|  |  |
| --- | --- |
| C:\Users\Admin\Desktop\cover.png  2048 Pets  final project DSA  *HomiesHCMIU* Android Game 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 friendsGame developer Vu Minh Tu Anh  Luu Tuan Hung  Bui Ngoc Thanh Hien  Ho Hai Nguyen ADvisor Prof. Tran Thanh Tung |

Contents

[CHAP 1: INTRODUCTION 3](#_Toc513630944)

[1.1 Gaming in the Field: 3](#_Toc513630945)

[1.2 About the game project: 4](#_Toc513630946)

[1.3 Our 2048 Pets game: 4](#_Toc513630947)

[CHAP 2: SOFTWARE REQUIREMENTS 5](#_Toc513630948)

[2.1 What we have: 5](#_Toc513630949)

[2.2 What we want: 5](#_Toc513630950)

[2.3 Working tools, platform: 6](#_Toc513630951)

[2.4 Use Case Scenario 6](#_Toc513630952)

[2.5 Use Case diagram 7](#_Toc513630953)

[CHAP 3: DESIGN & IMPLEMENTATION 8](#_Toc513630954)

[Package Diagram 8](#_Toc513630955)

[UI Design 8](#_Toc513630956)

[Basic Android 10](#_Toc513630957)

[1. Application 10](#_Toc513630958)

[2. Palette attributes: 11](#_Toc513630959)

[3. Map variables: 11](#_Toc513630960)

[4. Debug 12](#_Toc513630961)

[5. Activity 12](#_Toc513630962)

[6. Fragment 16](#_Toc513630963)

[7. Sound & Animation Effect 18](#_Toc513630964)

[8. Dialog 19](#_Toc513630965)

[Main methods 20](#_Toc513630966)

[1. Handle file 20](#_Toc513630967)

[2. Facebook Connect: 22](#_Toc513630968)

[3. Handle image 25](#_Toc513630969)

[4. Handle sound 26](#_Toc513630970)

[5. OnSwipeTouchListener (implements OntouchListener) 26](#_Toc513630971)

[6. Adapter 30](#_Toc513630972)

CHAP 1: INTRODUCTION

#### 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***…)

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, and so on.

#### 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

#### 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:
  + Buy more “Undo” and “Hammer” with the score you get.
  + Buy more “Avatar” for your profile
* Add more sounds and animation to make the game more attractive.

CHAP 2:   
SOFTWARE REQUIREMENTS

#### 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.

#### 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.

#### Working tools, platform:

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

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

* 1. Circle Image View

implementation 'de.hdodenhof:circleimageview:2.2.0'

* 1. Picasso

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

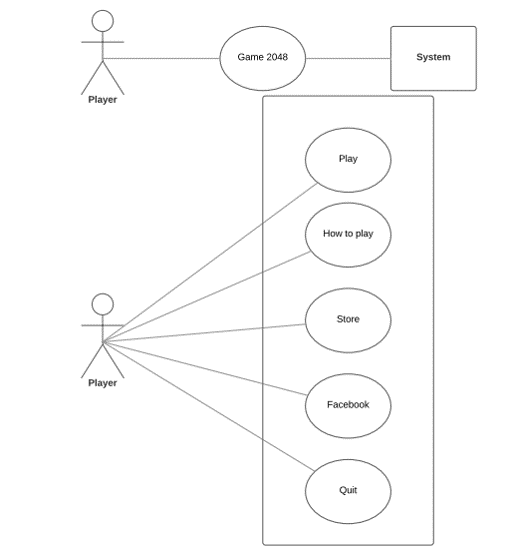
1. Photoshop

#### 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 |

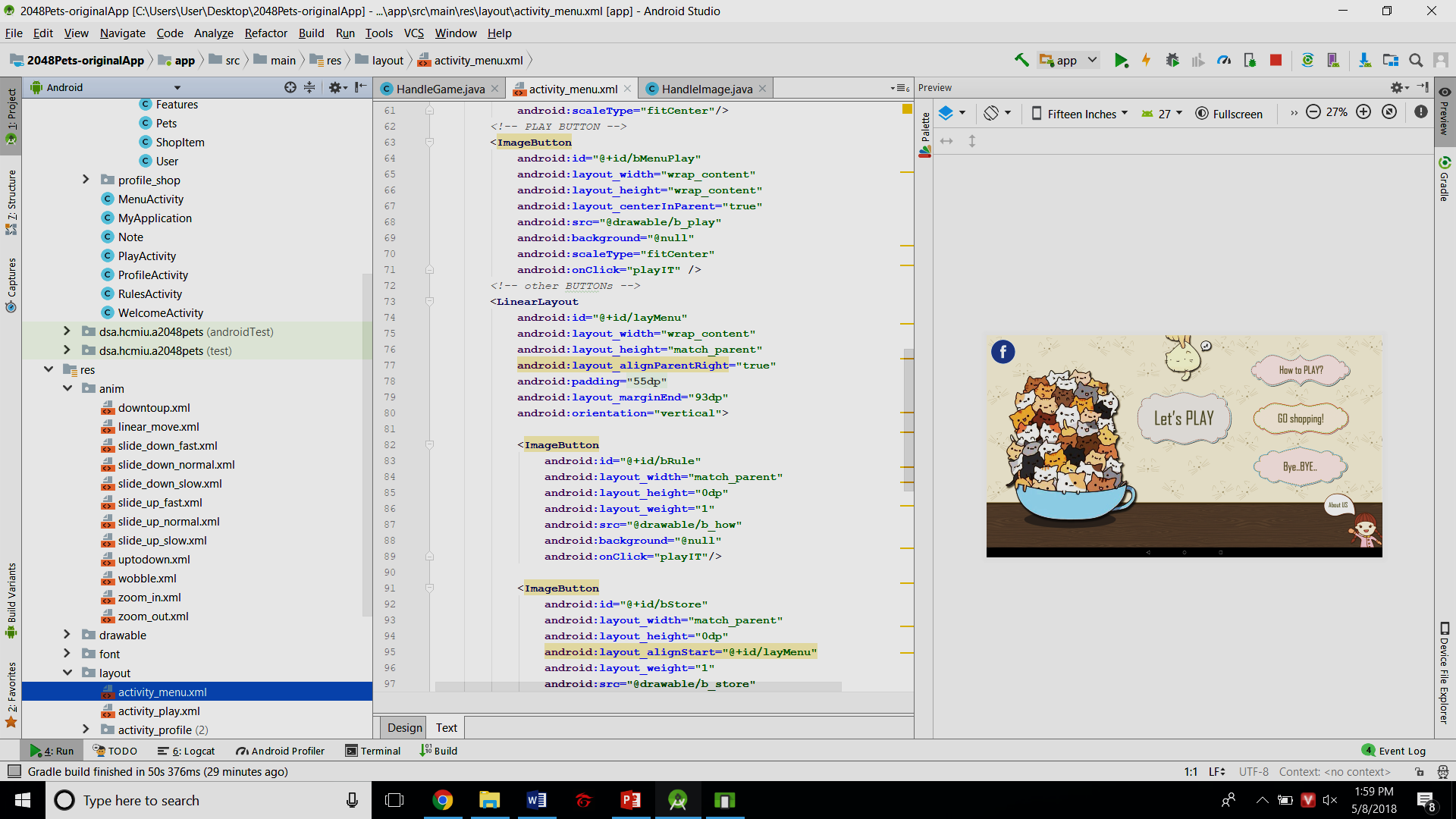
#### Use Case diagram

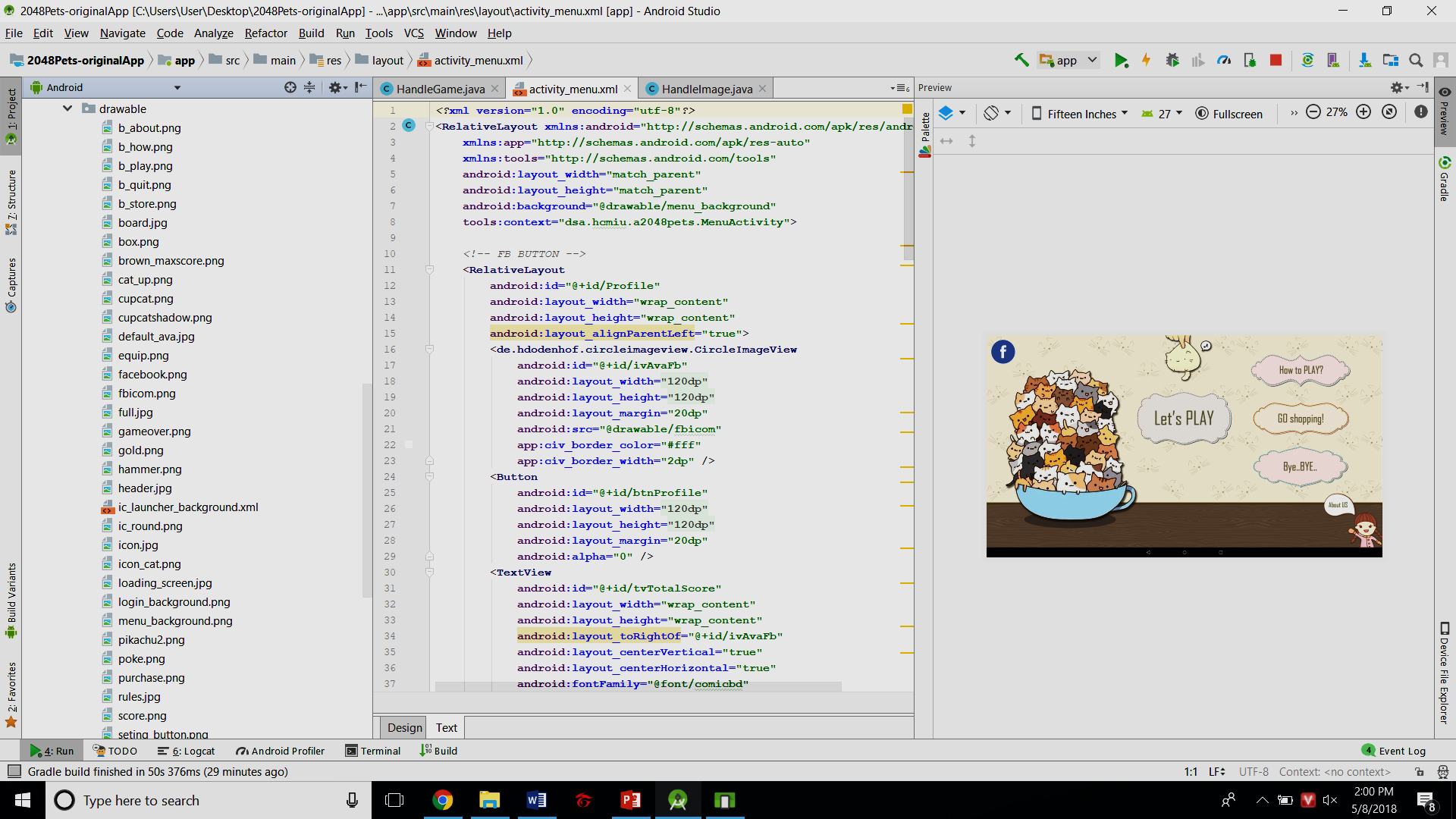
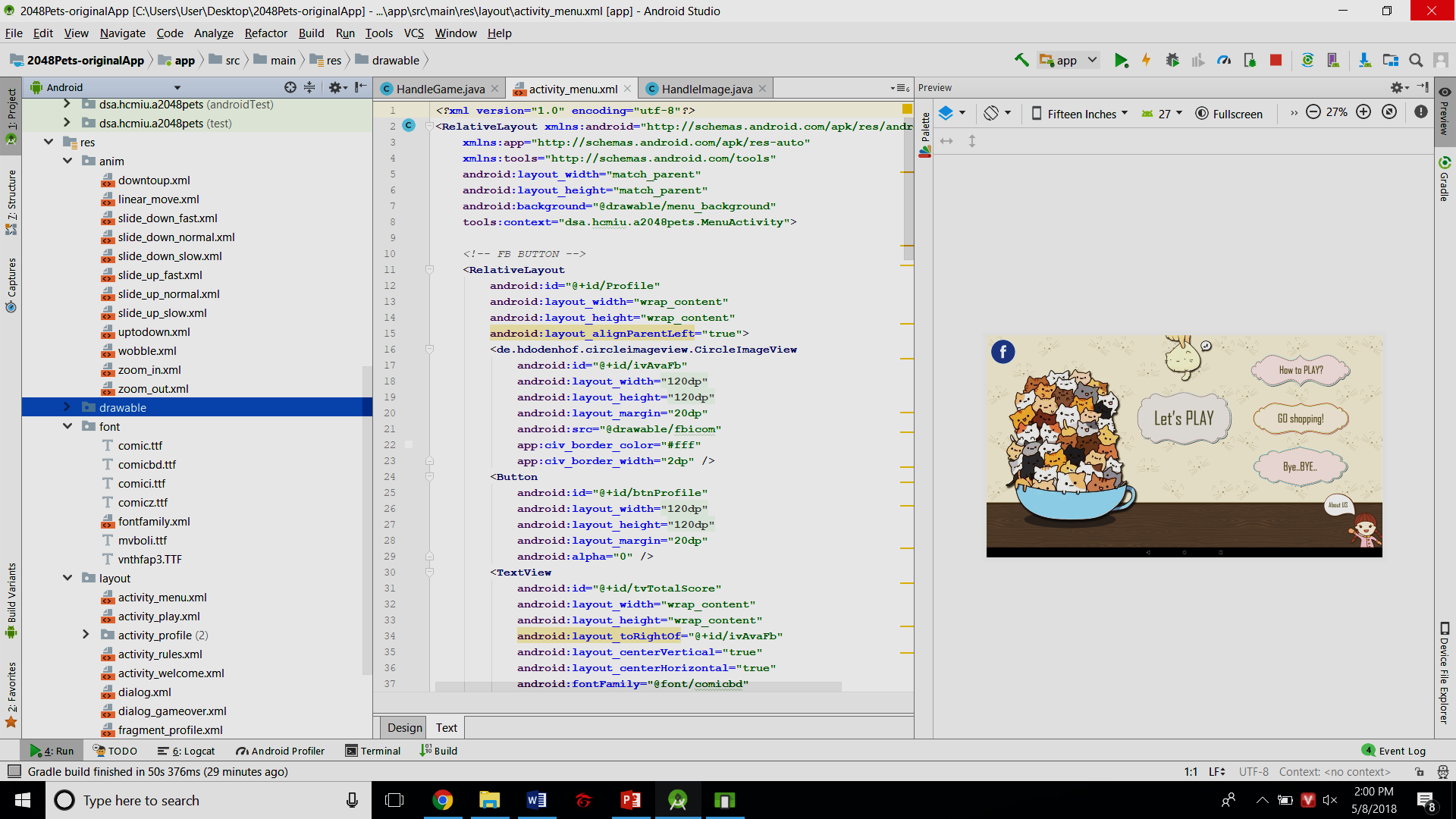
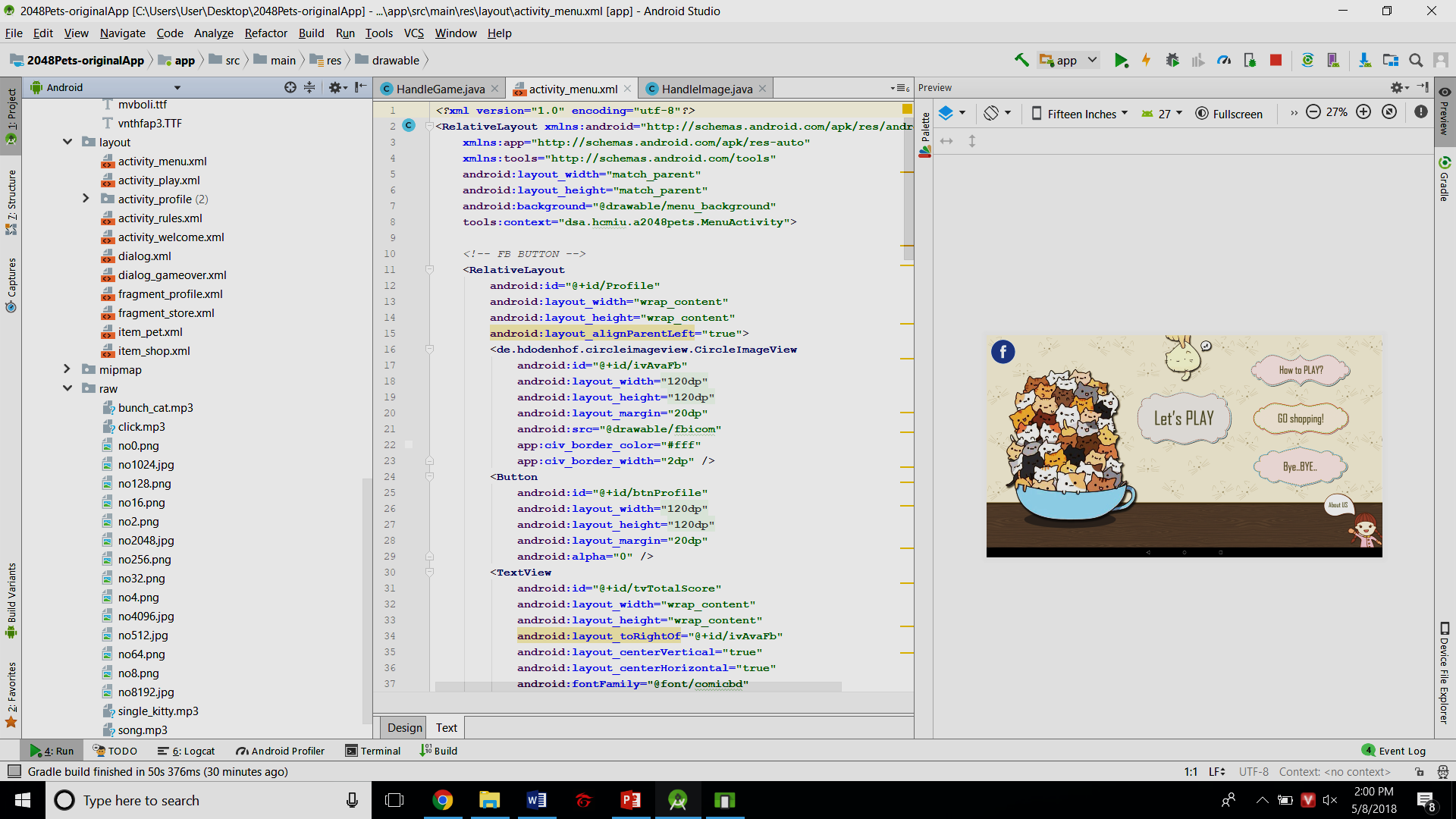


CHAP 3:  
DESIGN & IMPLEMENTATION

#### 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,…
* **drawable:** background, button images,.... – most of the images for UI
* **font:** Some fonts and file xml *fontfamily* which declairs 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



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

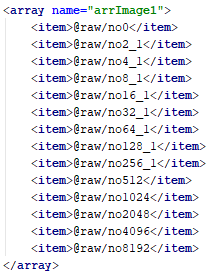
* **values:** Some default xml files for assigning color html code, dimension values, strings / string arrays, style layout into an id. For example with our strings.xml:

*single string:* 

* Access item in xml language:



* Access item in java language: R.string.facebook

*array string:*

* Access item in java language:

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

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));

}

#### Basic Android

##### Application

Exit App:

System.exit(0);

Open an url:

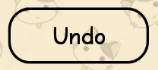
startActivity(new Intent(Intent.ACTION\_VIEW, Uri.parse(String url)));

##### 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

##### Map variables:

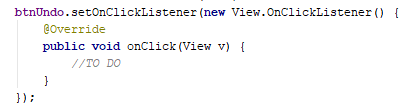
When we have a button on xml, and we want to manage it on java class, we need to map them. For example, our project has a button like this:



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



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



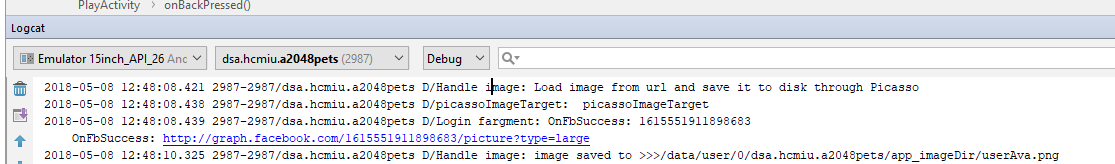
##### Debug

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:





##### 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 >

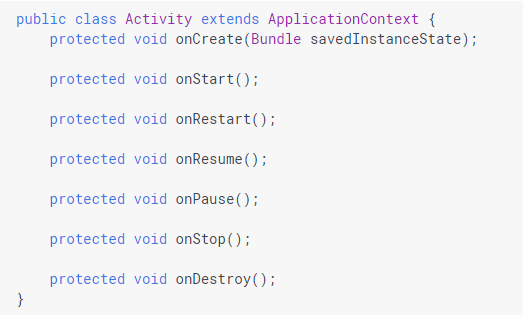
The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.

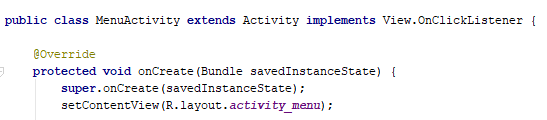


*State diagram for an Android Activity Lifecycle.*

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.



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

setContentView(int layout)

* Start another activity:

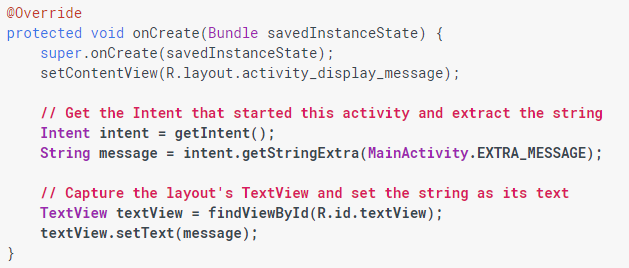
An Intent is an object that provides runtime binding between separate components, such as two activities. The Intent represents an app’s "intent to do something." You can use intents for a wide variety of tasks, but in this lesson, your intent starts another activity.



* Transfer data between activities: add the EXTRA\_MESSAGE constant and the sendMessage() code, as shown here:



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



* Handle other button of your mobile/tablet

@Override

public void onBackPress() {

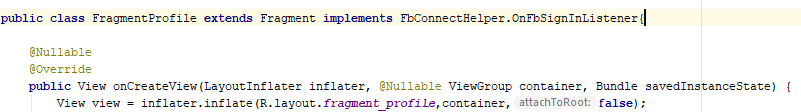
//TO DO

}

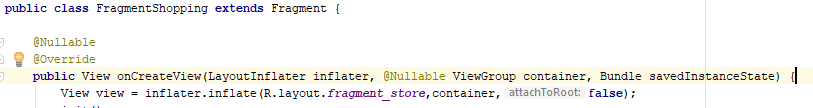
##### Fragment

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*



* *FragmentShopping* connects with layout *fragment\_store*

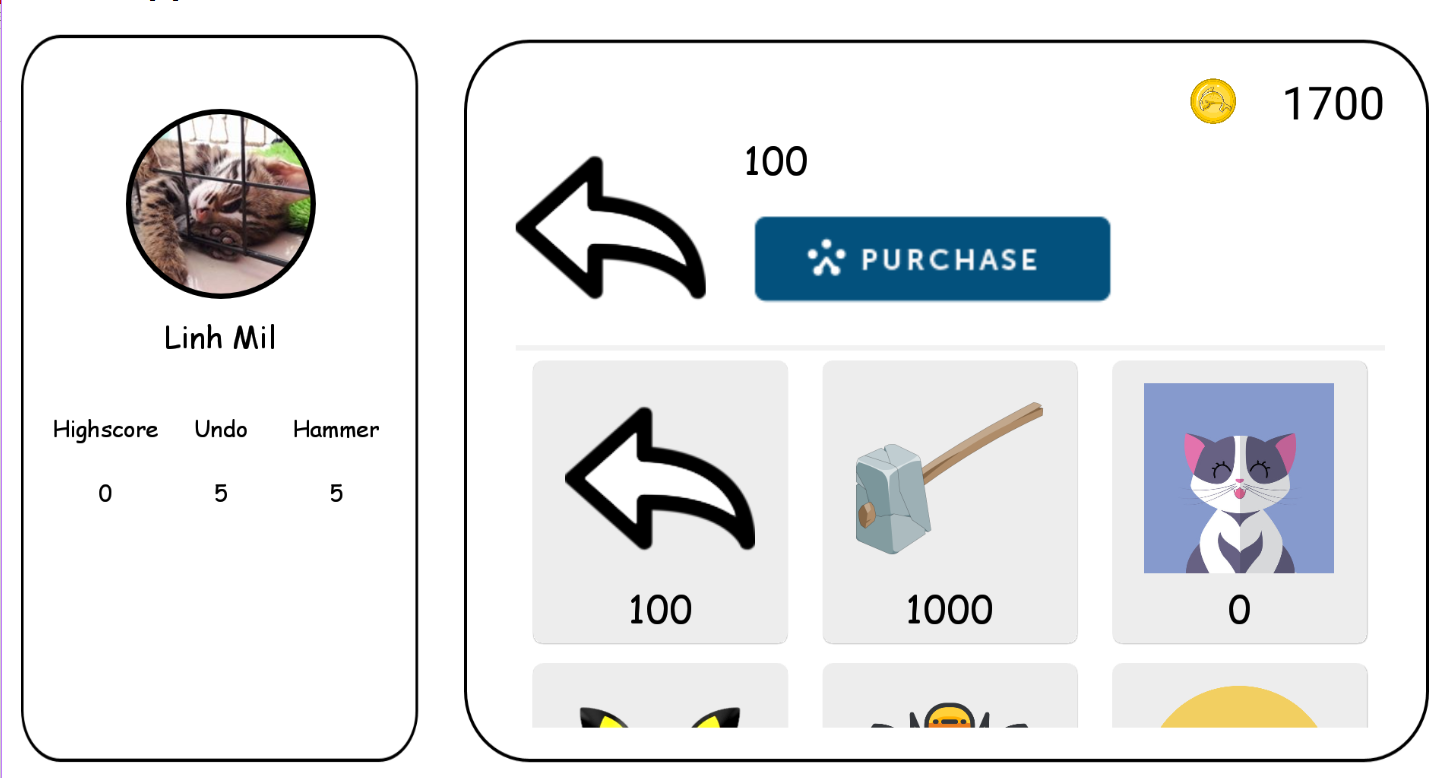


* Then add those 2 fragments to file xml which is connected with main activity:



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

ProfileActivity



FragmentShop

FragmentProfile

##### 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(),

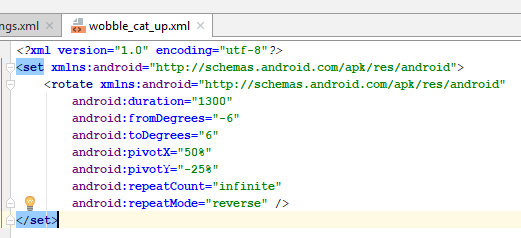
int idAnimationResources);

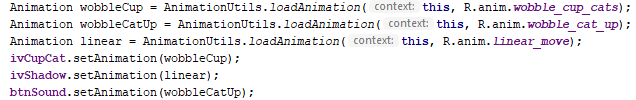
Then assign that animation to the attribute that we want:

source.setAnimation(animation);

For example:

wobble\_cat\_up.xml is stored in anim directory



Set animation to item

* **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

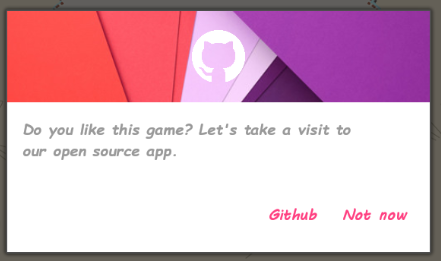
##### Dialog

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.

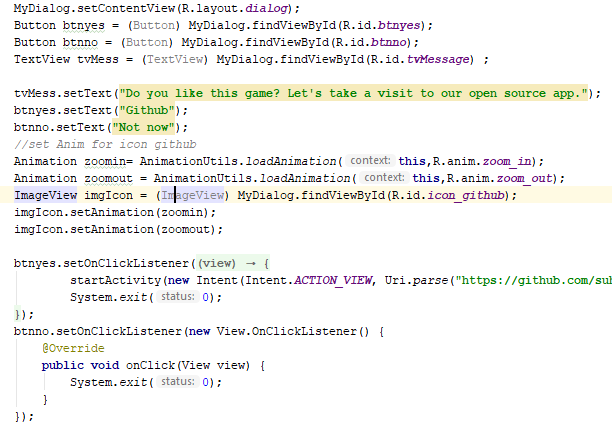
But in this project we only use, our custom Dialog. This is one of them:



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.



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



Finally, to make this dialog appear on screen we use method: 

Dialog gives us some more properties:

* Cancle dialog: Cancle()
* Unable touch outside the dialog: setCanceledOnTouchOutside(false)

#### Main methods

##### 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 |

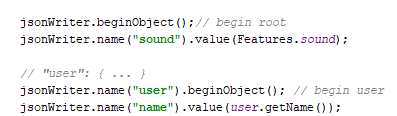
The method getJSONObject returns the JSON object. The method getString returns the string value of the specified key.

|  |
| --- |
| **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 |

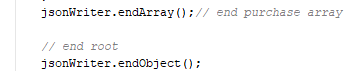
For example:

* Create JSON:

The root object which contains those variables

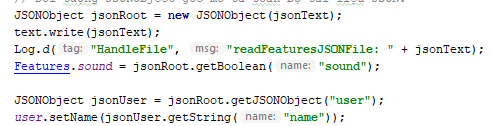


A child node of the above root



* Read JSON:





##### Facebook Connect:

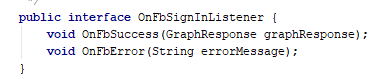
FbConnectHelper class helps this app connect with facebook account, follow the instruction of Facebook developers (<https://developers.facebook.com/docs/facebook-login/android>)

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

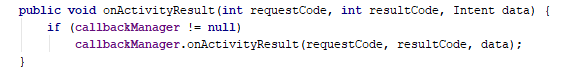
<uses-permission android:name="android.permission.INTERNET"/>

* 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

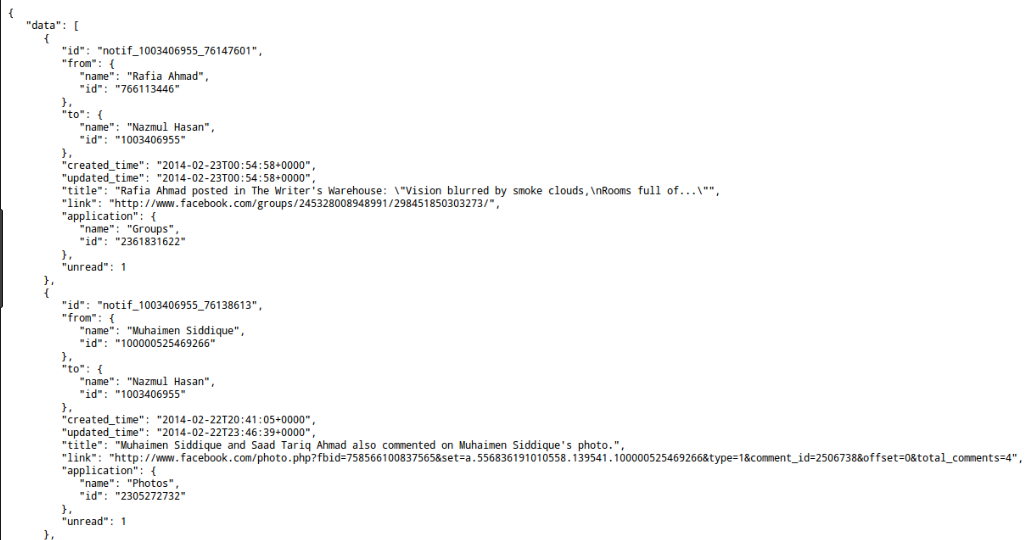




* call callbackManager.onActivityResult to pass the login results to the LoginManager via callbackManager.



* Then Facebook will response us by JSON file like this



After that, we can analyze that file like Handle File

Use case of *FbConnectHelper*:

implements FbConnectHelper.OnFbSignInListener

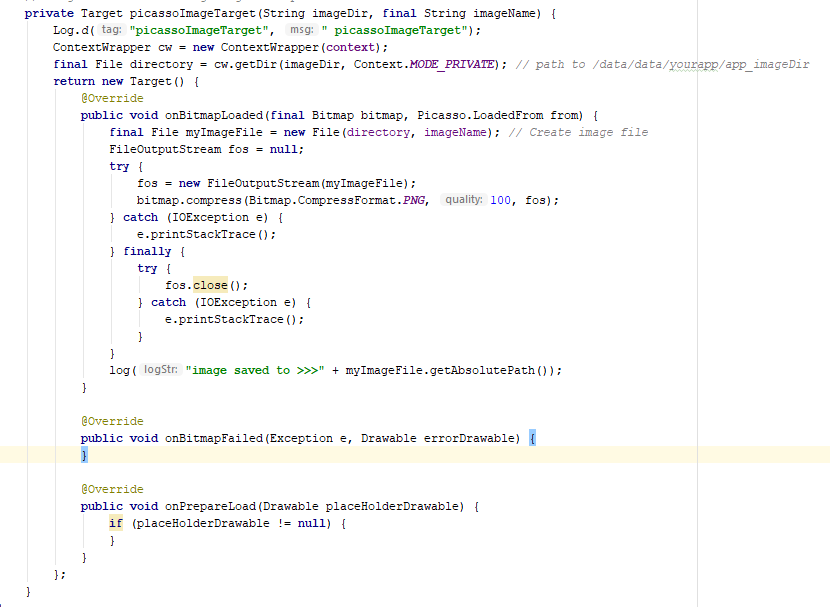


When we log in successfully and facebook feedback a JSON file

##### 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.



Create and write file mode:

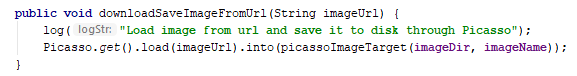
* MODE\_PRIVATE: overwrite if the file is existed)
* MODE\_APPEND: if file is exist, continue writing below.

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:



##### 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();

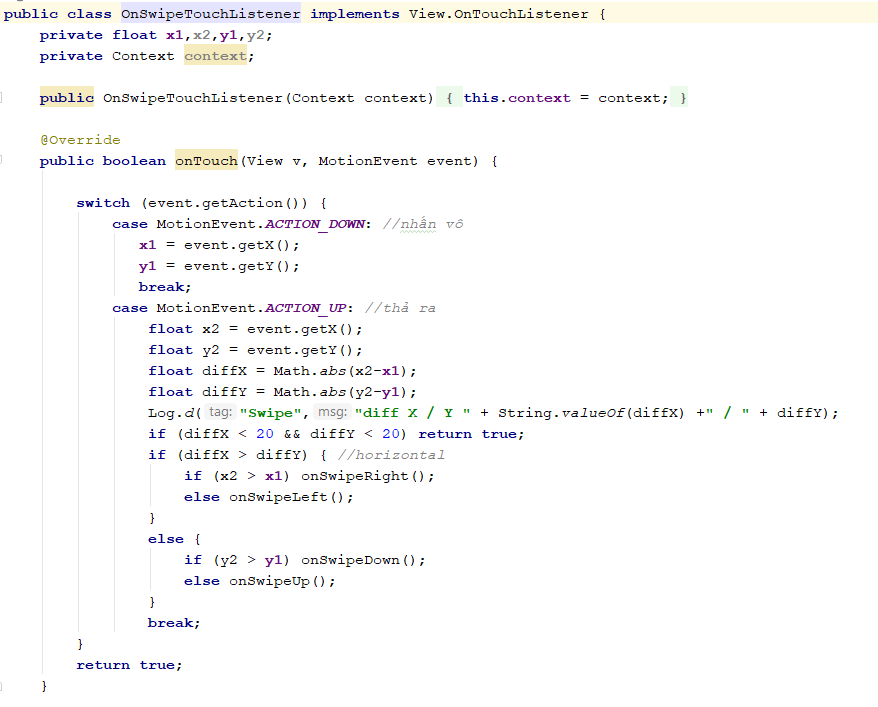
}

}

(Cancel and purge the Timer)

##### 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.



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:



*We define “event” as a variable of “Motion 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)***

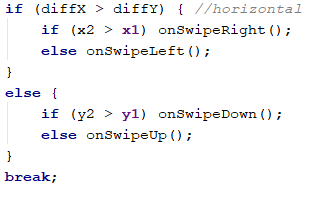
*Therefore, we know that variable “event” have a function to describe the position as axis values, because we are using the 2nd dimension array so X-axis and Y-axis values are all we need.*

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





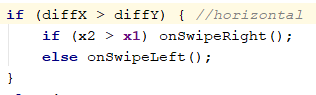
Next, there are 2 variables called “diffX” and “diffY”. Those variable store results which are used for identify kind of motions (left, right, up, down).



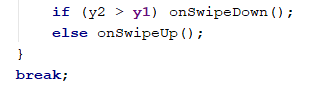
When the mouse move horizontally, no matter left or right the absolute value of “x2-x1” will be greater than “y2-y1”(equals to zero). Then we have



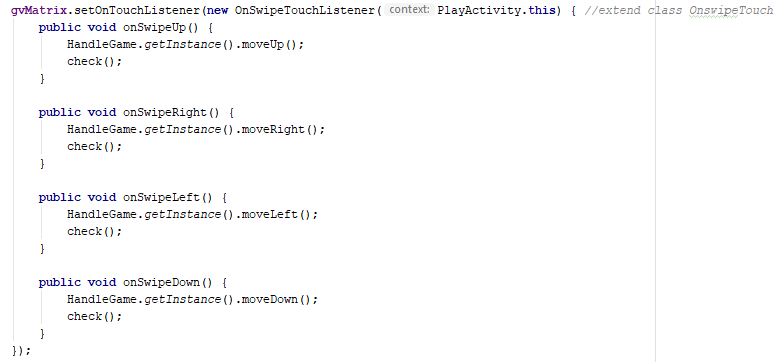
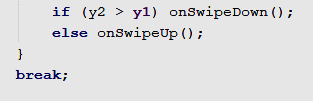
*After that we compare x1 and x2, as x1 is the first position and x2 is the last position. We know if x2 > x1 then move right, x2<x1 then move left.*



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



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



##### 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](http://abhiandroid.com/ui/xml) file) for list items in which you want to show as an item.

* [**objects**](http://abhiandroid.com/java/class-objects)**:**

The fourth parameter is an array of objects, used to set the array of elements. We can set the [object](http://abhiandroid.com/java/class-objects) of array or [array list](http://abhiandroid.com/java/arraylist) 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());

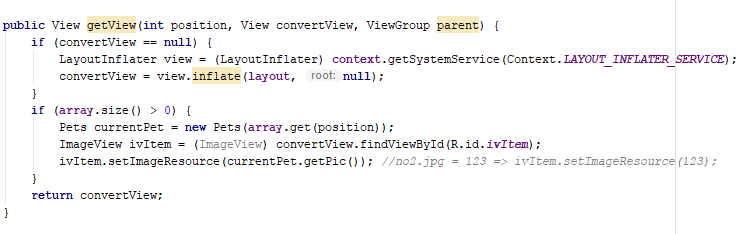
gvMatrix.setAdapter(adapter);

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.



# CHAP 4: