



# 2048 PETS

## FINAL PROJECT

### MA

*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.

**Simple game**

**Easy to be  
addicted**

**UI eye-catching**

**More features**

**Play with friends**

#### **GAME DEVELOPER**

Vu Minh Tu Anh

Nguyen Duc Cong

Nguyen Hoang Anh Tuan

#### **ADVISOR**

Prof. Le Duy Tan



## Contents

CHAP 1: INTRODUCTION.....	3
1.1    Gaming in the Field: .....	3
1.2    About the game project: .....	3
1.3    Our 2048 Pets game:.....	4
1.4    References: .....	4
1.5    Developer team: .....	5
CHAP 2: SOFTWARE REQUIREMENTS.....	7
2.1    What we have: .....	7
2.2    What we want:.....	7
2.3    Working tools, platform: .....	8
2.4    Use Case Scenario .....	8
2.5    Use Case diagram .....	9
2.6    Class diagram .....	10
Play process diagram .....	10
Shop process diagram .....	11
Profile diagram .....	12
User Profile Activity .....	13
CHAP 3: DESIGN & IMPLEMENTATION .....	14
Package Diagram.....	<b>Error! Bookmark not defined.</b>
UI Design .....	14
Basic Android .....	18
1.    Application .....	18
2.    Palette attributes: .....	18
3.    Map variables: .....	19
4.    Debug .....	19

5. Activity .....	20
6. Fragment .....	24
7. Sound & Animation Effect .....	27
8. Dialog .....	28
Main methods .....	29
1. Handle file .....	30
2. Facebook Connect: .....	32
3. Handle image .....	37
4. Handle sound .....	39
5. OnSwipeTouchListener (implements OntouchListener) .....	40
6. Adapter .....	49
7. Handle Game .....	51
8. Some Features .....	53
9. Shop (ProfileActivity): .....	54
10. User .....	55
CHAP 4: FINAL APP GAME .....	56
Source code (link github): .....	56
Demo video: .....	<b>Error! Bookmark not defined.</b>
Instruction .....	56
1. Begin the game: .....	56
2. Main Menu: .....	56
3. Facebook: .....	58
4. How to play: .....	59
5. Go shopping: .....	59
6. Bye.. bye – Backspace button: .....	61
7. Let's play: .....	62
CHAP 5: EXPERIENCE .....	66

# 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, Google 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=PL6gx4Cw19DGBsvRxJJ0zG4r4k\\_zLKrnXl](https://www.youtube.com/playlist?list=PL6gx4Cw19DGBsvRxJJ0zG4r4k_zLKrnXl)  
<https://www.youtube.com/playlist?list=PLzrVYRai0riSRJ3M3bifVwWRq5eJMu6tv>

- Facebook connect  
<https://developers.facebook.com/docs/facebook-login/android>
- Google connect  
<https://developers.google.com/identity/sign-in/android/start-integrating>

**Formatted:** Default Paragraph Font

### 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
Vu Minh Tu Anh - <a href="#">suhymn97</a>	ITITIU15093	Write report UI/UX Activities (Menu, shop, game...) Facebook login Socket
Nguyen Duc Cong - <a href="#">ITITIU19087</a>	ITITIU19087	Write report Google login + facebook login Support fix bugs
Nguyen Duc Cong - <a href="#">atuans</a>	ITITIU19233	Write report, power point Support fix bugs UI Testing

**Formatted:** Font: Italic

**Formatted:** Font: +Body (Georgia)





# 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: 31, target API 31) with addition library

a. Facebook sdk

```
implementation 'com.facebook.android:facebook-android-sdk:12.0.0'
```

b. Circle Image View

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

c. Picasso

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

d. Google sdk

```
implementation 'com.google.android.gms:play-services-auth:20.4.0'
```

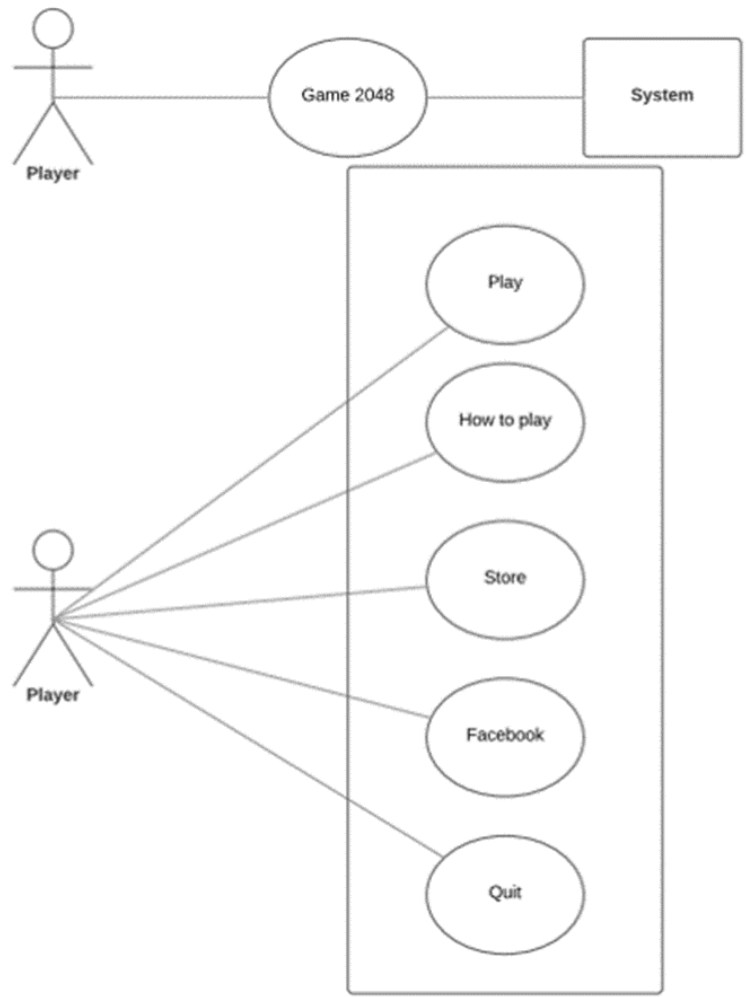
### 2.4 Use Case Scenario

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
	SOCIAL LOGIN	Log in to save your score

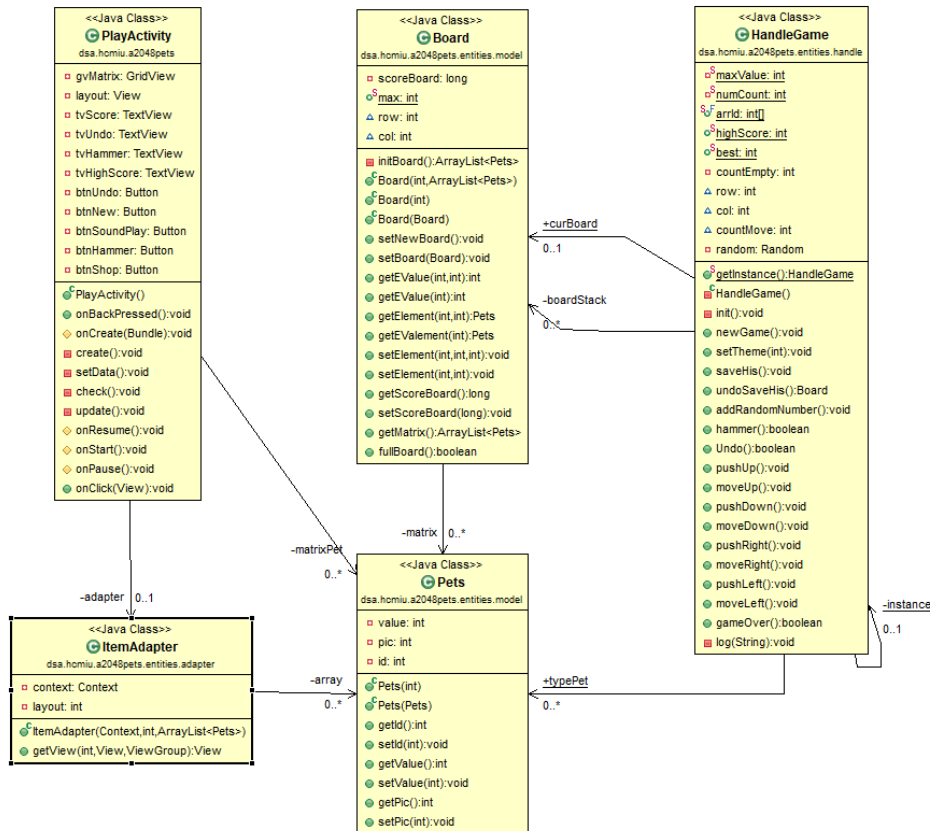
	QUIT	Exit the game
--	------	---------------

**2.5 Use Case diagram**

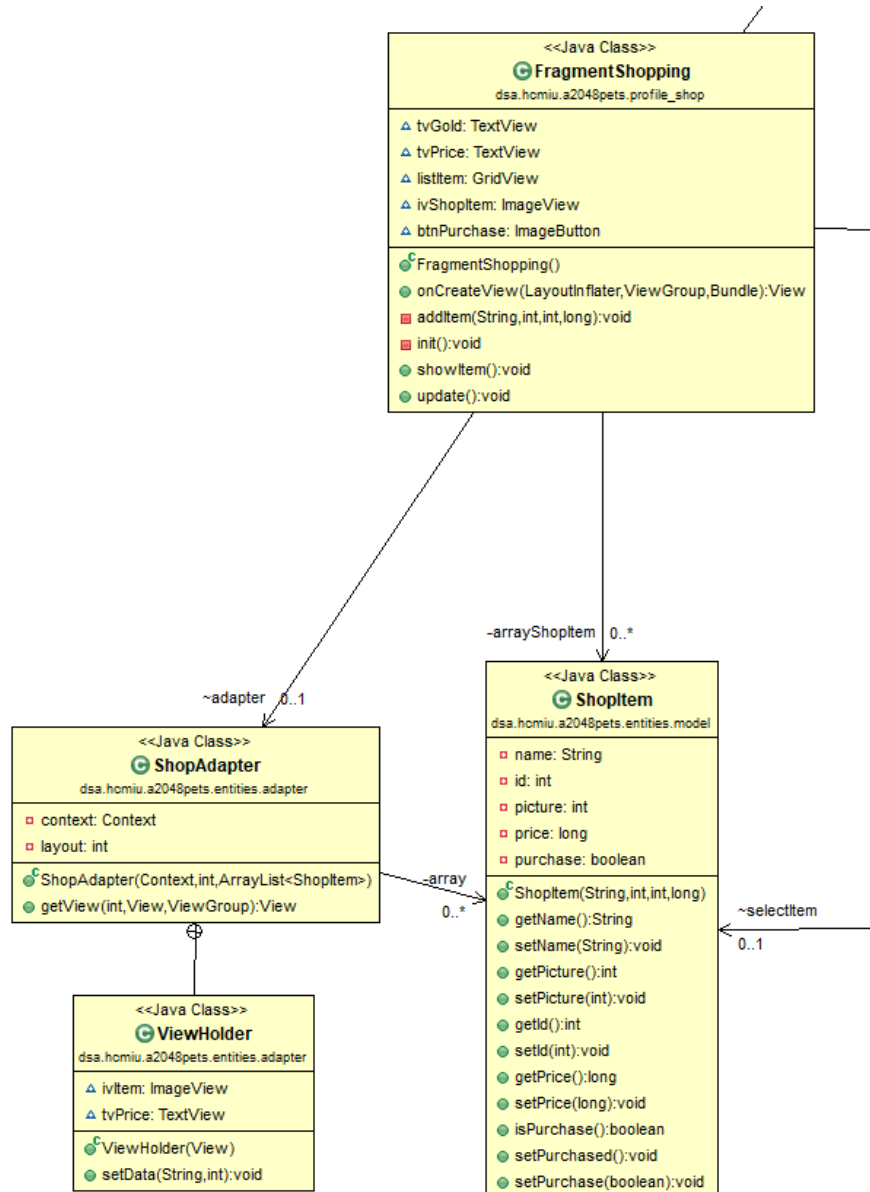


## 2.6 Class diagram

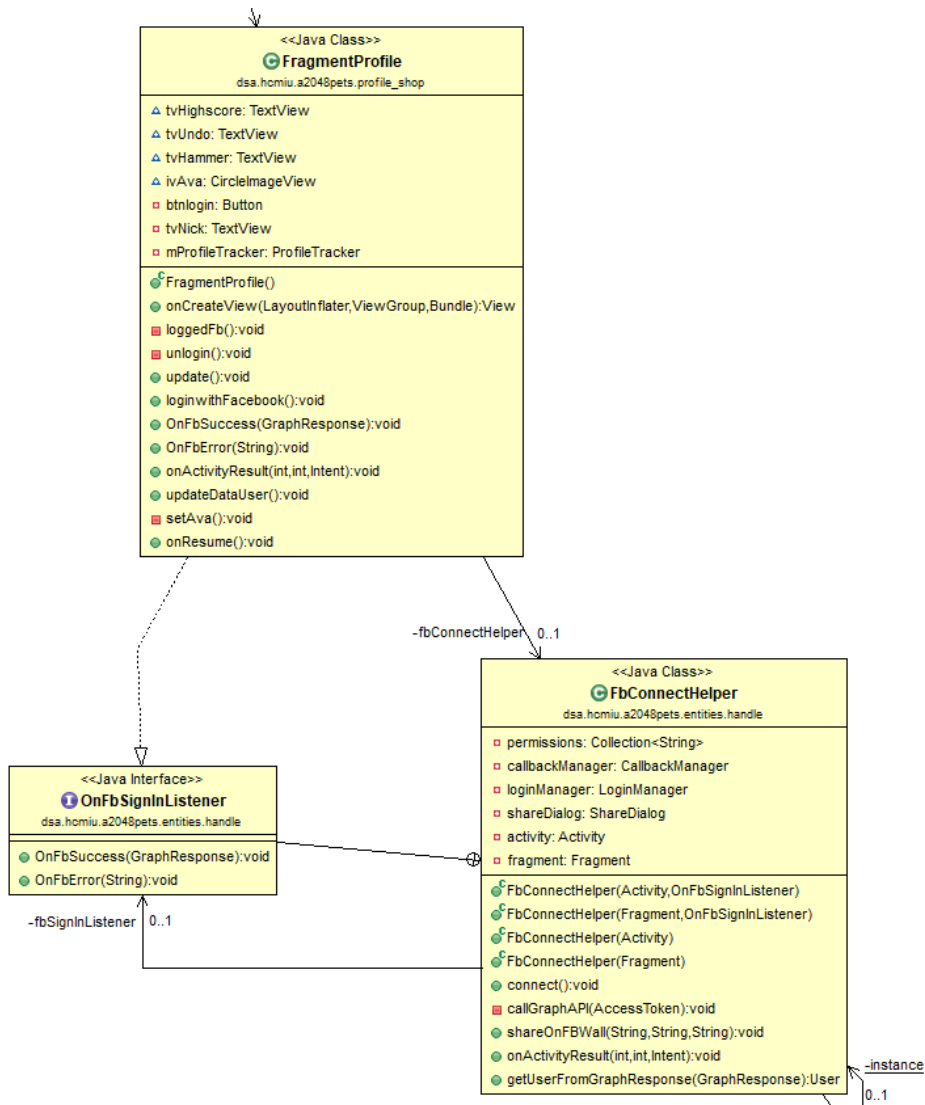
### Play process diagram



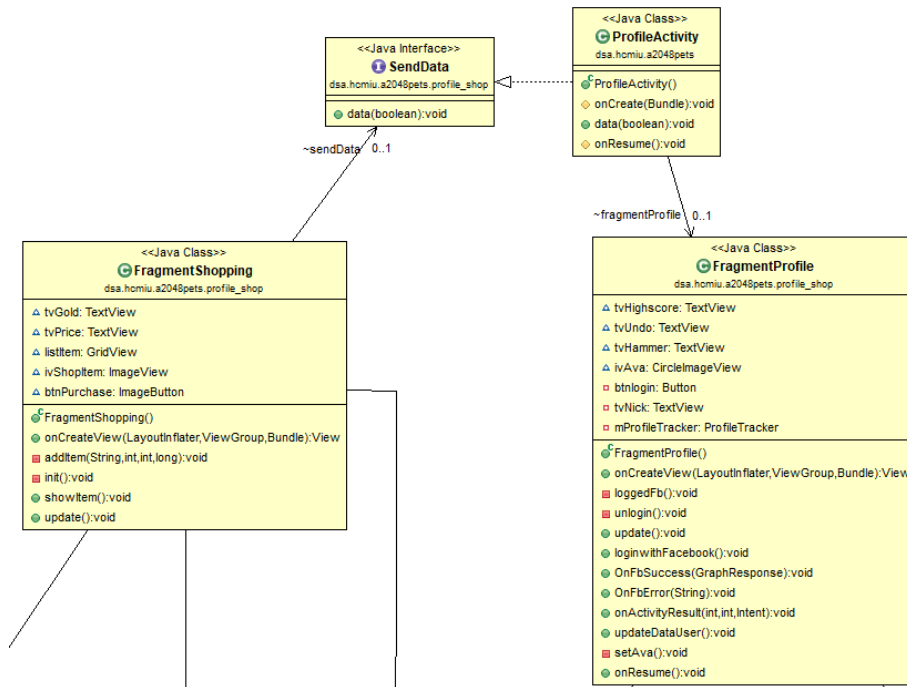
## Shop process diagram



## Profile diagram



## User Profile Activity

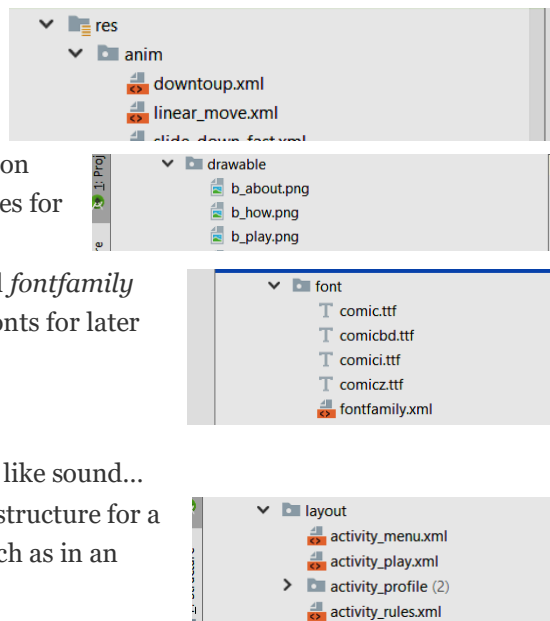


# CHAP 3: DESIGN & IMPLEMENTATION

## UI Design

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

- **anim:** xml files of animation: slide down-up, zoom in-out,...
- **drawable:** background, button images,.... – most of the images for UI
- **font:** Some fonts and file xml *fontfamily* which declares our addition fonts for later use.
- **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.



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

```
<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_parent"
        android:layout_height="0dp"
        android:src="@drawable/b_quit"
        android:background="@null"
        android:layout_weight="1" />

</LinearLayout>
```

Call a resource in drawable directory (\*)

Id to distinguish from other ones (\*\*)

(\*): 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`

In case of supporting many screen size, setting the fix height and width can not be flexible. To support as many devices as possible, android have Constraint layout library.

```
implementation 'com.android.support.constraint:constraint-layout:1.1.0'
```

Constraint layout let the user set up the layout by percentage. For example, if the width of the button takes 70% of screen:

```
<Button  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_constraintWidth_percent="0.7" />
```

Or creating a working area so items by guideline

```
<androidx.constraintlayout.widget.Guideline  
    android:id="@+id/glstartPvp"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="vertical"  
    app:layout_constraintGuide_percent="0.175"/>  
<androidx.constraintlayout.widget.Guideline  
    android:id="@+id/glendPvp"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="vertical"  
    app:layout_constraintGuide_percent="0.669"/>  
  
<androidx.constraintlayout.widget.Guideline  
    android:id="@+id/gltopPvp"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal"  
    app:layout_constraintGuide_percent="0.033"/>
```

Then use it to set inner item:

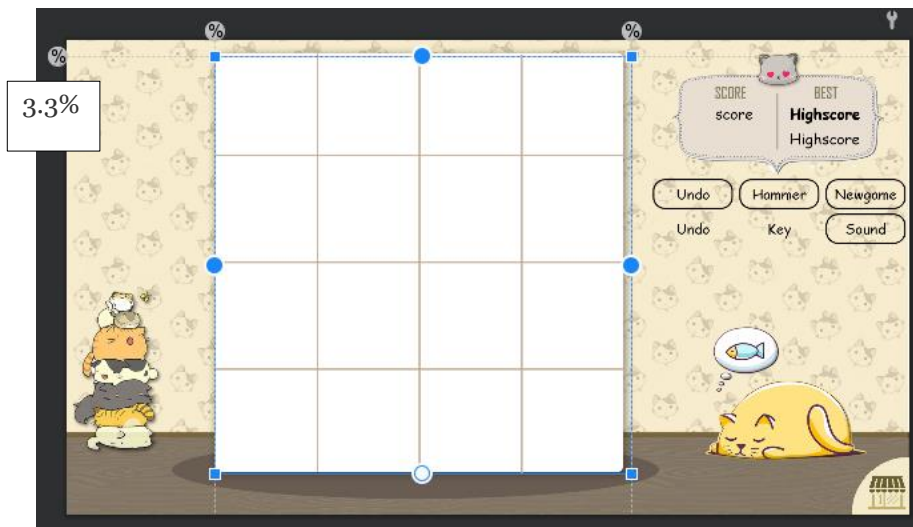
```

<GridView
    android:id="@+id/gvMatrixPvp"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:numColumns="4"
    android:paddingBottom="5dp"
    android:scrollbars="none"
    app:layout_constraintDimensionRatio="1:1"
    app:layout_constraintStart_toStartOf="@+id/glstartPvp"
    app:layout_constraintEnd_toEndOf="@+id/glendPvp"
    app:layout_constraintTop_toTopOf="@+id/gltopPvp"/>

```

17.5%

66.9%



- **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 FACEBOOK</string>`

- Access item in xml language:

43					android:text="SIGN IN WITH FACEBOOK"
43					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:

```
<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:

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

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

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

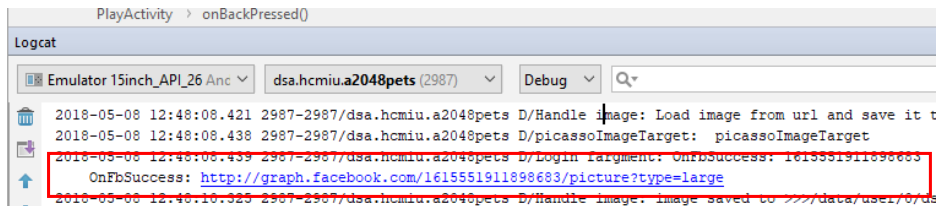
### 4. Debug

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



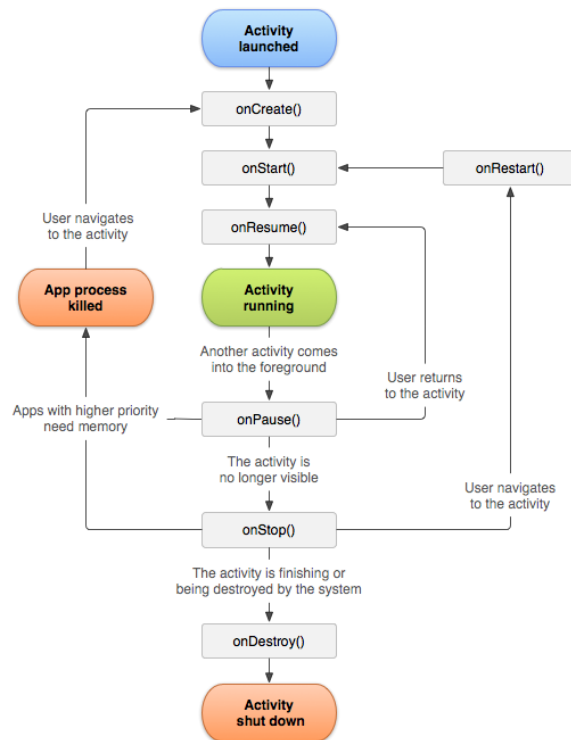
## 5. Activity

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.



*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 onBackPressed() {
    //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:

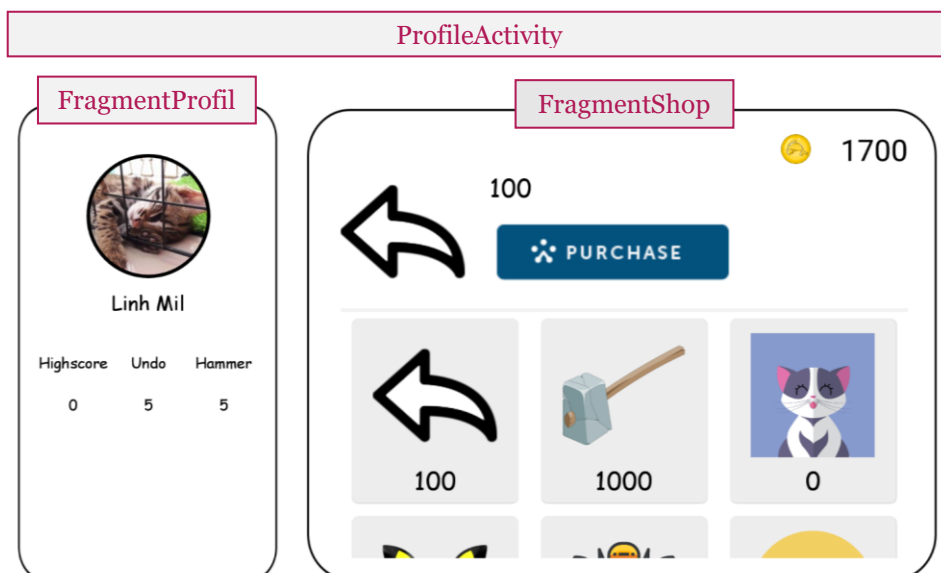
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="10">

    <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 create 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



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 directory  
mediaPlayer.start();  
mediaPlayer.setLooping(true); // loop the music infinitely
```

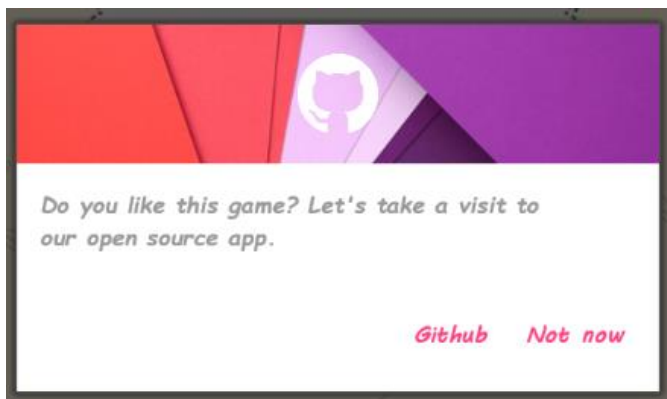
## 8. Dialog

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:



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:

- Cancele 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	<b>Array([])</b> In a JSON file , square bracket ([]) represents a JSON array
2	<b>Objects({})</b> In a JSON file, curly bracket ({} ) represents a JSON object
3	<b>Key</b> A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object
4	<b>Value</b> 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

2048Pets will store all the data to file as json. To write file in external storage, permission is needed in AndroidManifest.xml

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

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

Follow the instruction of Facebook developers (<https://developers.facebook.com/docs/facebook-login/android>) to declare the application on Facebook. Here are some steps need to be walk through before developing

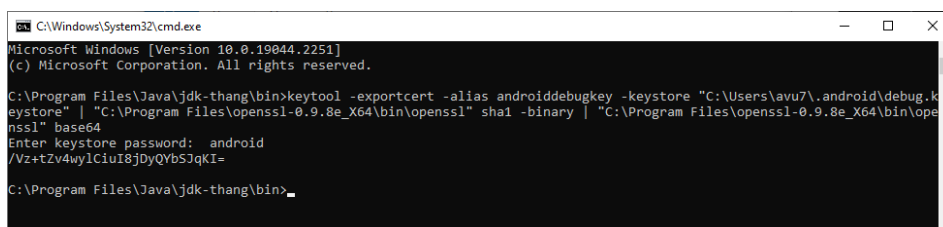
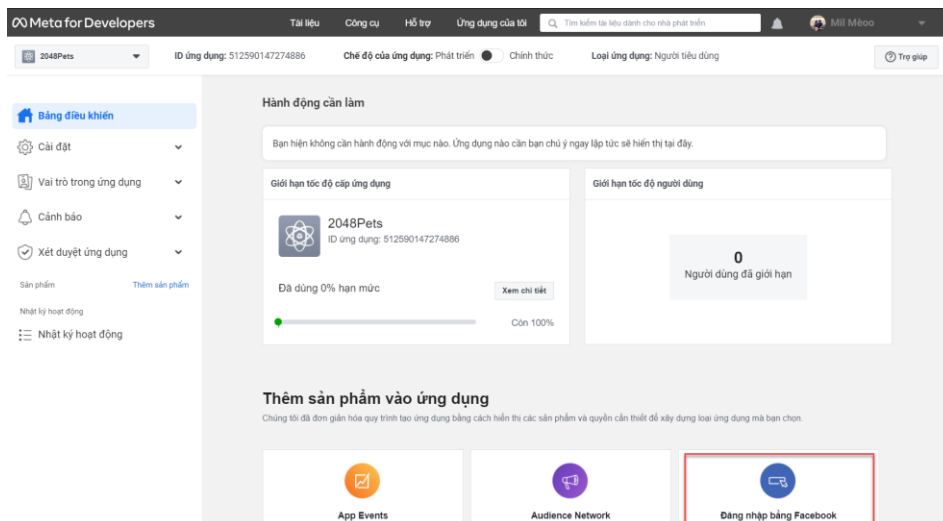
The image displays two sequential steps in the Facebook Developer portal for creating a new application.

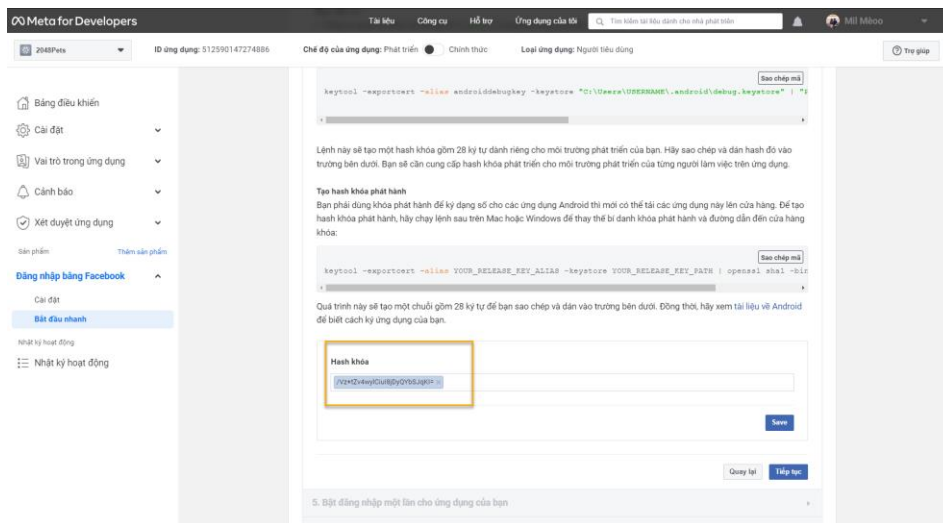
**Left Screenshot: Create new app**

- Loại (Type):** Selects "Chỉ kết" (Only connect).
- Chọn loại ứng dụng (Choose app type):** A list of categories with radio buttons:
  - Kinh doanh (Business):** Selects "Tôi hoặc quản lý các tài sản doanh nghiệp như Trang, Tài liệu, Nhóm, Quảng cáo, Messenger, WhatsApp và API để truy cập Instagram bằng các quyền, tính năng và sản phẩm hiện có dành cho doanh nghiệp." (I or manage business assets like Pages, Documents, Groups, Ads, Messenger, WhatsApp and APIs to access Instagram with permissions, features and products currently available for business).
  - Người tiêu dùng (Consumer):** Selects "Tôi chơi các trò chơi" (I play games).
  - Trò chơi tức thì (Instant Games):** Selects "Tôi chơi các trò chơi" (I play games).
  - Chơi game (Play Games):** Selects "Tôi chơi các trò chơi" (I play games).
  - Workplace:** Selects "Tôi chơi các trò chơi" (I play games).
  - Academic research:** Selects "Tôi chơi các trò chơi" (I play games).
  - Khác (Other):** Selects "Tôi chơi các trò chơi" (I play games).

**Right Screenshot: Provide basic app information**

- Trên ứng dụng (On app):** Selects "Chỉ kết" (Only connect).
- Cung cấp thông tin cơ bản (Provide basic app information):**
  - Tên ứng dụng (App name):** 2048Peta
  - Email liên hệ của ứng dụng (App contact email):** vulumpoang@gmail.com
  - Tại khoản kinh doanh (Business account):** Chọn loại tài khoản (Choose account type) - dropdown menu.
- Đăng cách tiếp tục (Continue):** Button to proceed.





FbConnectHelper class helps this app connect with facebook account

- 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. Google connect

Follow the instruction of [Google developers](#) to declare the application on Google. Here are some steps need to be walk through before developing action

#### Add Google Play service

In your project's top-level `build.gradle` file, ensure that Google's Maven repository is included:

```
allprojects {
    repositories {
        google()

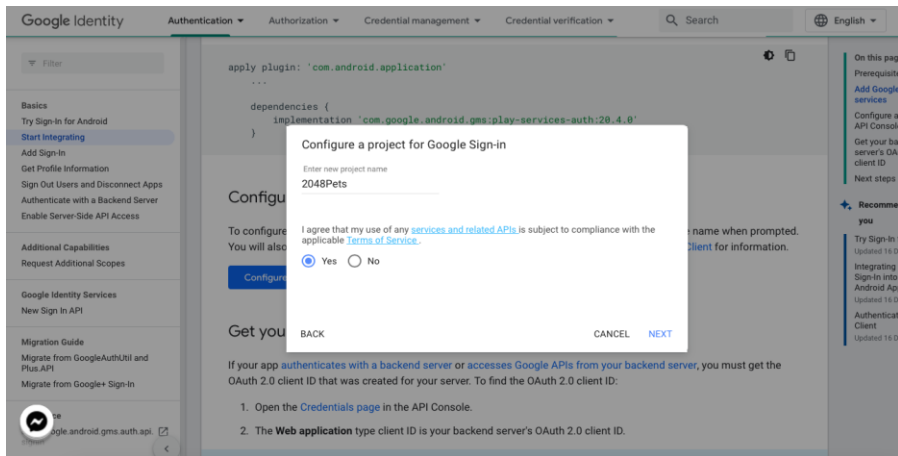
        // If you're using a version of Gradle lower than 4.1, you must instead use:
        // maven {
        //     url 'https://maven.google.com'
        // }
    }
}
```

Then, in your app-level `build.gradle` file, declare [Google Play services](#) as a dependency:

```
apply plugin: 'com.android.application'
...

dependencies {
    implementation 'com.google.android.gms:play-services-auth:20.4.0'
}
```

#### Configure Google API project at your android studio project



In sign-in activity's onCreate method, configure Google Sign-in to request user data required by your app

Create GoogleSignInOptions object with DEFAULT\_SIGN\_IN parameter.

```
// Configure sign-in to request the user's ID, email address, and basic
// profile. ID and basic profile are included in DEFAULT_SIGN_IN.
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
    .build();
```

In your sign-in activity's onCreate method, create a GoogleSignInClient object with the options you specified

```
// Build a GoogleSignInClient with the options specified by gso.
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

Check existing signed-In user

In onStart method

```
// Check for existing Google Sign In account, if the user is already signed in
// the GoogleSignInAccount will be non-null.
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);
updateUI(account);
```

If GoogleSignIn.getLastSignedInAccount return a GoogleSignInAccount object (rather than null), the user has already signed in to your app with Google.

If return null, the user has not yet signed in.

#### 4. 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));
}

```

## 5. 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);
```

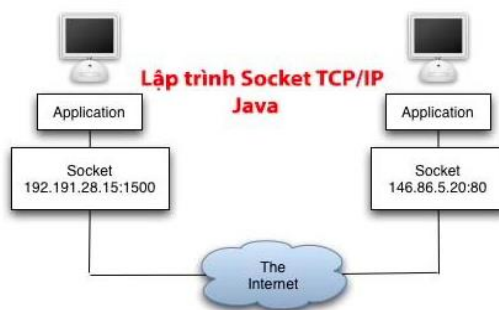
(create a new Timer task to run on UI thread)

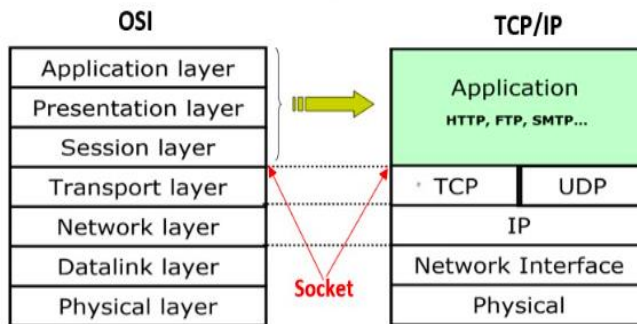
```
public void run() {  
    fadeInStep(deltaVolume);    //fade step  
    if(volume_fadein>1f){  
        timer.cancel();  
        timer.purge();  
    }  
}
```

(Cancel and purge the Timer)

## 6. JAVA socket TCP

Java provides a collection of classes and interfaces that take care of low-level communication details between the client and server.





The app has the following features:

- Sending the index number for each user joining to the room. (stop at 2 users in one room)
- Notifies when one player leaves.
- Sending score when player have new move.

Socket.IO provides an event-oriented API that works across all networks, devices and browsers. It's incredibly robust (works even behind corporate proxies!) and highly performant, which is very suitable for multiplayer games or realtime communication.

- Java Socket.IO client with dependency:

```
dependencies {
    implementation ('io.socket:socket.io-client:2.0.0') {
        exclude group: 'org.json', module: 'json'
    }
}
```

- Adding the internet permission to AndroidManifest.xml.

```

<!-- app/AndroidManifest.xml -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-permission android:name="android.permission.INTERNET" />
  ...
</manifest>

```

In this app, we prepared Server by Nodejs, and upload it on replit domain for global use:

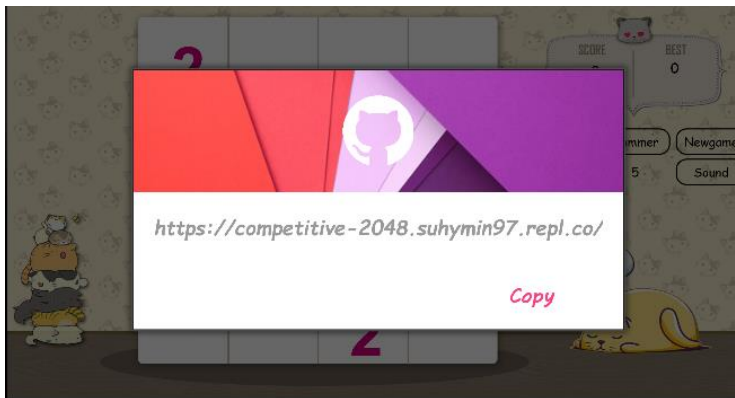
```

5  const app = express();
6  const server = http.createServer(app);
7  const io = socketio.listen(server); // Attach socket.io to our server
8
9  app.use(express.static('public')); // Serve our static assets from /public
10
11 server.listen(3000, () => console.log('server started'));
12
13 const connections = [null, null];
14
15 // Handle a socket connection request from web client
16 io.on('connection', function (socket) {
17
18   // Find an available player number
19   let playerIndex = -1;
20   for (const i in connections) {
21     if (connections[i] === null) {
22       playerIndex = i;
23     }
24   }
25
26   // Tell the connecting client what player number they are
27   socket.emit('player-number', playerIndex);
28
29   console.log(`Player ${playerIndex} has connected`);
30
31   // Ignore player 3
32   if (playerIndex == -1) return;
33
34   connections[playerIndex] = socket;
35
36   // Tell everyone else what player number just connected
37   socket.broadcast.emit('player-connect', playerIndex);
38

```

The 2048-server link will be <https://competitive-2048.suhymin97.repl.co>

User can copy and share it to his/her friend to start the competition



Firstly, Socket.IO is initialized with a new instance as follows:

```
import io.socket.client.IO;
import io.socket.client.Socket;

public class WelcomeActivity extends Activity {

    private Socket mSocket;
    {
        try {
            mSocket = IO.socket( uri: "https://competitive-2048.suhymin97.repl.co");
        } catch (URISyntaxException e) {}
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mSocket.connect();
    }
}
```

`IO.socket()` returns a socket for the URI with the default options. Notice that the method caches the result, so you can always get a same Socket instance for an url from any Activity or Fragment.

- Listening on event

Socket.IO is **bidirectional**, events are sent to the server, but also at any time during the communication the server can send events to client.

```

private void connect() {
    // mSocket.on(Socket.EVENT_DISCONNECT, onDisconnect);
    // mSocket.on(Socket.EVENT_CONNECT_ERROR, onConnectError);
    // mSocket.on(Socket.EVENT_CONNECT_TIMEOUT, onConnectError);
    // mSocket.on(Socket.EVENT_CONNECT, onConnect);
    mSocket.on(event: "player-number", playernumber);
    mSocket.connect();
}

```

Method for listening on the “player-number” event to receive messages from server.

```

private Emitter.Listener playernumber = new Emitter.Listener() {
    @Override
    public void call(Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String playerNumber = (String) args[0];
                if (playerNumber == "1") {
                    showMess("Waiting for player 2...");
                    mSocket.on(event: "player-connect", startGame);
                } else if (playerNumber == "-1") {
                    showMess("Game Full");
                } else { // Immediately start the game if we're player two
                    setUpGame();
                }
            }
        });
    }
};

```

Inside of `call()` is wrapped by `Activity#runOnUiThread()`, that is because the callback is always called on another thread from Android UI thread.

- Emitting Event

Same as Nodejs, Socket.io on Android also have “emit” function to submit a message and “on” function to listen. In this case, all the client will send to their current point to the

```
public void onSwipeUp() {  
    HandleGame.getInstance().moveUp();  
    mSocket.emit( event: "actuate", HandleGame.curBoard.getScoreBoard());  
    check();  
}
```

- Managing Socket state

Since an Android Activity has its own lifecycle, we should carefully manage the state of the socket also to avoid problems like memory leaks. In this app, we'll close the socket connection and remove all listeners on *onDestroy* callback of Activity.

```
@Override  
public void onDestroy() {  
    super.onDestroy();  
    mSocket.disconnect();  
    mSocket.off("new message", onNewMessage);  
}
```

Calling *off()* removes the listener of the new message event.

## 7. 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 vào
                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](http://develop.android.com))

Therefore, we know that variable “event” have a function to describe the position as axis values, because we are using the 2<sup>nd</sup> 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  $x_1$  and  $x_2$ , as  $x_1$  is the first position and  $x_2$  is the last position. We know if  $x_2 > x_1$  then move right,  $x_2 < x_1$  then move left.

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

Similarly, when  $(y_2 - y_1) > (x_2 - x_1)$  we know the mouse is moving vertically, then we compare  $y_2$  and  $y_1$  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;
```

## 8. 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;
}
```

## 9. 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.

1	2	1											1
↑	↑	2			2	2	2			2	2	4	
2					4		+			4			
4	2	2					2					2	
		2											
2	3	1	4		2	3	1	4		2	3	2	4

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

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

```

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.

## 10. 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;
}

```

countEmpty is used to count the empty tile

- **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>
</array>

```

This is the address of picture resource

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.

## 11. 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*.

## 12. 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.

## 13. Save data to local

HandleFile.java is used to help read and write file.

# CHAP 4:

## FINAL APP GAME

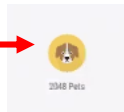
Source code (link github):

<https://github.com/suhymin97/MobileProj-2048Pets>

### Instruction

#### 1. Begin the game:

Click on the icon to open the game.



Loading screen will appear.

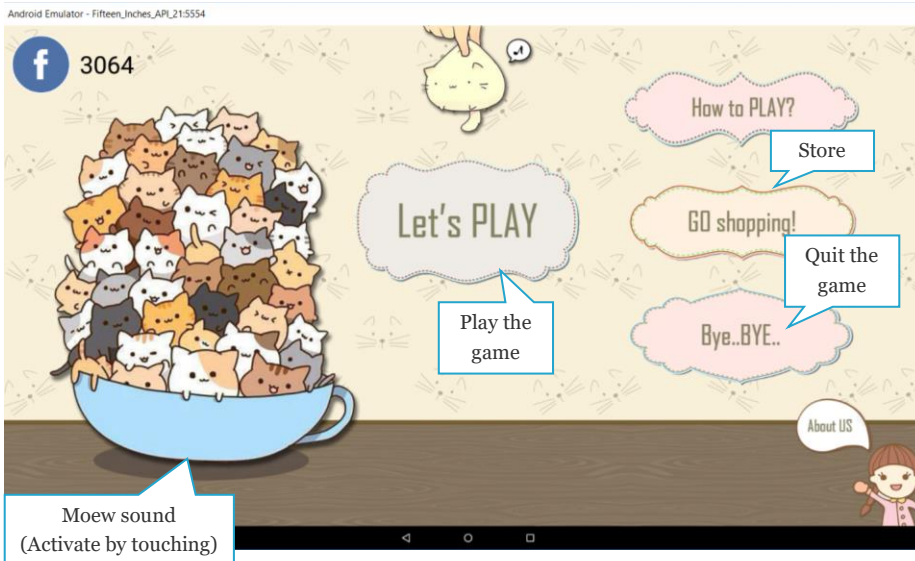


Click on the backspace to enter the "Main menu"

#### 2. Main Menu:

Facebook  
(hasn't logged in yet)

Rule of  
the game

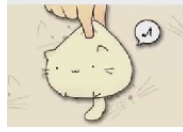


We add the animation and sounds for the “cup of cats” and the “sound cat” :

- The cup of cats moves “y-axis” both sides:

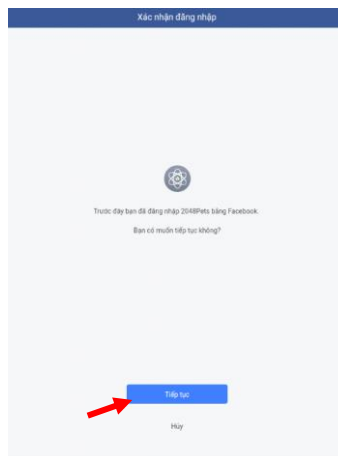
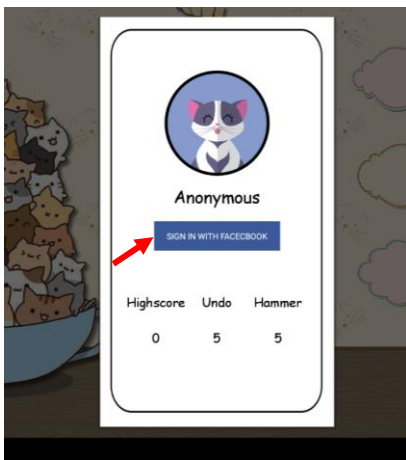


- When you move the mouse the “cup of cats”, the sound “meow” is ON with the **fade-in** effect, and it is OFF with the **fade-out** effect when your mouse is out of that.
  - The sound cat moves “y-axis” both sides:

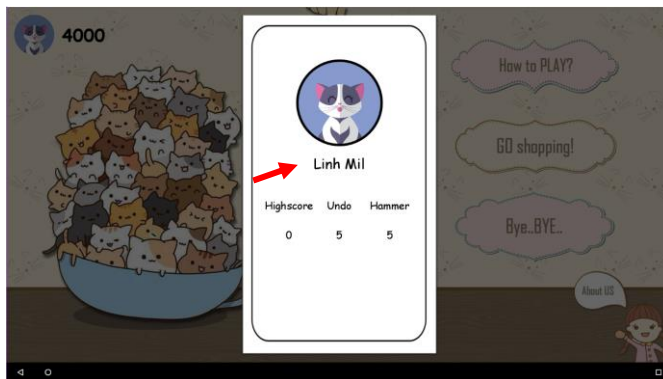


### 3. Facebook:

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

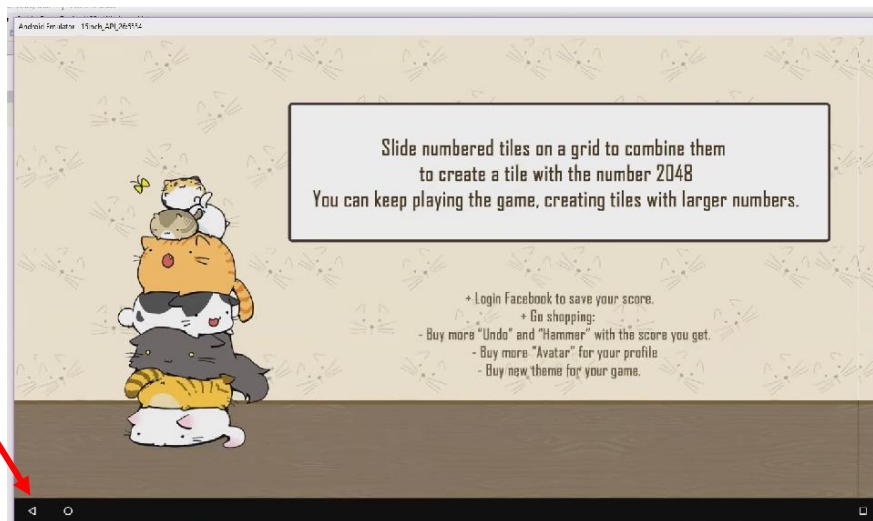


Now, you’ve already logged in your Facebook account to save your data of the game.



#### 4. How to play:

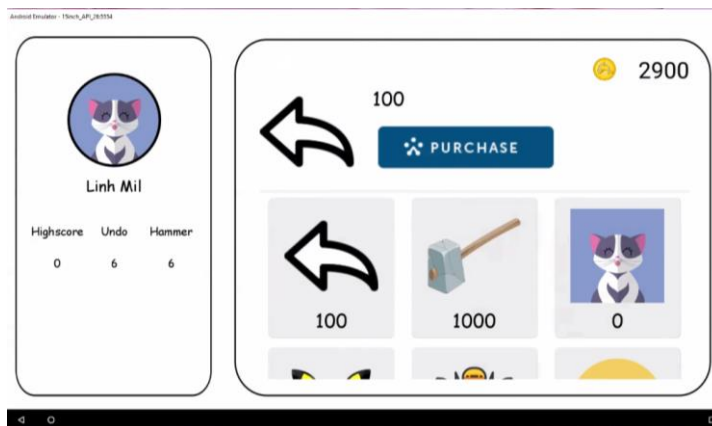
Show user the rule of the game:



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.

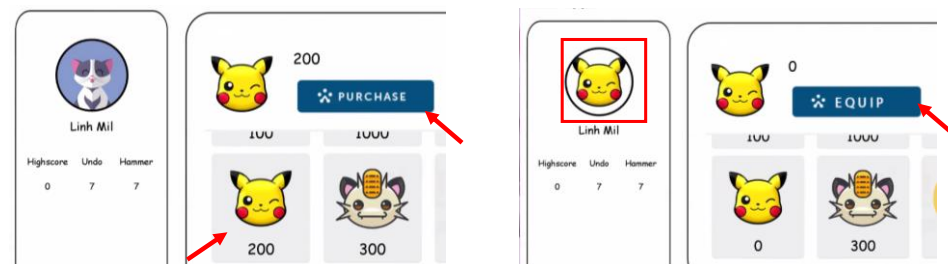


Use the money which is the score you gained from the game to buy the stuffs. Click on the stuff you want to buy and click PURCHASE. For example:

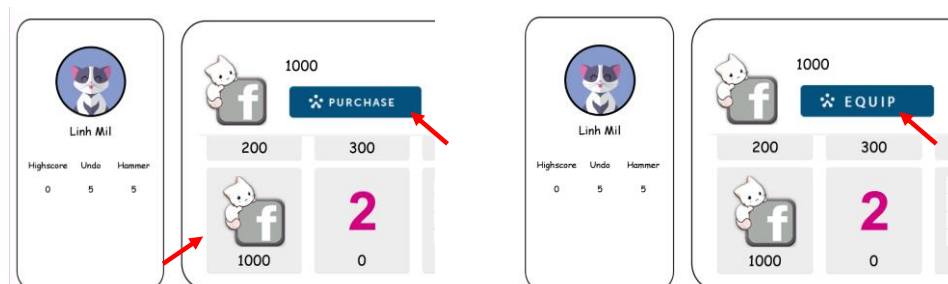
- Buy 1 Undo and 1 Hammer.

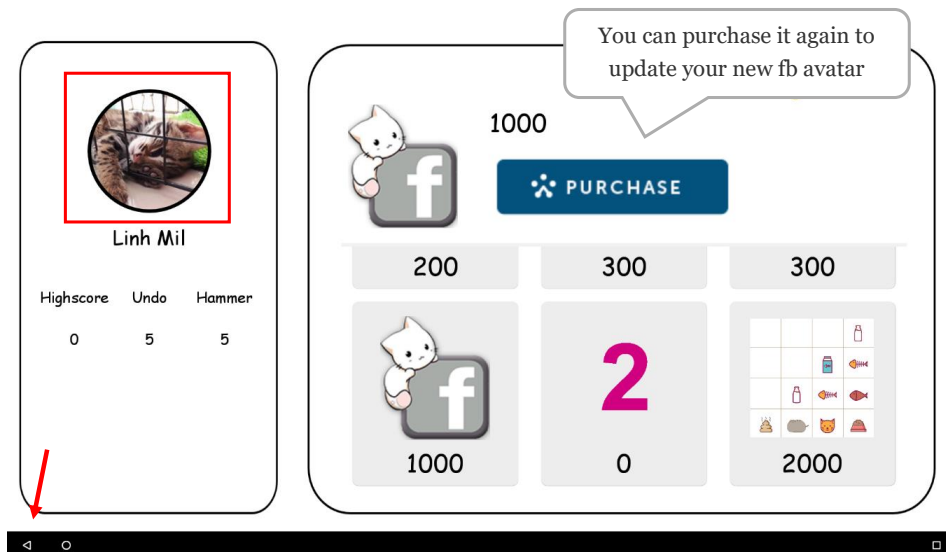


- Buy a new avatar for your profile and then apply it by clicking EQUIP



- Or change game avatar to facebook avatar:

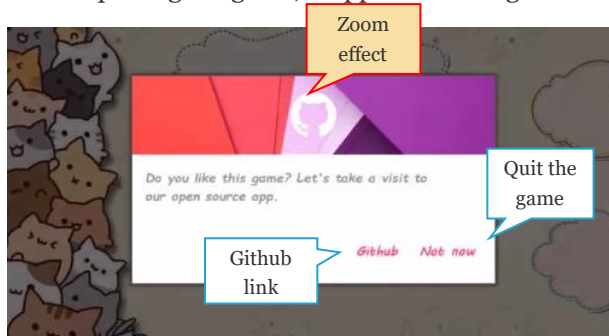




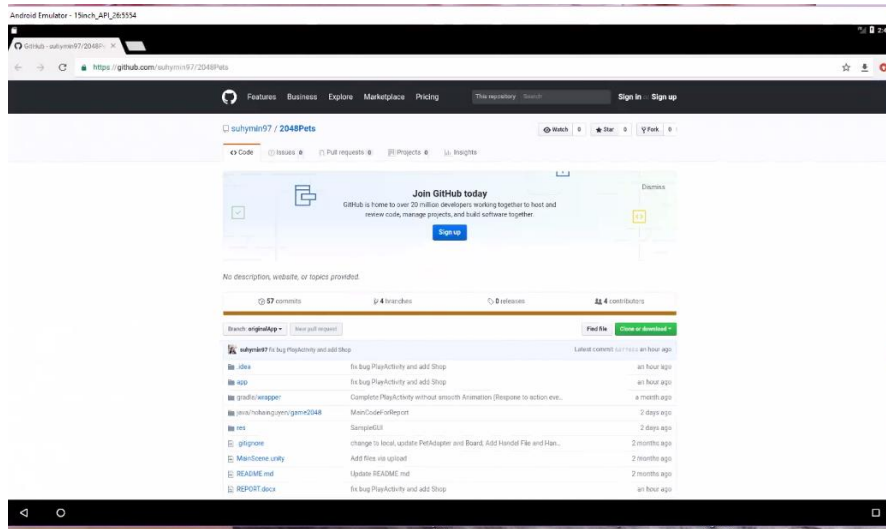
Click on the backspace to get back to the Previous screen.

## 6. Bye.. bye – Backspace button:

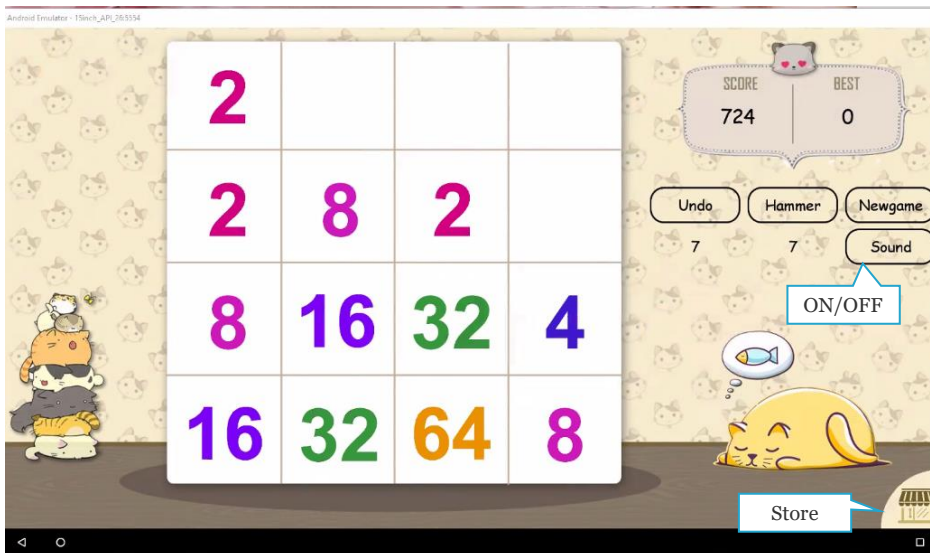
Before quitting the game, it appears a dialogue:



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

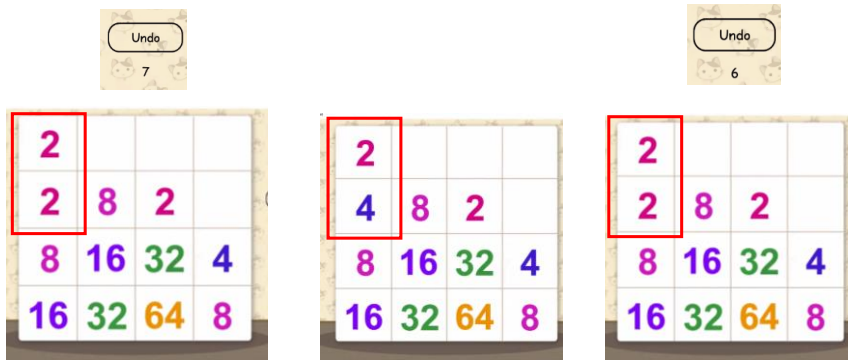


## 7. Let's play:

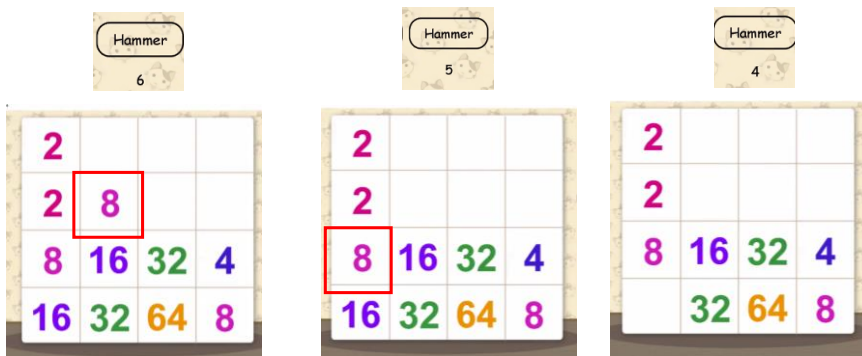




- **UNDO:** Back to previous step



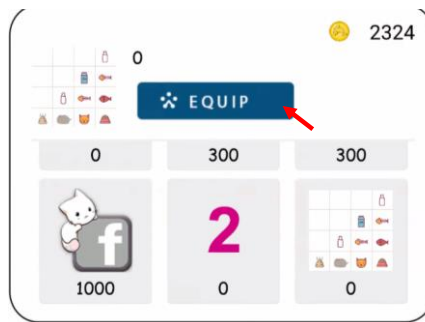
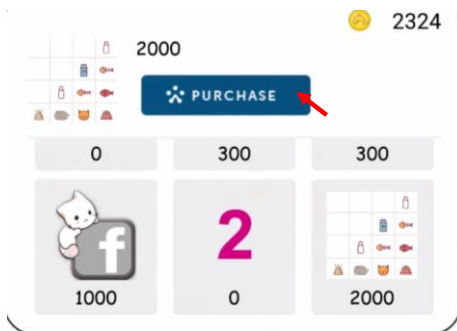
- **HAMMER:** randomly delete 1 element



- **NEWGAME:**



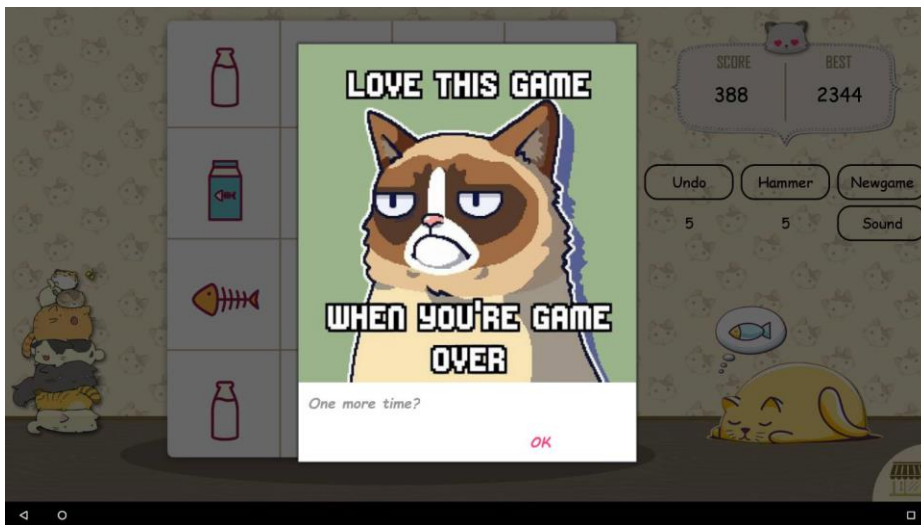
Let's go to the store to buy a new theme and apply it:



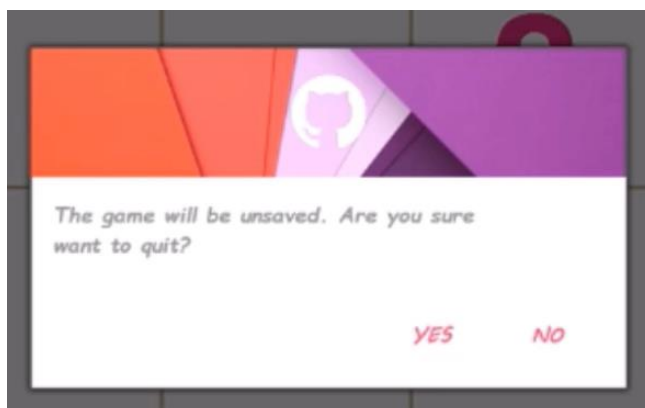
And enjoy your new theme:



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



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:



# 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 -