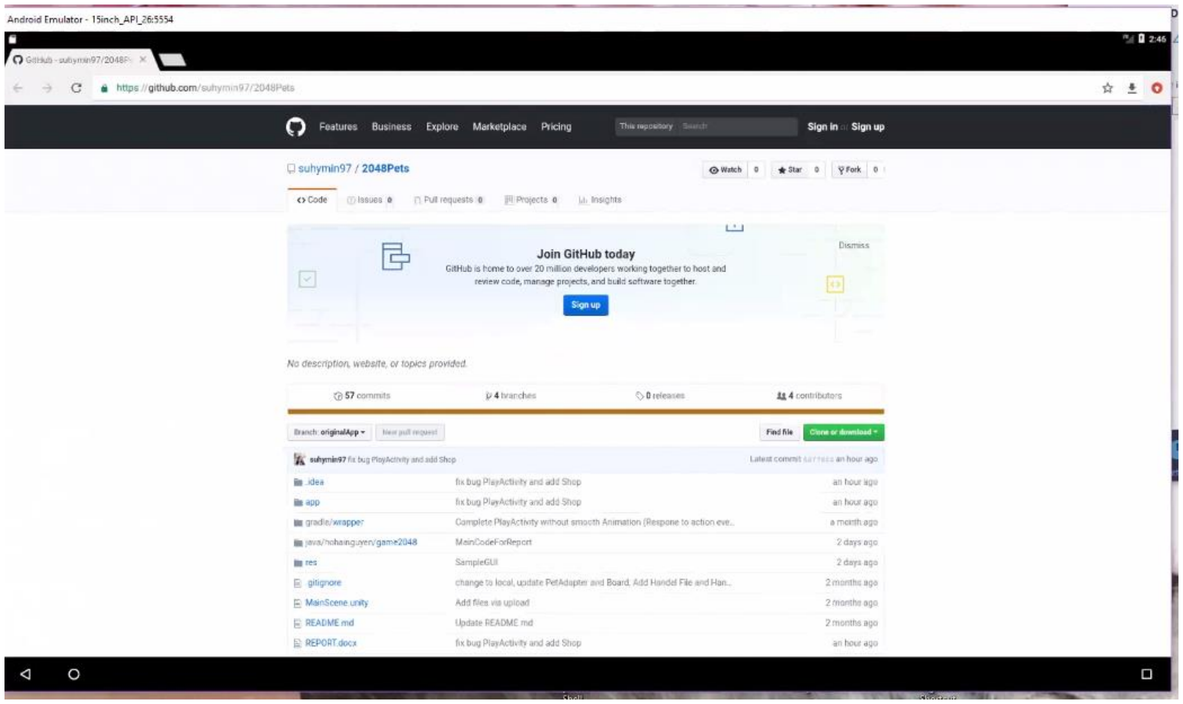
6. Bye.. bye – Backspace button:

Before quitting the game, it appears a dialogue:



Do you like this game? Let's take a visit to our open source app.

By clicking to "Github", you will be immediately moved to our Github link:



Android Emulator - 15inch_API_26:5554

https://github.com/suhymin97/2048Pets

Features Business Explore Marketplace Pricing Sign in or Sign up

suhymin97 / 2048Pets

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

No description, website, or topics provided.

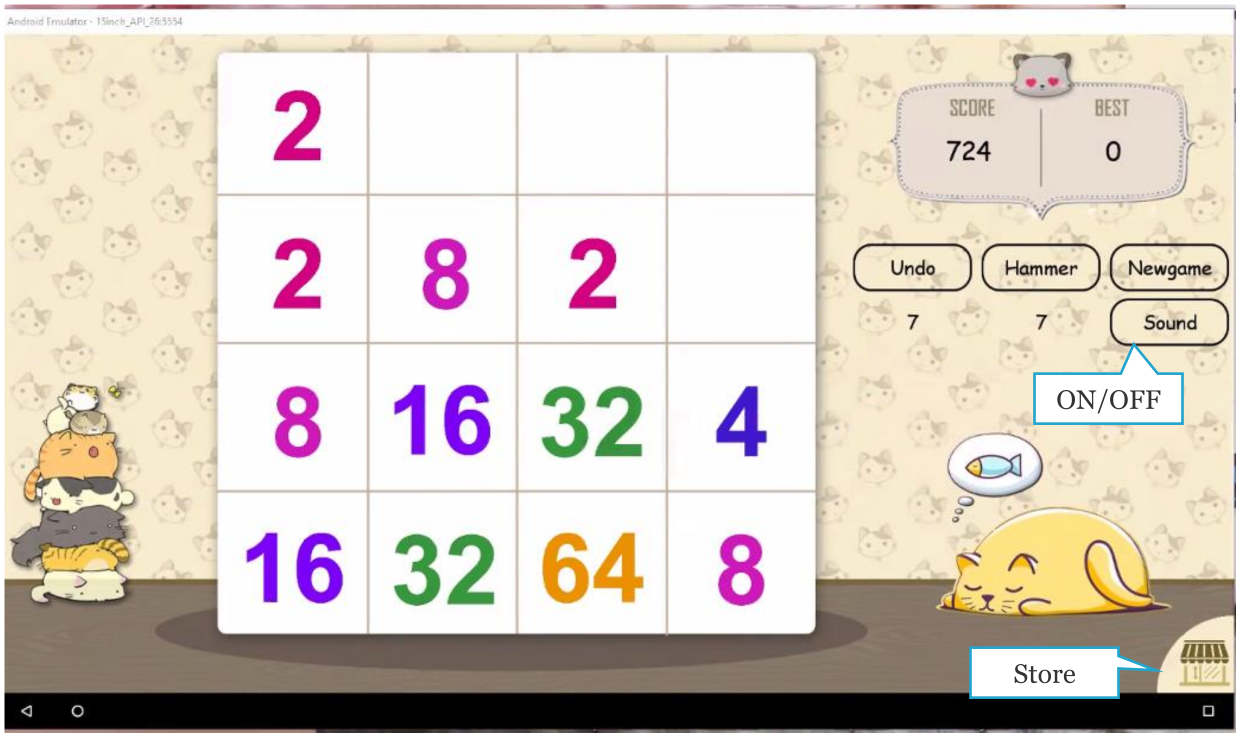
57 commits 4 branches 0 releases 4 contributors

Branch: originalApp New pull request Find file Clone or download

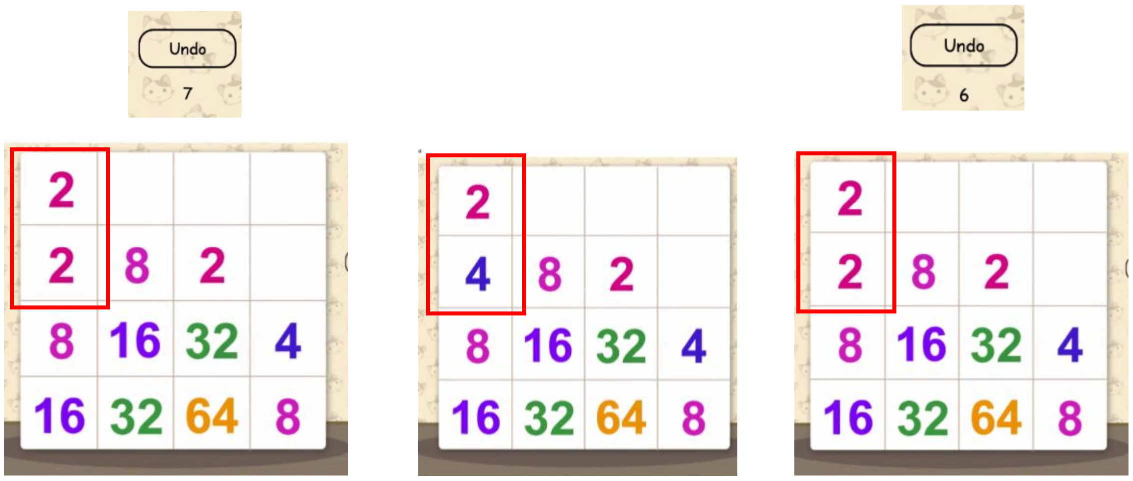
suhymin97 fix bug PlayActivity and add Shop — Latest commit an hour ago

.idea	fix bug PlayActivity and add Shop	an hour ago
app	fix bug PlayActivity and add Shop	an hour ago
gradle/wrapper	Complete PlayActivity without smooth Animation (Respone to action eve…	a month ago
java/nohainguyen/game2048	MainCodeForReport	2 days ago
res	SampleGUI	2 days ago
.gitignore	change to local, update PetAdapter and Board. Add Handel File and Han…	2 months ago
MainScene.unity	Add files via upload	2 months ago
README.md	Update README.md	2 months ago
REPORT.docx	fix bug PlayActivity and add Shop	an hour ago

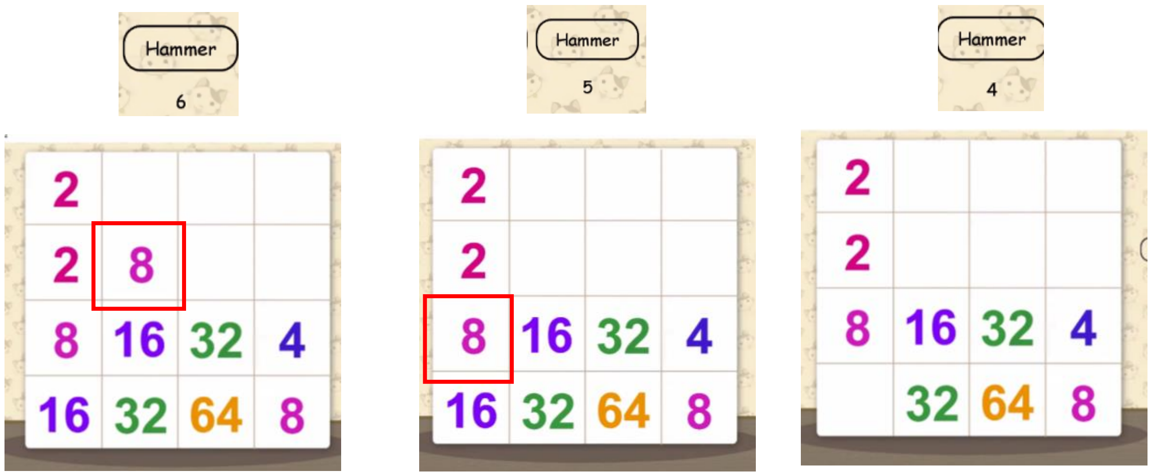
7. Let's play:



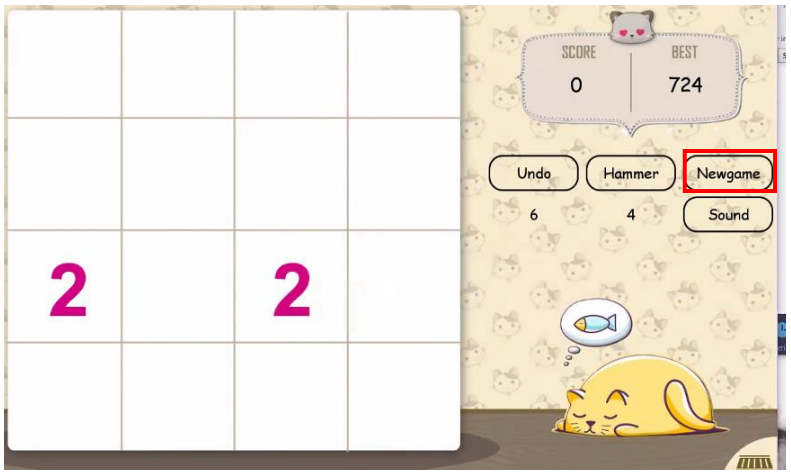
- **UNDO**: Back to previous step



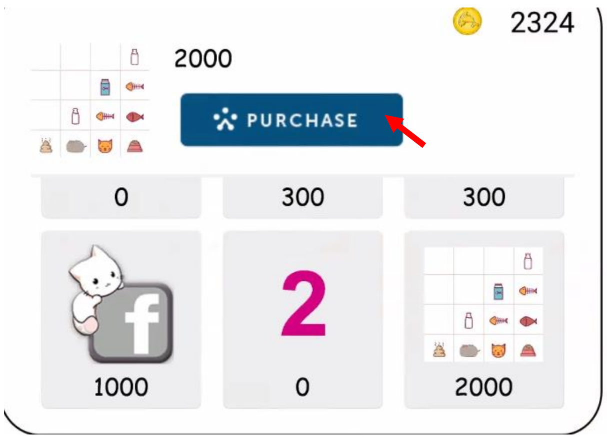
- **HAMMER**: **randomly** delete 1 element

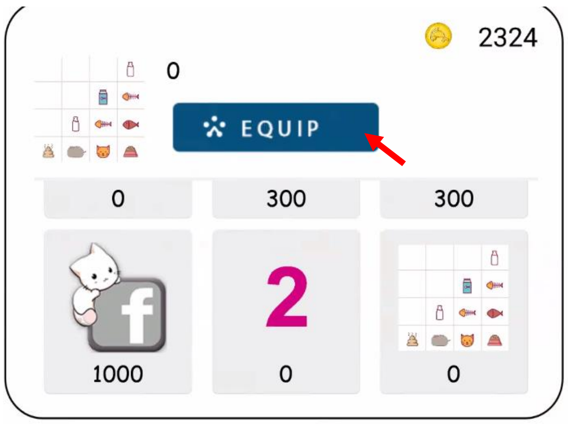


- **NEWGAME**:

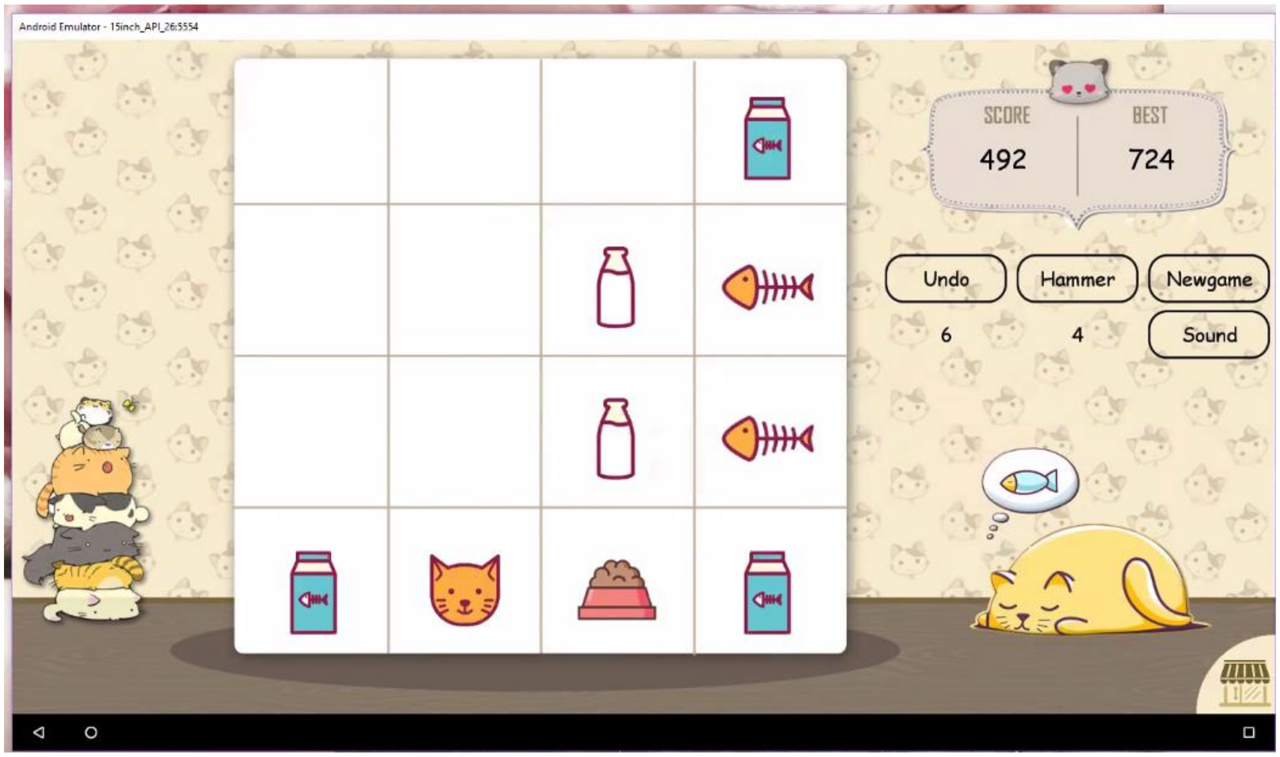


Let's go to the store to buy a new theme and apply it:

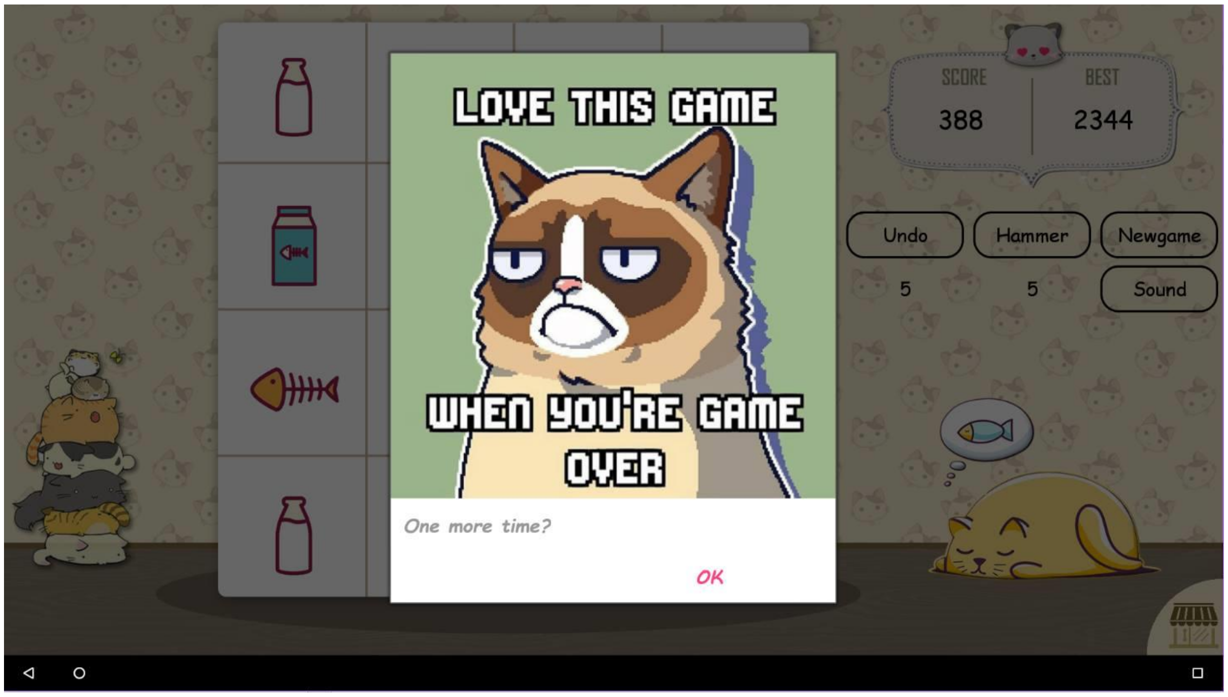




And enjoy your new theme:

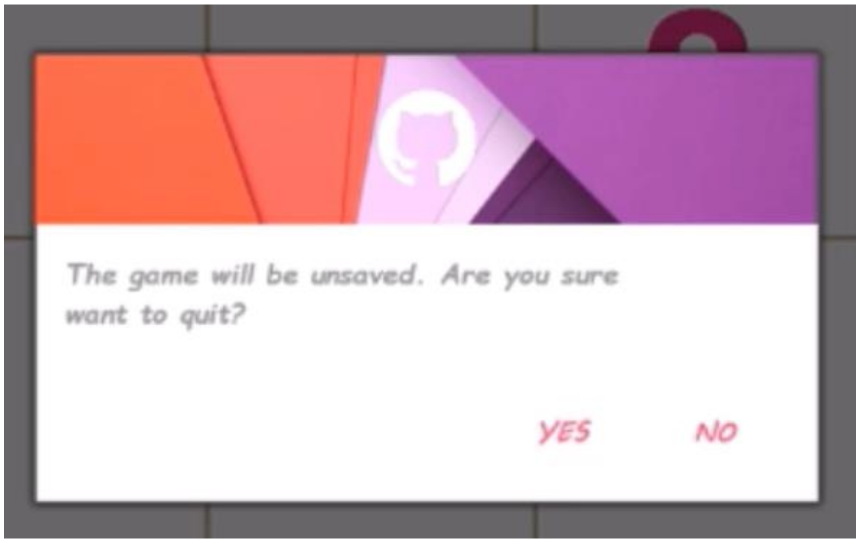


When you lose, a notification will appear, you cannot click on anything else except yes button:



One more time?

If you're playing the game and click the Backspace button, a dialogue will appear to notify you because, the match won't be save for later access:



The game will be unsaved. Are you sure want to quit?

CHAP 5: EXPERIENCE

In a game project, the product is a game. And after almost two months working on it, we all come to the point:

A game is much more than just its software.

It has to provide content to become enjoyable. Just like a web server: *Without content the server is useless, and the quality cannot be measured*.

During the time working on the project, our team realize that the software part of the game is not the only one, and it must be considered in connection to all other parts: the environment of the game, the story, game plays, the artwork, animation, and so on.

We have chance to practice a lot of lesson which we have learnt on class, and deals with plenty of bugs. Besides, it encourages students study more outside the class. Therefore, we not only firm the knowledge from lesson, but also update information technology and have new experience.

Finally, our team realizes that Computer Science major need its student self-study most of the time. IT world is too big to wait and just learn from school.

Homies team will try our best to complete this game as first project to publish it as soon as we can and make more android app to support for users.

- END -