Author: Daniel DANG
School of Computing (SoC), Eastern Institute of Technology (EIT)
Email: daniel.dang.nz@gmail.com

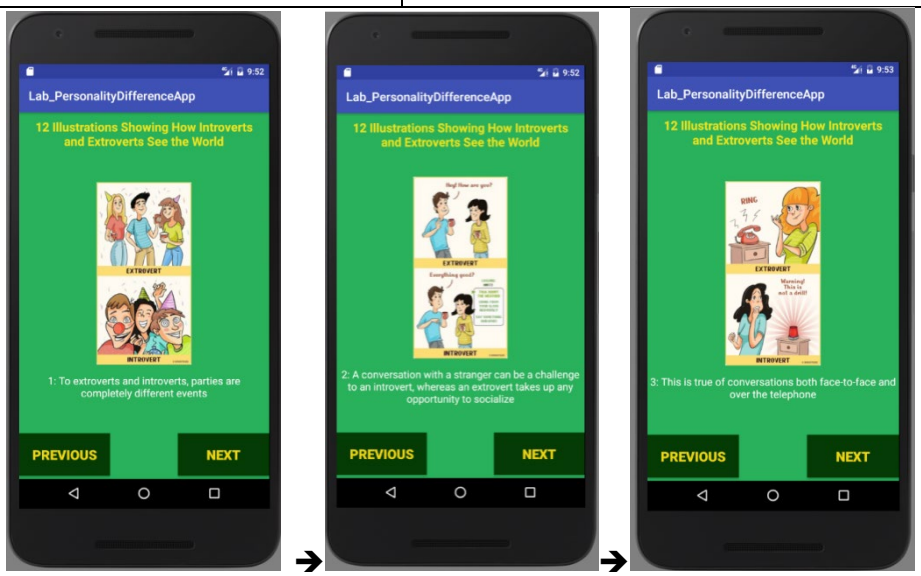
Lab 2: Personality Difference App
Activity life cycle and User Interaction in Android

1. Objectives

In this practical lab, students will learn how to develop a simple app by converting an article ("12 Illustrations Showing How Introverts and Extroverts See the World") into an Android Mobile App.

In this lab, students learn:

Design Graphic User Interface (GUI) in Android:	Java coding in Android:
- **LinearLayout** - **Design nested layout** - **Visual controls**: **Button, TextView, ImageView**	- **Find id** (visual controls) on Layout: **findViewbyID()** - **Set Click listener** and respond for visual controls - **Java array of data**: 1D-array - **Use Resources:** Colors & Strings - **Internet connections** in Android

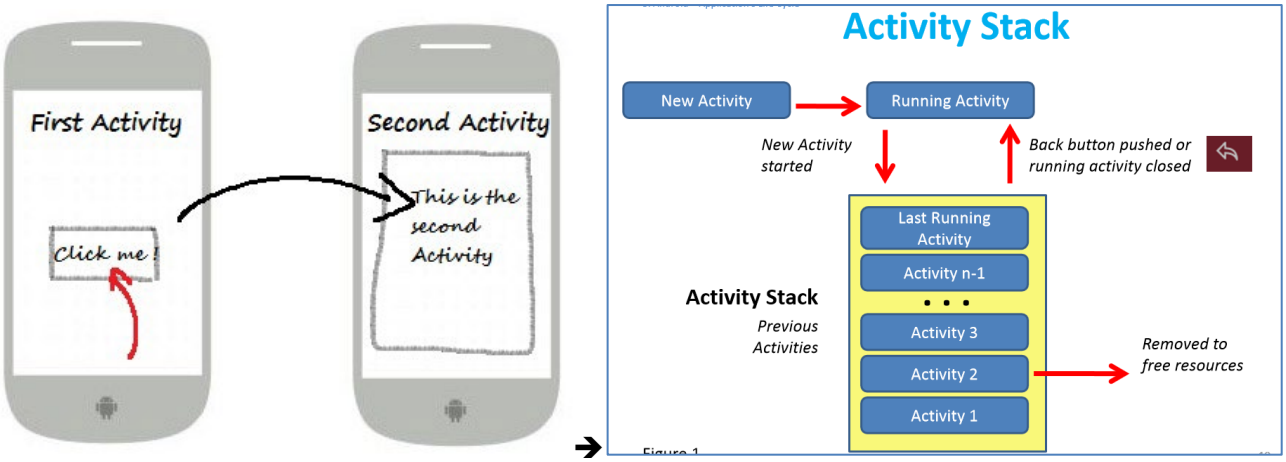


"Activity" in Android

Source: https://www.tutorialspoint.com/android/android_acitivities.htm

*An activity represents a **single screen** with a **user interface** just like **window** or **frame of Java**. Android activity is the subclass of **ContextThemeWrapper class**.*

If you have worked with C, C++ or Java programming language then you must have seen that your program starts from **main() function**. Very similar way, Android system initiates its program with in an **Activity starting with a call** on ***onCreate()* callback method**.



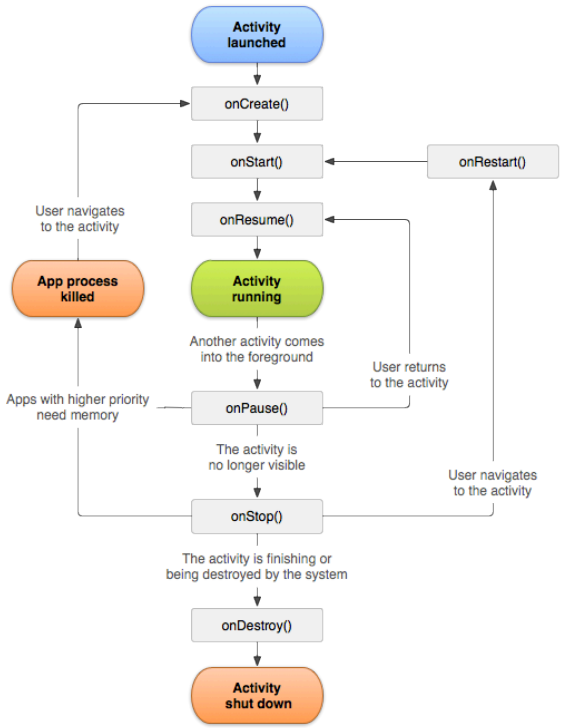
There are **two methods almost all subclasses of Activity** will implement:

- onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI, and using findViewById(int) to retrieve the widgets in that UI that you need to interact with programmatically.
- onPause() is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the ContentProvider holding the data).

Activities are one of the fundamental building blocks of apps on the Android platform. They serve as the entry point for a user's interaction with an app, and are also central to how a user navigates within an app (as with the Back button) or between apps (as with the Recents button).

There is a sequence of **callback methods** that start up an activity and a sequence of callback methods that tear down an activity as shown in the beside Activity life cycle diagram

The Activity class defines the following call backs i.e. events. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.



Activity life cycle diagram

	Callback & Description
1	**onCreate():** This is the first callback and called when the activity is first created.
2	**onStart():** This callback is called when the activity becomes visible to the user.
3	**onResume():** This is called when the user starts interacting with the application.
4	**onPause():** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
5	**onStop():** This callback is called when the activity is no longer visible.
6	**onDestroy():** This callback is called before the activity is destroyed by the system.
7	**onRestart():** This callback is called when the activity restarts after stopping it.

Convert the below article into a Mobile App

12 Illustrations Showing How Introverts and Extroverts See the World

Source: https://brightside.me/inspiration-psychology/12-illustrations-showing-how-introverts-and-extroverts-see-the-world-239260/?utm_source=fb_brightside&utm_medium=fb_organic&utm_campaign=fb_gr_brightside

*Everyone knows that **extroverts** and **introverts** see the world somewhat differently. We at **Bright Side** thought it would be interesting to illustrate these differences using specific examples. The results really make you think just how different we can be. **Illustrated by Anna Syrovatkina** for **BrightSide.me**.*

1. To extroverts and introverts, parties are completely different events.

2. A conversation with a stranger can be a challenge to an introvert, whereas an extrovert takes up any opportunity to socialize.

3. This is true of conversations both face-to-face and over the telephone.

4. Meeting a chatty person in the elevator is a great start to the day for an extrovert. But an introvert dreams of riding it alone.

5. A large office space is like heaven to an extrovert, but it's like being in a zoo for an introvert.

6. Both have their own strengths at work.

7. They understand leadership differently.

8. Home for an introvert is the best place on Earth. If you're an extrovert, it's just a place to catch your breath.

9. After an entire day of socializing, an extrovert is still full of energy. An introvert doesn't feel the same.

10. Which is why they see the subway trip home somewhat differently.

11. The same is true of their ideal evening after a hard day.

12. On the other hand, both look forward to relaxing at the weekend...but in their own way!

Analyze app function & design its user interface

App Graphic User Interface (GUI):

Below is the structure of the main layout:

- **2 TextViews: one displays "title" (*12 Illustrations Showing How Introverts and Extroverts See the World*) on top, another displays "caption" (*To extroverts and introverts, parties are completely different events*) below "illustration" image;**
- **1 ImageView: displays illustration image (located in the screen centre);**
- **Two Buttons: "Next" button (at the bottom right) and "Previous" button (at the bottom left);**

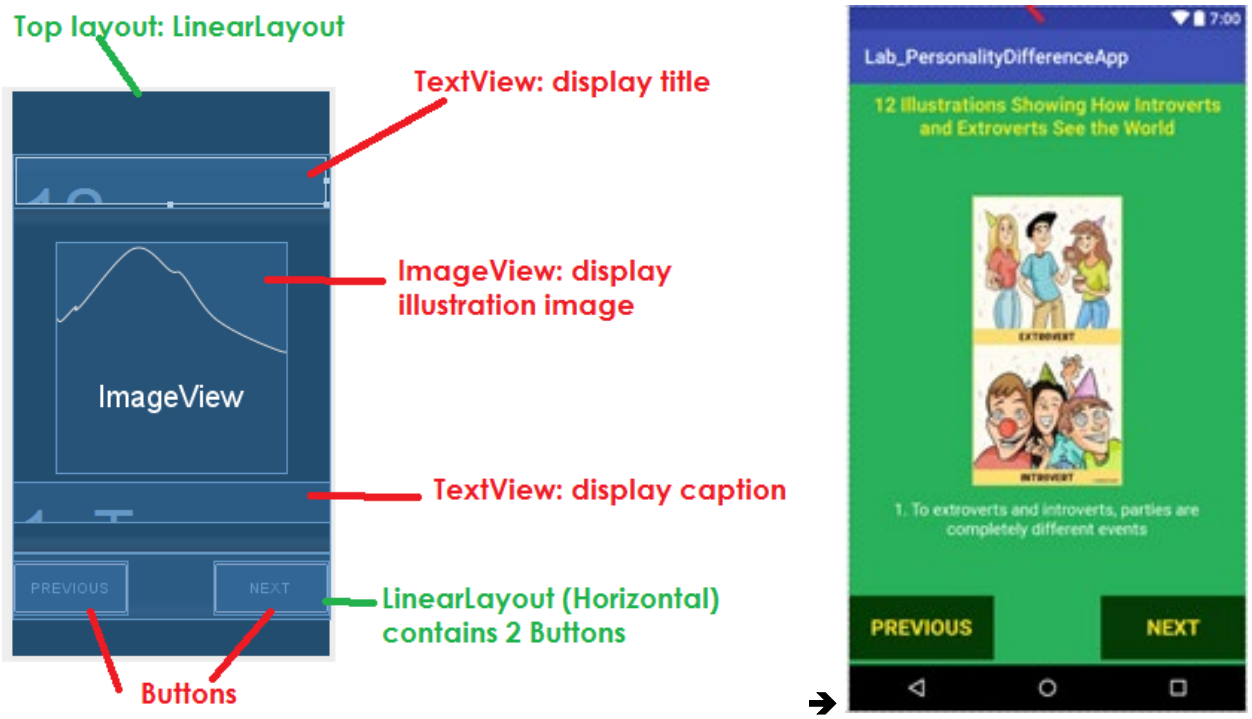


Figure 1: main layout structure of the app

App Function or Operation:

- When users launch app, the first screen will display the **first illustrationimage** and the **first caption**;
- When users click "**Next**" **button**, the app will display the next "**illustration**" image and its associated caption;
- When the "illustration" image is the 12th image (the last one) and users click "**Next**" button, the app will start over from the first illustration;
- When users click "**Previous**" **button**, the app will display the previous "**illustration**" image and its associated caption;
- When the "**illustration**" image is the first image and users click "**Previous**" button, the app will display the 12th illustration image (the last image);

PART 1: Create new project & Design main layout

Step 1: Create a new Android Project

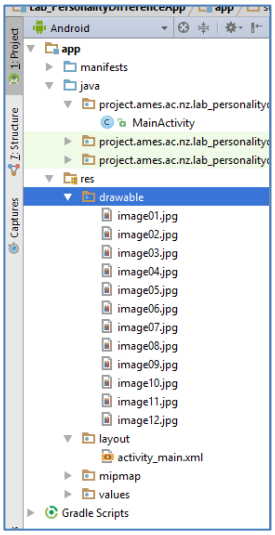
Create a new Android Project in Android Studio IDE:

- Application name: **Lab_PersonalityDifferenceApp_[Yourname]**
- Company Domain: **ac.ames.project.[yourlastname].[yourfirstname]**
- Target Android devices: **Phone and Tablet**;
  - Minimum SDK: **API 21: Android 5.0**
- Add an Activity to Mobile: **Empty Activity**;
  - Activity name: **MainActivity**
  - Layout name: **activity_main**

+ Target devices: run on 71.3% devices
_Smartphone & Tablet
_Android KitKat (5.0) & API21 (min)

Step 2: Design Graphic User Interface (GUI)

+ First of all, prepare resources (images & texts) for the app: copy 12 images into the /res/drawable folder:



+ Then, open and design the activity_main.xml layout, the first version of main layout contains:

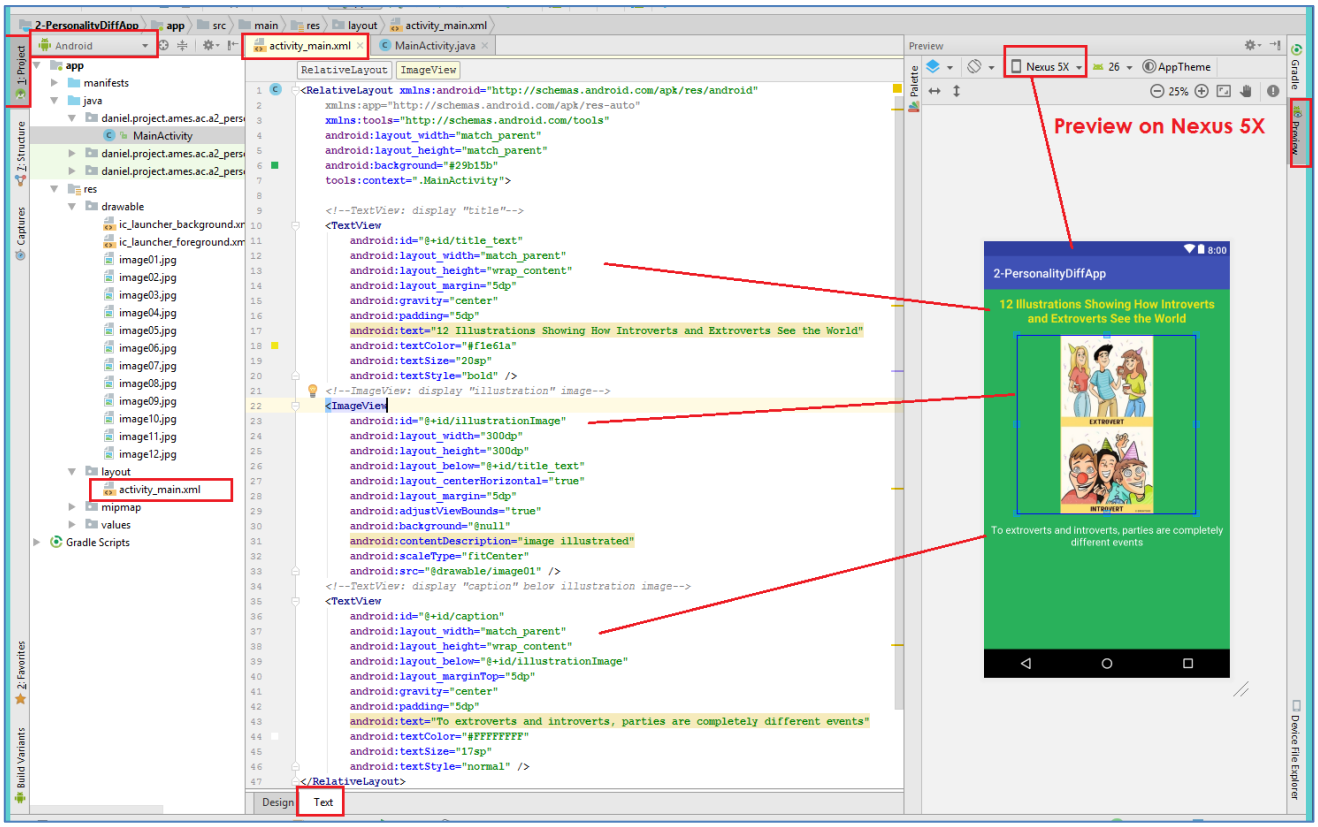
- 1 **TextView**: displays "title" (*12 Illustrations Showing How Introverts and Extroverts See the World*) on top
- 1 **ImageView**: displays illustration image (located in the screen centre);
- 1 **TextView**: displays "caption" (*To extroverts and introverts, parties are completely different events*) below "illustration" image;

+ Finally, the content of activity_main.xml file as below:

activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#29b15b"
    tools:context=".MainActivity">
```

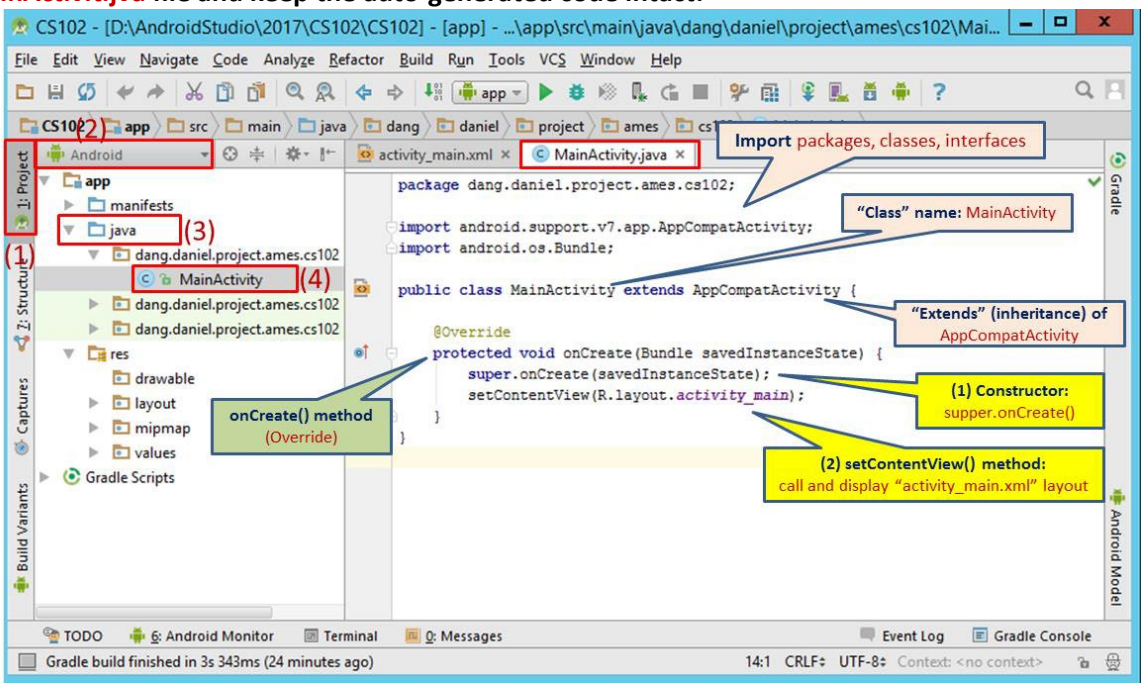
```
    <!--TextView: display "title"-->
    <TextView
        android:id="@+id/title_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="12 Illustrations Showing How Introverts and Extroverts See the World"
        android:textColor="#f1e61a"
        android:textSize="20sp"
        android:textStyle="bold" />

    <!--ImageView: display "illustration" image-->
    <ImageView
        android:id="@+id/illustrationImage"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_below="@+id/title_text"
        android:layout_centerHorizontal="true"
        android:layout_margin="5dp"
        android:adjustViewBounds="true"
        android:background="@null"
        android:contentDescription="image illustrated"
        android:scaleType="fitCenter"
        android:src="@drawable/image01" />

    <!--TextView: display "caption" below illustration image-->
    <TextView
        android:id="@+id/caption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/illustrationImage"
        android:layout_marginTop="5dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="To extroverts and introverts, parties are completely different events"
        android:textColor="#FFFFFFFF"
        android:textSize="17sp"
        android:textStyle="normal" />
</RelativeLayout>
```



+ Run your app on AVD (Nexus 5X) to see the result:

Step 3: Code the MainActivity: java coding

+ Open MainActivit.jva file and keep the auto-generated code intact:



+ In the MainActivity.java file, we will implement the below app functions:

- When users launch app, the first screen will display the first illustrationimage and the first caption;
- When users click "Next" button, the app will display the next "illustration" image and its associated caption;
- When the "illustration" image is the 12th image (the last one) and users click "Next" button, the app will start over from the first illustration;
- When users click "Previous" button, the app will display the previous "illustration" image and its associated caption;
- When the "illustration" image is the first image and users click "Previous" button, the app will display the 12th illustration image (the last image);

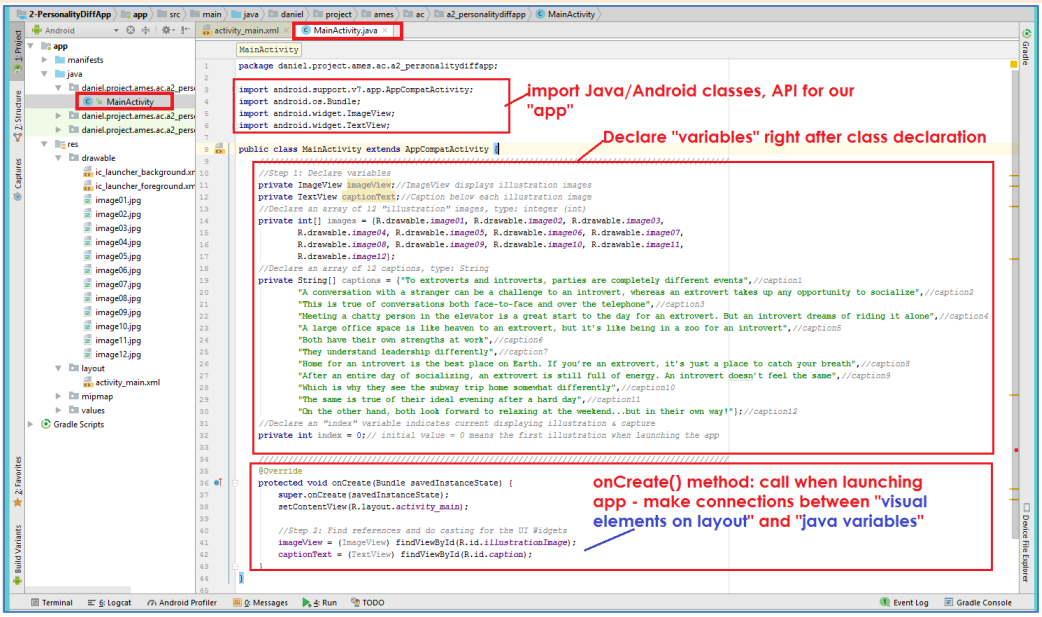
+ In term of java coding, we need to:

- **Declare few variables for the app**: ImageView, TextView, Button, 1D-array;
- **Create an array that contains 12 images**;
- **Create another array containing 12 captions accordingly**;
- **Set click listener** for the **"NEXT" and "PREVIOUS" buttons** so that when the users **click "NEXT" button**, the next illustration will be displayed. When users **click "PREVIOUS" button,** the previous illustration in comparison to the current one will be displayed.

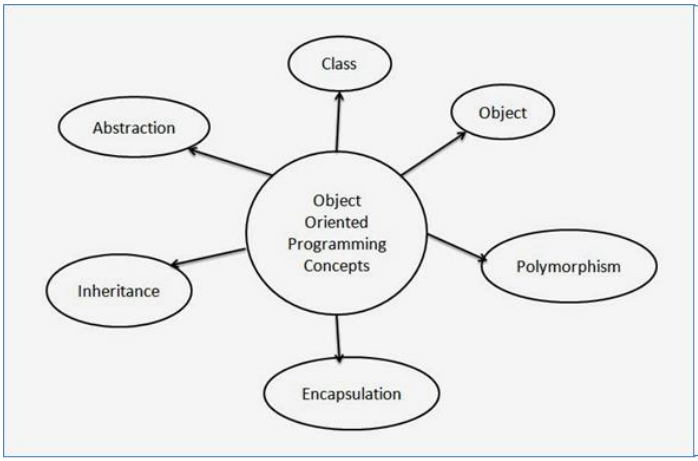
+ Now open and edit the MainActivity.java as below: *Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:*

MainActivity.java:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    ////////////////////////////////////////////////////////////////////////////////////////////
    //Step 1: Declare variables
    private ImageView imageView;//ImageView displays illustration images
    private TextView captionText;//Caption below each illustration image
    //Declare an array of 12 "illustration" images, type: integer (int)
    private int[] images = {R.drawable.image01, R.drawable.image02, R.drawable.image03,
            R.drawable.image04, R.drawable.image05, R.drawable.image06, R.drawable.image07,
            R.drawable.image08, R.drawable.image09, R.drawable.image10, R.drawable.image11,
            R.drawable.image12};
    //Declare an array of 12 captions, type: String
    private String[] captions = {"To extroverts and introverts, parties are completely different
events",//caption1
            "A conversation with a stranger can be a challenge to an introvert, whereas an extrovert takes up
any opportunity to socialize",//caption2
            "This is true of conversations both face-to-face and over the telephone",//caption3
            "Meeting a chatty person in the elevator is a great start to the day for an extrovert. But an
introvert dreams of riding it alone",//caption4
            "A large office space is like heaven to an extrovert, but it's like being in a zoo for an
introvert",//caption5
            "Both have their own strengths at work",//caption6
            "They understand leadership differently",//caption7
            "Home for an introvert is the best place on Earth. If you're an extrovert, it's just a place to
catch your breath",//caption8
            "After an entire day of socializing, an extrovert is still full of energy. An introvert doesn't
feel the same",//caption9
            "Which is why they see the subway trip home somewhat differently",//caption10
            "The same is true of their ideal evening after a hard day",//caption11
            "On the other hand, both look forward to relaxing at the weekend...but in their own
way!"};//caption12
    //Declare an "index" variable indicates current displaying illustration & capture
    private int index = 0;// initial value = 0 means the first illustration when launching the app

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

        //Step 2: Find references and do casting for the UI Widgets
        imageView = (ImageView) findViewById(R.id.illustrationImage);
        captionText = (TextView) findViewById(R.id.caption);
    }
}
```



Explanation java code:

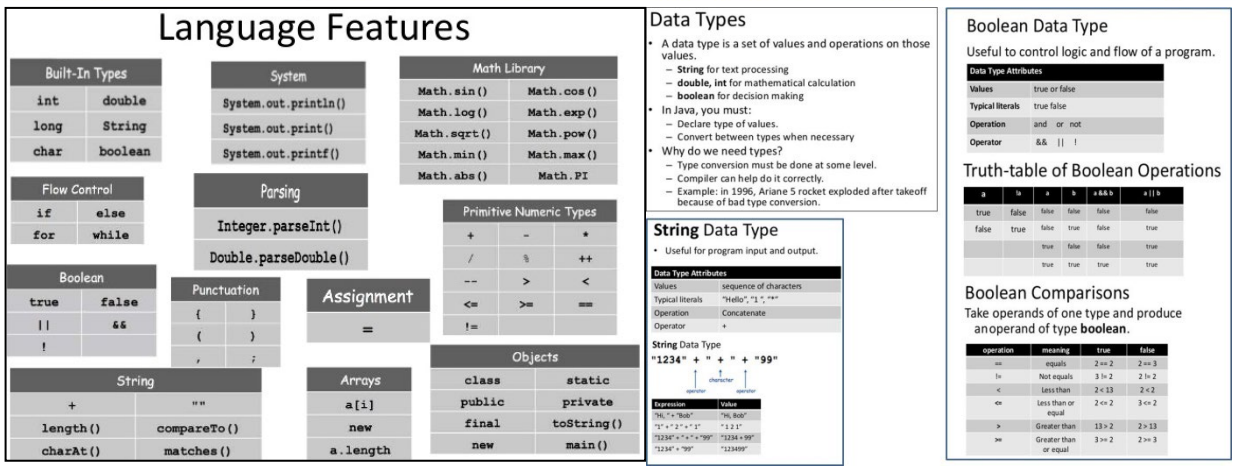


Object-oriented Programming
– Fundamentals
- Classes & Objects
- Methods & Constructors
- Encapsulation
- Inheritance

– Libraries & Clients

Java programming language was originally developed by **Sun Microsystems** which was initiated by James Gosling and **released in 1995** as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

- **Object:** is the **basic unit** of object oriented programming. That is **both data and function that operate on data** are bundled as a unit called as object.
- **Class:** When you define a class, **you define a blueprint for an object**. This doesn't actually define any data, but it does define **what the class name** means, **what data (properties) an object of the class will consist of** and **what operations (functions) can be performed** on such an object.
- **Encapsulation:** is binding the class's data and behaviours together in a single unit. Also it is a language mechanism for **restricting access to some components** (achieved by access modifiers like **private**, **public**, **protected** etc.).
- **Inheritance:** One of the **most useful aspects of object-oriented programming** is **code reusability**. As the name suggests Inheritance is **the process of forming a new class** (called as derived class or subclass) **from an existing class** (called as base class or superclass or parent class). This feature **helps to reduce the code size.**
- **Abstraction:** is a process where you **show only "relevant" data and "hide" unnecessary details of an object** from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, what happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.
- **Polymorphism:** the ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.
- **Overloading:** the concept of overloading is also **a branch of polymorphism**. When the exiting operator or function is made to **operate on new data type**, it is said to be overloaded.
- **Override:** to override the **functionality of an existing method**. If a class inherits a method from its **superclass**, then there is a chance to override the method provided that it is not marked final. The benefit of overriding is: ability to define a behaviour that's specific to the **subclass type**, which means a subclass can implement a parent class method based on its requirement.



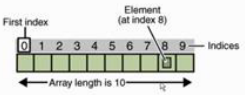
What is an array?

Dimensions	Example	Terminology
1	0 1 2	Vector
2	0 1 2 / 3 4 5 / 6 7 8	Matrix
3	0 1 2 / 3 4 5 / 6 7 8	3D Array (3rd order Tensor)
N	...	ND Array

Creating 1-D Array

- *type arrayname [] = new type [size];*
- Example

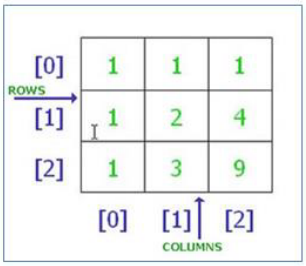
  double stockPrices [] = new double[10];




```
int resultArray[] = new int[6];
```

Type — Array Name — Array Size

Two-Dimensional Arrays



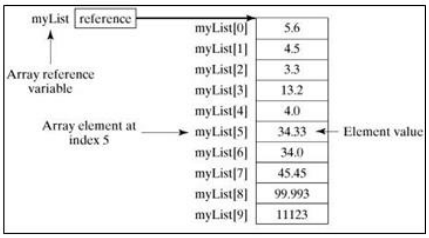

```
int M = 10;
int N = 3;
double[][] a = new double[M][N];
```

The Arrays Class

- Defined in the `java.util` package.
- Contains static methods for manipulating arrays:
  - `binarySearch`: search for a value.
  - `equals`: compare the contents of two arrays.
  - `fill`: fill an array with a particular value.
  - `sort`: sort the contents of an array.

Difference between array and an Arraylist

- An array is of fixed size
- An ArrayList can grow and reduce in size
- Any collection can grow and reduce in size but arrays cannot
- i.e. ArrayList list = new ArrayList();
  list.add(new Integer()); //is allowed
- But this is not allowed:
  Integer[] intArr = new Integer[3];
  intArr.add(new Integer());



String Methods

- The **String** class contains many useful methods for string-processing applications.
  - A **String** method is called by writing a **String** object, a dot, the name of the method, and a pair of parentheses to enclose any arguments
  - If a **String** method returns a value, then it can be placed anywhere that a value of its type can be used

```
String greeting = "Hello";
int count = greeting.length();
System.out.println("Length is " + greeting.length());
```

String method

  - Always count from zero when referring to the *position* or *index* of a character in a string

String Methods

Advantage of String class: many built-in methods for String manipulation

```
str.length();          // get length of string
str.toLowerCase()      // convert to lower case
str.toUpperCase()      // convert to upper case
str.charAt(i)          // what is at character i?
str.contains(..)       // String contains another string?
str.startsWith(..)     // String starts with some prefix?
str.indexOf(..)        // what is the position of a character?
....many more
```

+ Compile and run your app on AVD to see the result:

So far, the app only displays the first illustration image and the first caption.

PART 2: Implement user interaction: when users click "Next" Button, the app will display the next "illustration" image and its associated "caption"

Now, have a look again at the app function:

- When users launch app, the first screen will display the first illustrationimage and the first caption;
- When users click "Next" button, the app will display the next "illustration" image and its associated caption;
- When the "illustration" image is the 12th image (the last one) and users click "Next" button, the app will start over from the first illustration;
- When users click "Previous" button, the app will display the previous "illustration" image and its associated caption;
- When the "illustration" image is the first image and users click "Previous" button, the app will display the 12th illustration image (the last image);

In this part, we will implement the "NEXT" Button:

- Add "**NEXT**" button to the **main layout**;
- **Set click listener** for the **"NEXT" and "PREVIOUS" buttons** so that when the users **click "NEXT" button**, the next illustration will be displayed.

+ First of all, add a Button to the main layout: This button locates at the right botton of screen:

```
<!--Button: "NEXT"-->
<Button
    android:id="@+id/nextBtn"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_margin="5dp"
    android:background="#053b05"
    android:gravity="center"
    android:padding="20sp"
    android:text="Next"
    android:textColor="#efd514"
    android:textSize="22sp"
    android:textStyle="bold" />
```

+ Compile and run your app on AVD to see the result:

+ Open the MainActivity and add new java code to implement the function:

- When users click "**Next**" button, the app will display the next "**illustration**" image and its associated caption;
- When the "illustration" image is the 12th image (the last one) and users click "**Next**" button, the app will start over from the first illustration;

In term of java coding, we need to:

- **Set click listener** for the **"NEXT" and "PREVIOUS" buttons** so that when the users **click "NEXT" button**, the next illustration will be displayed.

+ Now open the MainActivity.java and follow the below instructions:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "nextBtn" right after the class declaration:

```
//"Next" button - 1: Declare a variable "nextBtn"
private Button nextBtn;//Buttons allow users to navigate illustrations back and fort
```

_Inside onCreate() method, add code to find reference for "nextBtn" Button:

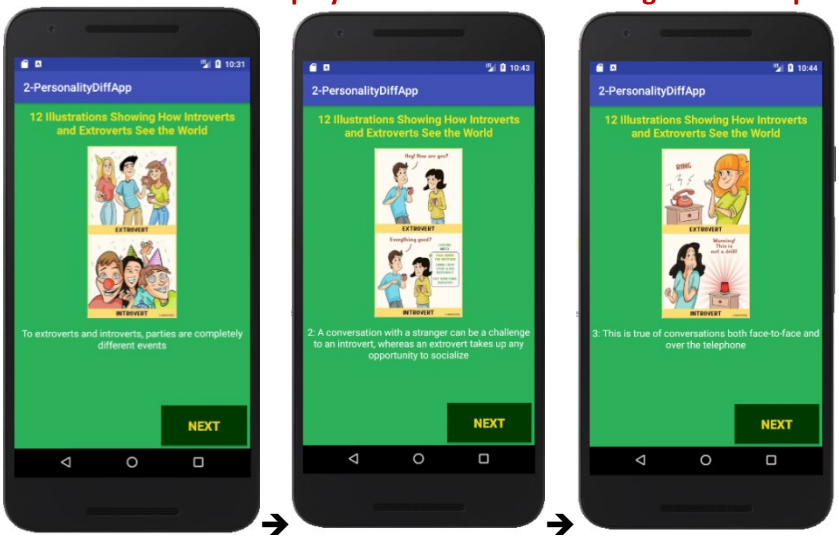
```
//"Next" button - 2: Find references and do casting for "nextBtn" Button
nextBtn = (Button) findViewById(R.id.nextBtn);
```

_Inside onCreate() method, add code to set click listener for "nextBtn" Button:

```
//"Next" button - 3: Set click listener for "nextBtn" Button
nextBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Check if index is equal to 11 (12nd illustration)? If yes, set index=0 to start over
        if (index == 11) {
            index = 0;//start over slide show from beginning
            imageView.setImageResource(images[index]);
            captionText.setText((index + 1) + ": " + captions[index]);
        } else {
            index++;//Increase index by 1 to move to the next illustration
            imageView.setImageResource(images[index]);
            captionText.setText((index + 1) + ": " + captions[index]);
        }
    }
});
```

+ Compile and run your app on AVD to see the result:

Click "NEXT" button to display the next illustration image and its caption:

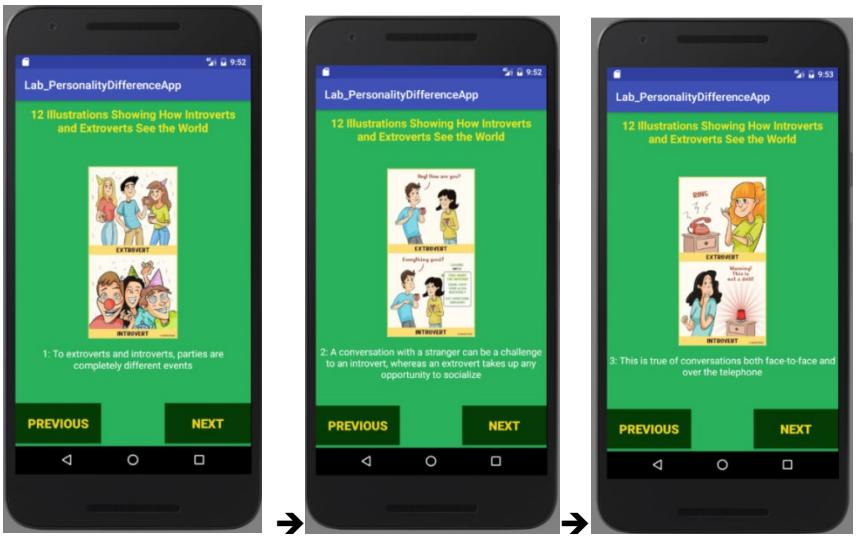


Experiment:

Exercise 1: Add "PREVIOUS" BUTTON

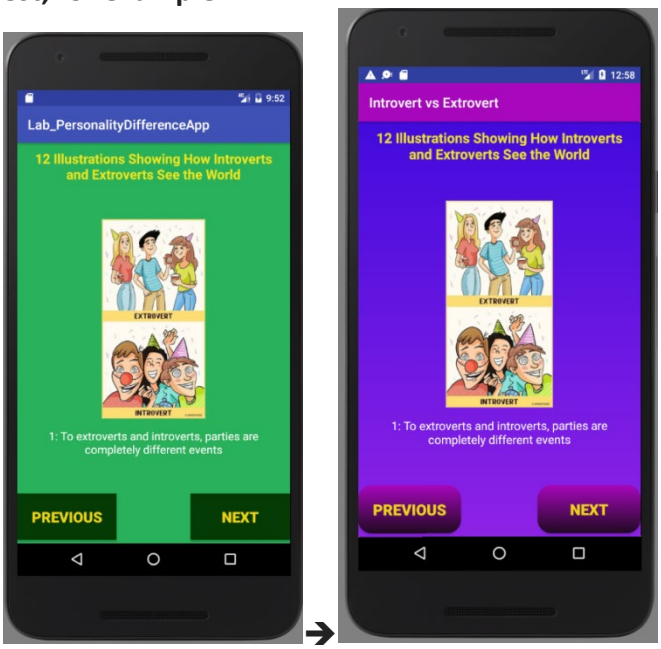
It's your turn to add another Button "PREVIOUS" located at the left bottom of screen:

- When users click "**Previous**" button, the app will display the previous "illustration" image and its associated caption;
- When the "illustration" image is the first image and users click "**Previous**" button, the app will display the 12th illustration image (the last image);



Exercise 2: Improve "app" GUI

Change the look of user interface (app style, app name, background color for layout and buttons) according to your own interest, for example:



Exercise 3: Use "resource" in Android

At the moment, we declare an array of 12 captions in MainActivity.java file. It's a best practice if we declare all those captions in "resource" file - strings.xml.

The below instructions will show you how to declare an array of strings (or texts) in **strings.xml file**:

+ First of all, open strings.xml file in "values" folder and declare a caption_array:

```
<resources>
    <string name="app_name">Introvert vs Extrovert</string>

    <!--Declare an array of 12 captions-->
    <string-array name="caption_array">
        <!--Caption 1-->
        <item>To extroverts and introverts, parties are completely different events</item>
        <!--Caption 2-->
        <item>A conversation with a stranger can be a challenge to an introvert, whereas an extrovert takes up any opportunity to socialize</item>
        <!--Caption 3-->
        <item>This is true of conversations both face-to-face and over the telephone</item>
        <!--Caption 4-->
        <item>Meeting a chatty person in the elevator is a great start to the day for an extrovert. But an introvert dreams of riding it alone</item>
        <!--Caption 5-->
        <item>A large office space is like heaven to an extrovert, but it's like being in a zoo for an introvert</item>
        <!--Caption 6-->
        <item>Both have their own strengths at work</item>
        <!--Caption 7-->
        <item>They understand leadership differently</item>
        <!--Caption 8-->
        <item>Home for an introvert is the best place on Earth. If you're an extrovert, it's just a place to catch your breath</item>
        <!--Caption 9-->
        <item>After an entire day of socializing, an extrovert is still full of energy. An introvert doesn't feel the same</item>
        <!--Caption 10-->
        <item>Which is why they see the subway trip home somewhat differently</item>
        <!--Caption 11-->
        <item>The same is true of their ideal evening after a hard day</item>
        <!--Caption 12-->
        <item>On the other hand, both look forward to relaxing at the weekend... but in their own way!</item>
    </string-array>
</resources>
```

+ Open MainActivity.java file, and edit it as below:

_Delete "captions" variable and declare a new variable "caption_array" that contains 12 elements:

```
//Declare an array of 12 captions, type: String
private String[] caption_array = new String[12];
/*
private String[] captions = {"To extroverts and introverts, parties are completely different events",//caption1
        "A conversation with a stranger can be a challenge to an introvert, whereas an extrovert takes up any opportunity to socialize",//caption2
        "This is true of conversations both face-to-face and over the telephone",//caption3
        "Meeting a chatty person in the elevator is a great start to the day for an extrovert. But an introvert dreams of riding it alone",//caption4
        "A large office space is like heaven to an extrovert, but it's like being in a zoo for an introvert",//caption5
        "Both have their own strengths at work",//caption6
        "They understand leadership differently",//caption7
        "Home for an introvert is the best place on Earth. If you're an extrovert, it's just a place to catch your breath",//caption8
        "After an entire day of socializing, an extrovert is still full of energy. An introvert doesn't feel the same",//caption9
        "Which is why they see the subway trip home somewhat differently",//caption10
        "The same is true of their ideal evening after a hard day",//caption11
        "On the other hand, both look forward to relaxing at the weekend...but in their own way!"};//caption12
*/
```

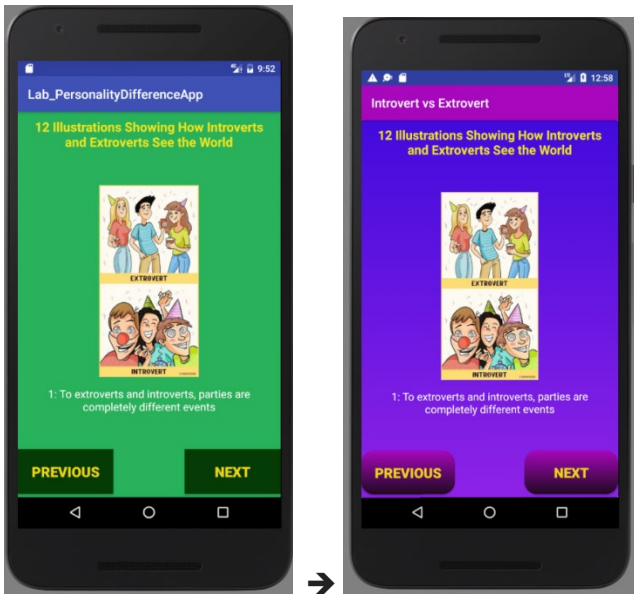
_Inside onCreate() method, retrieve the caption_array declared in strings.xml file by using getResource() method:

```
//Retrieve the "caption_array" declared in "strings.xml" file by using getResource() method
caption_array = getResources().getStringArray(R.array.caption_array);
```

_Inside onCreate() method, whenever appearing "captions" variable will be replaced by "caption_array" variable:

```
//3: Display the first image and first caption when launching the app
imageView.setImageResource(images[index]);
//captionText.setText((index + 1) + ": " + captions[index]);
captionText.setText((index + 1) + ": " + caption_array[index]);

//4: Set click listener for "NEXT" button
nextBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //5: Check if index is equal to 11 (12nd illustration)? If yes, set index=0 to start over
        if (index == 11) {
            index = 0;//start over slide show from beginning
            imageView.setImageResource(images[index]);
            //captionText.setText((index + 1) + ": " + captions[index]);
            captionText.setText((index + 1) + ": " + caption_array[index]);

        } else {
            index++;//Increase index by 1 to move to the next illustration
            imageView.setImageResource(images[index]);
            //captionText.setText((index + 1) + ": " + captions[index]);
            captionText.setText((index + 1) + ": " + caption_array[index]);
        }
    }
});

//6: Set click click listener for "PREVIOUS" button
previousBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //7: Check if index is equal to 0? If yes, set index=11 to display the 12th illustration
        if (index == 0) {
            index = 11;//set the 12nd illustration
            imageView.setImageResource(images[index]);
            //captionText.setText((index + 1) + ": " + captions[index]);
            captionText.setText((index + 1) + ": " + caption_array[index]);
        } else {
            index--;//Decrease index by 1 to move back an illustration
            imageView.setImageResource(images[index]);
            //captionText.setText((index + 1) + ": " + captions[index]);
            captionText.setText((index + 1) + ": " + caption_array[index]);
        }
    }
});
```

+ Now run the app to the result:



PART 3: Download Image from Internet (URL) and load it to ImageView
Network connection & Internet in Android (Self-directed learning)

The app "**Introvert vs Extrovert**" only displays illustration images stored in "**drawable**" folder. You can see it takes us time to **download images from Internet and then copy & paste to "drawable" folder**. Furthermore, the app size will increase significantly because the app containing all pictures in it.

Now, we want the app download images directly from Internet and display them on ImageView.

Android - Network Connection

Source: https://www.tutorialspoint.com/android/android_network_connection.htm

Android lets your application **connect to the internet** or any **other local network** and allows you to perform network operations. A device can have various types of network connections. This chapter focuses on using either a **Wi-Fi or a mobile network connection**.

Checking Network Connection

Before you perform any network operations, you must first check that are you connected to that network or internet e.t.c. For this android provides **ConnectivityManager** class. You need to instantiate an object of this class by calling **getSystemService()** method. Its syntax is given below:

```
ConnectivityManager check = (ConnectivityManager)
this.context.getSystemService(Context.CONNECTIVITY_SERVICE);
```

Once you instantiate the object of ConnectivityManager class, you can use **getAllNetworkInfo** method to get the information of all the networks. This method returns an array of **NetworkInfo**. So you have to receive it like this.

```
NetworkInfo[] info = check.getAllNetworkInfo();
```

The last thing you need to do is to check **Connected State** of the network. Its syntax is given below:

```
for (int i = 0; i<info.length; i++){
  if (info[i].getState() == NetworkInfo.State.CONNECTED){
    Toast.makeText(context, "Internet is connected
    Toast.LENGTH_SHORT).show();
  }
}
```

Apart from this connected states, there are other states a network can achieve. They are listed below:

	State
1	Connecting
2	Disconnected
3	Disconnecting
4	Suspended
5	Unknown

Performing Network Operations

After checking that you are connected to the internet, you can perform any network operation. Here we are fetching the html of a website from a **url**.

Android provides **HttpURLConnection** and **URL** class to handle these operations. You need to instantiate an object of URL class by providing the link of website. Its syntax is as follows:

```
String link = "http://www.google.com";
URL url = new URL(link);
```

After that you need to call **openConnection** method of url class and receive it in a HttpURLConnection object. After that you need to call the **connect** method of HttpURLConnection class.

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.connect();
```

And the last thing you need to do is to fetch the HTML from the website. For this you will use **InputStream** and **BufferedReader** class. Its syntax is given below:

```
InputStream is = conn.getInputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(is, "UTF-8"));
String webPage = "",data="";

while ((data = reader.readLine()) != null){
  webPage += data + "\n";
}
```

Apart from this **connect() method**, there are other methods available in HttpURLConnection class. They are listed below:

	Method & description
1	**disconnect():** This method releases this connection so that its resources may be either reused or closed
2	**getRequestMethod():** This method returns the request method which will be used to make the request to the remote HTTP server
3	**getResponseCode():** This method returns response code returned by the remote HTTP server
4	**setRequestMethod(String method):** This method Sets the request command which will be sent to the remote HTTP server
5	**usingProxy():** This method returns whether this connection uses a proxy server or not

Now, open the "Introvert vs Extrovert" app and then follow the below instructions:

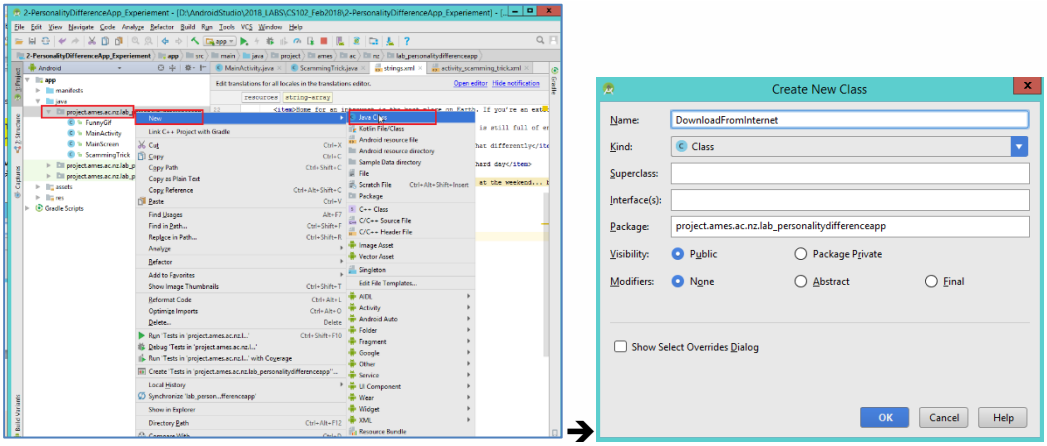
Step 1: Create a new java class "DownloadFromInternet" containing one function DownloadImage() allow us to download an image from Internet

+ In the same folder containing MainActivity.java, add a new class (java file):

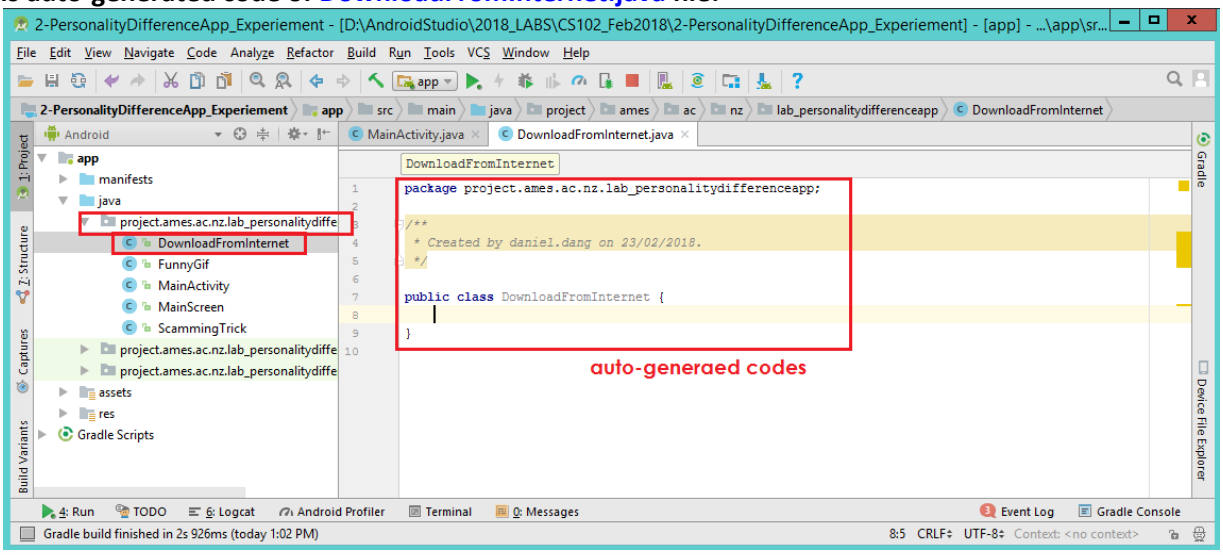
*_Right click on the package folder → **"New"** → **"Java class"** → a window is appeared;*

*_Enter the "name": **DownloadfromInternet;***

*_Click "**Ok**";*



+ Here is auto-generated code of DownloadFromInternet.java file:



+ Keep the auto-generated codes and add the following codes to DownloadFromInternet.java file: add downloadImage() method that takes in a string of URL where to download image on Internet:

Step 1: Declare variables

Step 2: Call OpenHttpConnection() method to open a connection to a HTTP server by using "try ...catch" to detect error if the connection fails to establish

- ✓ 1: Create URL or link of image to be downloaded
- ✓ 2: Establish or open an "Internet connection"
- ✓ 3: Check whether the "internet connection" is valid? If not, throw an Exception "Not an HTTP connection"
- ✓ 4: When the "internet connection" is valid, download image from Internet in the stream format by creating a httpsURLConnect object is cast into an HttpURLConnection object a
- ✓ 5: Set properties of the HTTP connection
- ✓ 6: Connect to the HTTP server and get a response from the server
- ✓ 7: If the "response" code is HTTP_OK, you then get the InputStream object from the connection so that you can begin to read incoming data from the server

Step 3: Decode the downloaded image by using decodeStream() method of the BitmapFactory class

- ✓ 8: Use decodeStream() method to decode the downloaded "InputStream" into a bitmap image
- ✓ 9: Close the inputStream when finishing decode process

DownloadFromInternet.java

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import javax.net.ssl.HttpsURLConnection;

public class DownloadFromInternet {
    ////////////////////////////////////////////////////////////////////////////////////////////////
    //Add downloadImage() method  that takes in a string of URL where to download image on Internet
    public static Bitmap DownloadImage(String downloadImageURL) {
        ////////////////////////////////////////////////////////////////
        //Step 1: Declare variables
        Bitmap bitmap = null;//This variable to store downloaded image from Internet
        InputStream inputStream = null;//Image will be downloaded in "input-stream" form from Internet
        int response = -1;//Store the resulted "status" when connecting app to Internet
        String urlString = downloadImageURL;//Unique Resource Locator:  web address which is a reference
                                            //to a web resource that specifies its location on a computer network

        ////////////////////////////////////////////////////////////////
        //Step 2: Call OpenHttpConnection() method to open a connection to a HTTP server by using
        //        "try ...catch" to detect error if the connection fails to establish
        try {
            //1: Create URL or link of image to be downloaded
            URL imageURL = new URL(urlString);
            //2: Establish or open an "Internet connection"
            URLConnection connection = imageURL.openConnection();
            //3: Check whether the "internet connection" is valid? If not, throw an Exception "Not an HTTP  connection"
            if (!(connection instanceof HttpsURLConnection)) {
                throw new IOException("URL is not an HTTP  URL");
            }
            //4: When the "internet connection" is valid, download image from Internet in the stream format by
            //    creating a httpsURLConnect object is cast into an HttpURLConnection object
            HttpsURLConnection httpsURLConnection = (HttpsURLConnection) connection;
            //5: Set properties of the HTTP connection
            httpsURLConnection.setAllowUserInteraction(false);
            httpsURLConnection.setInstanceFollowRedirects(true);
            httpsURLConnection.setRequestMethod("GET");//Type of Request: GET
            httpsURLConnection.connect();
            //6: Connect to the HTTP server and get a response from the server
            response = httpsURLConnection.getResponseCode();
            //7: If the "response" code is HTTP OK, you then get the InputStream object from the connection
            //    so that you can begin to read incoming data from the server
            if (response == httpsURLConnection.HTTP_OK) {
                //If the connection is OKAY, read incoming data from the server
                inputStream = httpsURLConnection.getInputStream();
            }

        } catch (Exception e) {
            //In case of error detected, display a message "Error Connecting" on "Logcat" screen
            Log.e("OpenHttpConnection", "Error connecting");
        }

        ////////////////////////////////////////////////////////////////
        //Step 3: Decode the downloaded image by using decodeStream() method of the BitmapFactory class
        try {
            //8: Use decodeStream() method to decode the downloaded "InputStream" into a bitmap image
            bitmap = BitmapFactory.decodeStream(inputStream);
            //9: Close the inputStream when finishing decode process
            inputStream.close();

        } catch (IOException ex) {
            //In case of error detected, display the type of error on "Logcat" screen
            ex.printStackTrace();
        }

        ////////////////////////////////////////////////////////////////
        //Step 4: Return the decoded bitmap as output
        return bitmap;
    }

}
```

Step 2: Inside MainActivity.java and create a "Thread" to download Images from Internet

+ First of all, we collect 12 URL of 12 illustration images from link:
https://brightside.me/inspiration-psychology/12-illustrations-showing-how-introverts-and-extroverts-see-the-world-239260/?utm_source=fb_brightside&utm_medium=fb_organic&utm_campaign=fb_gr_brightside

 URL: https://files.brightside.me/files/news/part_23/239260/12550610-24329505-1-1474470168-650-9-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12550660-24329605-2-1474470185-650-19-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12550710-24329705-3-1474470203-650-11-1488295420-650-d81043295b-1490686715.jpg
 URL: https://files.brightside.me/files/news/part_23/239260/12550760-24329805-4-1474470223-650-12-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12550810-24329905-5-1474470240-650-7-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12550860-24330205-6-1474470289-650-8-1488295420-650-d81043295b-1490686715.jpg
 URL: https://files.brightside.me/files/news/part_23/239260/12550910-24330105-7-1474470272-650-8-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12550960-24330305-8-1474470302-650-18-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12551010-24330605-9-1474470375-650-10-1488295420-650-d81043295b-1490686715.jpg
 URL: https://files.brightside.me/files/news/part_23/239260/12551060-24330705-10-1474470387-650-27-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12551110-24330505-11-1474470362-650-9-1488295420-650-d81043295b-1490686715.jpg	 URL: https://files.brightside.me/files/news/part_23/239260/12551160-24330405-12-1474470342-650-6-1488295420-650-d81043295b-1490686715.jpg

+ Then open MainActivity.java file and edit it as below:

_Delete "images" variable and declare a new variable "imageURLs" (type: String) that contains 12 elements:

```
//Declare an array of 12 "illustration" image urls, type: String
private String[] imageURLs = new String[12];
/*
```

```
//Declare an array of 12 "illustration" images, type: integer (int)
private int[] images = {R.drawable.image01, R.drawable.image02, R.drawable.image03,
        R.drawable.image04, R.drawable.image05, R.drawable.image06, R.drawable.image07,
        R.drawable.image08, R.drawable.image09, R.drawable.image10, R.drawable.image11,
        R.drawable.image12};
*/
```

_Declare a new variable, called imageDownload_Handler, that creates a THREAD in Android to download image:

```
//Declare a Hanlder or Thread
private Handler imageDownload_Handler = new Handler();
```

_Inside onCreate() method, populate the imageURLs array with 12 illustration image URLs:

```
//Populate populate the imageURLs array with 12 illustration image URLs
imageURLs[0] = "https://files.brightside.me/files/news/part_23/239260/12550610-24329505-1-1474470168-650-9-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[1] = "https://files.brightside.me/files/news/part_23/239260/12550660-24329605-2-1474470185-650-19-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[2] = "https://files.brightside.me/files/news/part_23/239260/12550710-24329705-3-1474470203-650-11-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[3] = "https://files.brightside.me/files/news/part_23/239260/12550760-24329805-4-1474470223-650-12-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[4] = "https://files.brightside.me/files/news/part_23/239260/12550810-24329905-5-1474470240-650-7-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[5] = "https://files.brightside.me/files/news/part_23/239260/12550860-24330205-6-1474470289-650-8-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[6] = "https://files.brightside.me/files/news/part_23/239260/12550910-24330105-7-1474470272-650-8-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[7] = "https://files.brightside.me/files/news/part_23/239260/12550960-24330305-8-1474470302-650-18-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[8] = "https://files.brightside.me/files/news/part_23/239260/12551010-24330605-9-1474470375-650-10-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[9] = "https://files.brightside.me/files/news/part_23/239260/12551060-24330705-10-1474470387-650-27-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[10] = "https://files.brightside.me/files/news/part_23/239260/12551110-24330505-11-1474470362-650-9-
1488295420-650-d81043295b-1490686715.jpg";
imageURLs[11] = "https://files.brightside.me/files/news/part_23/239260/12551160-24330405-12-1474470342-650-6-
1488295420-650-d81043295b-1490686715.jpg";
```

_Inside onCreate() method, whenever we display image on ImageView, we change the existing code *imageView.setImageResource(images[index])* by below java code snippet that calls DownloadImage() method of DownloadFromInternet class:

Delele this code:

```
imageView.setImageResource(images[index]);
```

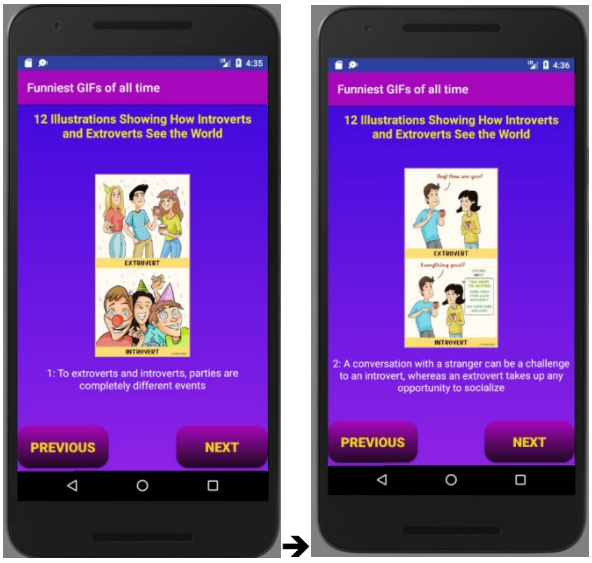
and replace it with below snippet:

```
//Calls DownloadImage() method of DownloadFromInternet class in a THREAD
new Thread(new Runnable() {
    @Override
    public void run() {
        imageDownload_Handler.post(new Runnable() {
            //Perform loading image from Internet
            Bitmap bitmap = DownloadFromInternet.DownloadImage(imageURLs[index]);

            @Override
            public void run() {
                //Display the download image on the ImageView
                imageView.setImageBitmap(bitmap);
            }
        });
    }
}).start();
```

+ Add "permission" for the app to access the Internet in AndroidManifest.xml file:

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

+ Compile and run your app on AVD to see the result:



You can observe that illustration images are loaded from Internet **quite slowly** in comparision with loading from "**drawable**" folder.

Note: Image formats support by Android

Source: https://developer.android.com/guide/topics/media/media-formats.html

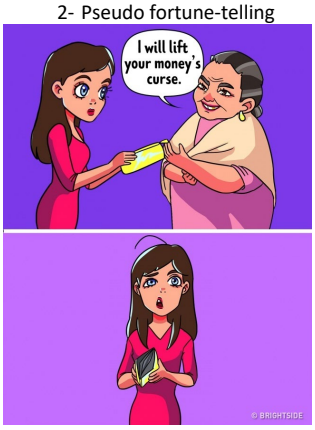
Format / Codec	Encoder	Decoder	Details	Supported File Type(s) / Container Formats
BMP		•		BMP (.bmp)
GIF		•		GIF (.gif)
JPEG	•	•	Base + Progressive	JPEG (.jpg)
PNG	•	•		PNG (.png)
WebP	• (Android 4.0+) (Lossless, Transparency, Android 4.2.1+)	• (Android 4.0+) (Lossless, Transparency, Android 4.2.1+)		WebP (.webp)

PART 4: "10 Devious Scamming Tricks" App (Self-directed learning)

Your task is to convert the below article "10 Devious Scamming Tricks That Tourists Regularly Fall For" into an Android Mobile App:

Link: https://brightside.me/inspiration-tips-and-tricks/10-devious-scamming-tricks-that-tourists-regularly-fall-for-428010/

"In 2016, there were approximately 1.235 billion international tourists around the world. In relation to 2015, this was an increase of 4%. At the same time, France, the USA, Spain, China, Italy, and the United Kingdom became the most popular countries among tourists. Along with the growing number of tourists, the number of deceived travelers also increases. After all, when visiting foreign countries, we often trust locals a bit too much. And the realization that we have been deceived might come too late.
We at Bright Side decided to share the most common tricks that scammers and thieves use in almost every country around the world".

1-Lost items	2- Pseudo fortune-telling	3- Private taxis	4-Fares
			
Imagine yourself walking quietly through an unfamiliar city. You suddenly notice that on the sidewalk there is a ring, a bracelet, or some other rather expensive item. You pick it up, and at this very moment someone comes up to you and starts strongly encouraging you not to look for the owner (they say it's practically impossible) but to split the value of the item in half. Since he does not have any cash with him, the stranger asks you to give him a certain amount, and you keep the found piece for yourself. **Later, however, it turns out that the jewelry is fake and worthless and that you gave a large sum of money to a fraudster.**	In many countries of the world, local residents earn their living by pseudo fortune-telling with twigs, valuable personal items, or money. Be vigilant. Do not be fooled by such cheap tricks even if the false fortune-teller tries to convince you that they can save you from financial collapse or something worse. **Most likely, she will simply steal your money or valuables, skillfully substituting them with fakes**. And in the case of the twigs and other nonsense, she will simply try to get as much money out of you as possible for her very dubious services.	If you visit a different city or a foreign country, it is advisable to book a transfer in advance with a trusted company. But if this has not been prearranged, **use public transport (metro, trams, buses), or book a taxi with a large company that tells you the fees before the trip.** Do not trust private cabbies as they can overcharge or deliberately take you on the longest possible route.	Nevertheless, you should keep your guard up on all forms of private and public transport. The driver might give you your change in the local currency, and it can be worth a lot less than the dollars or euros that you paid with. Be careful.

5- Dishonest seller

Such crooks can be found in small shops or street tents with souvenirs. The dishonest seller will be doing multiple things at the same time: talking on the phone, bargaining with 3 customers, taking money, and giving change. **And in the process, he can write down your credit card information** if you decide to pay with a card. Or he'll just give you less change, hoping that he managed to distract you.

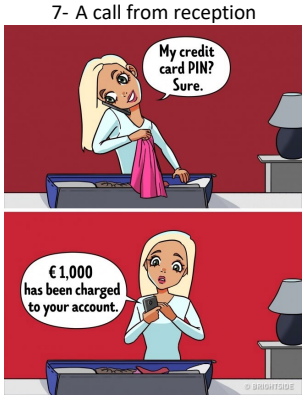
Another situation is possible: when one vendor captures your attention with a conversation, the other charges you and either finds out your card information or gives less change than he has to.

6- Unofficial ticket sales

When you buy tickets to the theatre, to a concert, to a museum, or anywhere else from street vendors, **there is a good chance that the ticket will turn out to be fake**. Even if the ticket is real, you might pay a much higher price for it than you would if you bought it officially.

It is better to purchase tickets at official ticket offices or from the website of the company.

7- A call from reception



When you stay at a hotel, someone might call your room and, after introducing themselves as a member of the guest service team, ask for your credit card details to pay for services.

Never give this information to anyone. If you are insistently pushed to do so, tell them that you don't mind going down to reception and paying for the services in person. Most likely, the scammer will back off.

It is also worth noting that in decent hotels payment is made at the time of departure when you hand over the room keys.

8- Rental housing

Sometimes tourists prefer not to stay in hotels but to rent apartments or even entire houses. There is a high probability that the owner of the property will try to make you pay a huge additional charge for "damaging property."

To prevent this, sign an agreement with the owner of the apartment before moving in, which will stipulate all the information about damage to property and any additional fees.

9- New friends

This method is very common in warmer countries where tourists live not in the main building of the hotel but in bungalows. Scammers get acquainted with you, and you have fun together for a few evenings. Then one day they will find out the exact time you will be away from your room, and they break in. As a result, you risk losing all your money and valuables.

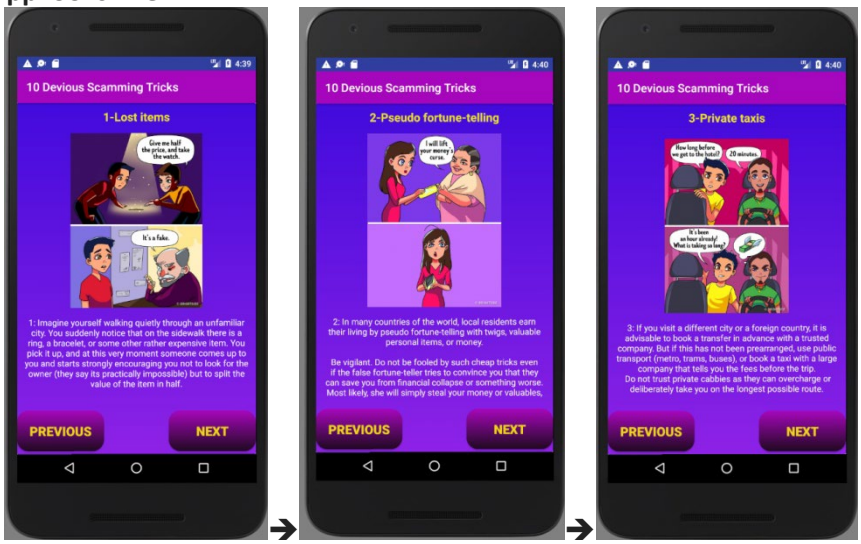
To avoid this, you should choose hotel rooms or bungalows that have a safe in them. And try not to reveal too much information about your vacation plans to unfamiliar people.

10- Asking to make a call

A very frightened or puzzled person may approach you on the street. He will ask you to let him make a call because he lost his phone, or he might tell you that he was robbed. As a result, the call will cost you a fortune. The swindler will simply call a prepared number, and you will be charged a ton of money for this "conversation." And this money is most likely going to the thief's account.

To save your money, use plans that do not allow you to have a negative balance. If you lose or lend your phone, you lose the funds that were on your account, but at least you will not have a huge debt to deal with.

Here is how the final app looks like:



HINT:

1. **Download, copy and paste 10 trick illustration images into "drawable" folder;**
2. **The TextView to display caption should be placed inside ScrollView because the caption is too long and we need to use scroll view to contain it;**

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <!--TextView: display a long caption-->
    <TextView
        android:id="@+id/caption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:gravity="center"
        android:padding="5dp"
        android:scrollbars="vertical"
        android:text="Imagine yourself walking quietly through an unfamiliar city. You suddenly notice
that on the sidewalk there is a ring, a bracelet, or some other rather expensive item. You pick it up,
and at this very moment someone comes up to you and starts strongly encouraging you not to look for the
owner (they say it's practically impossible) but to split the value of the item in half.."
        android:textColor="#FFFFFFFF"
        android:textSize="15sp" />
</ScrollView>
```

3. **Inside strings.xml file, declare 2 arrays: an array of 10 captions, called caption_array, and one more array of 10 trick titles, called trick_title_array:**

```
<!--Declare an array of 10 trick titles-->
<string-array name="trick_title_array">
    <!--Trick title 1-->
```

```
    <item>1-Lost items</item>
    <!--Trick title 2-->
    <item>2-Pseudo fortune-telling</item>
    <!--Trick title 3-->
    <item>3-Private taxis</item>
    <!--Trick title 4-->
    <item>4-Fares</item>
    <!--Trick title 5-->
    <item>5-Dishonest seller</item>
    <!--Trick title 6-->
    <item>6-Unofficial ticket sales</item>
    <!--Trick title 7-->
    <item>7-A call from reception</item>
    <!--Trick title 8-->
    <item>8-Rental housing</item>
    <!--Trick title 9-->
    <item>9-New friends</item>
    <!--Trick title 10-->
    <item>10-Asking to make a call</item>
</string-array>

<!--Declare an array of 10 captions-->
<string-array name="caption_array">
    <!--Caption 1-->
    <item>.....</item>
    <!--Caption 2-->
    <item>.....</item>
    <!--Caption 3-->
    <item>.....</item>
    <!--Caption 4-->
    <item>.....</item>
    <!--Caption 5-->
    <item>.....</item>
    <!--Caption 6-->
    <item>.....</item>
    <!--Caption 7-->
    <item>.....</item>
    <!--Caption 8-->
    <item>.....</item>
    <!--Caption 9-->
    <item>.....</item>
    <!--Caption 10-->
    <item>.....</item>
</string-array>
```

4. **Modify a little bit java code in MainActivity.java file:**
   - **array length now is equal to 10 instead of 12:**

```
//Declare an array of 10 "illustration" images, type: integer (int)
private int[] images = {R.drawable.trick1, R.drawable.trick2, R.drawable.trick3,
        R.drawable.trick4, R.drawable.trick5, R.drawable.trick6, R.drawable.trick7,
        R.drawable.trick8, R.drawable.trick9, R.drawable.trick10};
//Declare an array of 10 captions, type: String
private String[] caption_array = new String[10];
//Declare an array of 10 trick title, type: String
private String[] trick_title_array = new String[10];
```

   - **you display image, caption and trick title at same time:**

```
//Display image, title & caption
trickTitle.setText(trick_title_array[index]);
imageView.setImageResource(images[index]);
captionText.setText((index + 1) + ": " + caption_array[index]);
```

Experiment:

Exercise 1: Use VideoClip element to display video clips

Convert the below article "**11 of the funniest video clips of all time**" into an Android Mobile App:
Link: http://www.telegraph.co.uk/news/newstopics/howaboutthat/12187754/17-of-the-funniest-GIFs-of-all-time.html

1. 2 Birds 1 Almond Link: https://imgur.com/gallery/4az5Vt0	2. SantaCat can't get the Santa Hat Link: https://imgur.com/gallery/KZf5kWZ	5. Insert card as shown Link: https://imgur.com/gallery/zEH1zQV	6. ATM disaster Link: http://giant.gfycat.com/WatchfulAlarmedHorseshoecrab.gif
7. Nelly the Calm Owl Link: https://imgur.com/gallery/IEl6yJl	8. When Life Hits You Link: https://imgur.com/gallery/N2Ip9lw	9. Camouflage Link: https://imgur.com/gallery/FddlLSN	10. High Five Link: http://i.imgur.com/euLq22x.gif
11. Garbage truck Link: https://giphy.com/gifs/good-garbage-dun-Y5LkIG7z8l6YE?utm_source=iframe&utm_medium=embed&utm_campaign=Embeds&utm_term=http%3A%2F%2Fwww.telegraph.co.uk%2Fnews%2Fnewstopics%2Fhowaboutthat%2F12187754%2F17-of-the-funniest-GIFs-of-all-time.html	12. Bernie Sanders Link: https://giphy.com/gifs/justin-bernie-sanders-socialism-startle-l2JI29ccohFCPowxi?utm_source=iframe&utm_medium=embed&utm_campaign=Embeds&utm_term=http%3A%2F%2Fwww.telegraph.co.uk%2Fnews%2Fnewstopics%2Fhowaboutthat%2F12187754%2F17-of-the-funniest-GIFs-of-all-time.html	13. Robot fail Link: https://giphy.com/gifs/robot-garbage-truck-l7kkegrRyNrk4?utm_source=iframe&utm_medium=embed&utm_campaign=Embeds&utm_term=http%3A%2F%2Fwww.telegraph.co.uk%2Fnews%2Fnewstopics%2Fhowaboutthat%2F12187754%2F17-of-the-funniest-GIFs-of-all-time.html	14. Slinky kitten Link: https://i.imgur.com/btkuyR1.mp4
15. One of these owls is a jerk Link: https://i.imgur.com/f4HjAgD.mp4	16. Horse head feeder Link: https://i.imgur.com/Yuv8m82.mp4	17. High jump Link: https://giphy.com/gifs/fail-high-jump-kDSi9szKYImSQ?utm_source=iframe&utm_medium=embed&utm_campaign=Embeds&utm_term=http%3A%2F%2Fwww.telegraph.co.uk%2Fnews%2Fnewstopics%2Fhowaboutthat%2F12187754%2F17-of-the-funniest-GIFs-of-all-time.html	

Hint: In order to display video clips, we use "VideoView" element.

+ Add VideoView to the layout:

```
<VideoView
    android:layout_width="300dp"
    android:layout_height="300dp"
    android:id="@+id/videoView"/>
```

+ Prepare video clip:

Copy the video clips (example **test.mp4**) into "**raw**" folder. If the "**raw**" folder doesn't exisit already, add it the project.

*Issue: the main thing to consider here is the **size of your video**. As video files can be quite large, your resulting **apk file can also get unacceptably large**. Personally, I would rarely want to download an app from the market that weighs in at 100MB.*

+ Java code:

```
//Find references and do casting for VideoView
videoView = (VideoView) findViewById(R.id.videoView);

//Display video clip (mp4 format) on VideoView
Uri uri = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.test);
videoView.setVideoURI(uri);
videoView.start();
```

Exercise 2: user WebView to display gif image

Develop an app to display a collection of **funny cartoon animated gifs**:
Link: https://www.buzzfeed.com/ryanhatesthis/the-54-best-animated-gifs-of-2012?utm_term=.pcjElGYej#.tv14bq8PN

(Or link: https://giphy.com/explore/funny-cartoon)

HINT: In order to display gif images, we use "WebView" element

+ Add a Webview to Layout:

```
<WebView
    android:id="@+id/webView"
    android:layout_width="300dp"
    android:layout_height="300dp"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_weight="1"
    android:adjustViewBounds="true"
    android:scaleType="fitCenter" />
```

+ Copy the gif images into "Asset" folder:

Copy a gif image (example **test.gif**) into "**raw**" folder. If the "**asset**" folder doesn't exisit already, add it the project.

+ Load gif image into webView:

```
////////////////////////////////////////////////////////////////////////////////////////
//Animation ImageView: webView
webView = (WebView) findViewById(R.id.webView);
webView.setWebViewClient(new WebViewClient());
webView.getSettings().setJavaScriptEnabled(true);
webView.getSettings().setLoadWithOverviewMode(true);
webView.getSettings().setUseWideViewPort(true);
webView.loadUrl("file:///android_asset/test.gif");
```

PART 5: "How Long Does It Take to Make an App? 5 Phases of Mobile App Development Lifecycle" App
(Self-directed learning)

Your task is to convert the below article "How Long Does It Take to Make an App? 5 Phases of Mobile App Development Lifecycle" into an Android Mobile App:
Link: https://lvivity.com/5-phases-mobile-app-development-lifecycle

How Long Does It Take to Make an App? 5 Phases of Mobile App Development Lifecycle

We often come across clients with great ideas for a new app, and they wish to have the app developed and got into the market as soon as possible. But the truth is that even when using lean software development approach or Agile methodologies, it still takes time and money to build an app.
While various factors come into play when building a mobile app, it takes a minimum of three months to make a mobile app's initial version, and it takes more than six months to build apps with advanced functionality and more features.

To get a rough idea of how long building a mobile app takes let us take a more thorough look at the process of production from start to finish.

1. The Discovery Phase

All apps begin with an idea. It is this idea that is usually refined into a firm basis for a mobile application.
The discovery or inception phase is about refining the idea for the app. To build a successful product, it's critical that you ask yourself some fundamental questions. The following are some of the factors you need to consider before you publish your app in any app store.
Competitive Advantage: does the market have similar apps already? if the answer is yes, in what way does your app differentiate from those?
Infrastructure Integration: what are the existing infrastructures that the app will integrate or extend? Also, apps need to be examined in the context of mobile form factor:

- Form/Mobility- how will the app work in a mobile form factor? In what ways can I add value using technologies like camera, location awareness, etc.
- Value: what value will the users get from this app? How are they going to use it?

To help with functionality and design of an app, you may want to define Use Cases and Actors. Actors refer to roles with an app and are users in most cases. Use cases, on the other hand, are intents or actions.
For example, a task tracking app may have two Actors who could be User and Friend. A user could Create and Share a Task with a Friend. In such a situation, creating and sharing a task are two distinctively different use cases which, along with the Actors, will guide you on the screens you have to build, and the kind of logic and business entities you will need to develop.

Once a suitable number of actors and use cases has been captured, it becomes much easier to start the process of designing an app.

2. The Design Phase

At this stage, you already know the kind of an app you want to build. You've put all the nuts and bolts you need. Now is the time to start system development.

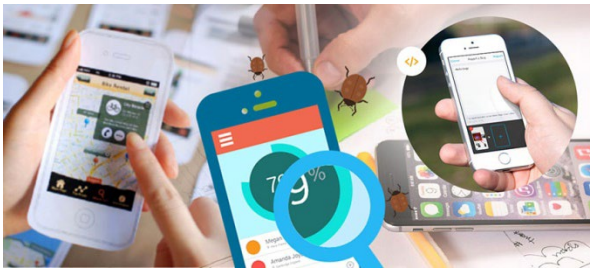
Start by assembling your dream team since it knows all the project requirements. Create a workflow design and chart for the project to set boundaries for certain responsibilities. The role of the team, core responsibilities, and functions are spelled out and documented in this stage.

This will save a significant amount of time and project costs as the development proceeds. It is also during the development stage that personnel are trained with regards to any extra requirements of the project.

The rule of thumb is that mobile developers should first make a lightweight prototype to determine whether it is functional and feasible or not. The prototype is also important in seeing if the project has access to required technologies that will achieve the app's aims from the start.

Once approvals on the tasks assigned to relevant departments are obtained, it's time to go to the next stage.

3. The Development and Testing Phase

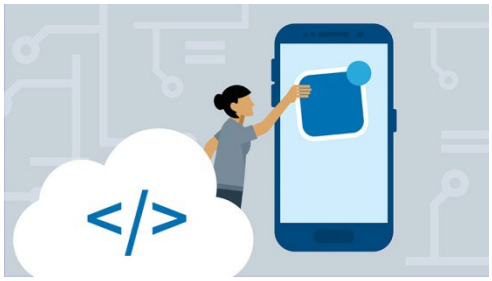


After successfully dispensing with the design stage, it is now time for software development guys to start working on the project. The development process is cyclical and iterative. It goes hand in hand with testing so that any bugs or errors in the program can be caught and rectified at the earliest opportunity.

This is also the time to perform regular reviews to keep tabs on the project's progress and to ensure that it proceeds like clockwork.

After completing the product, it has to go through further testing to ensure that all the components of the application work in harmony and unison.

4. The Deployment Phase



At this point, the app is ready to be released to the world after undergoing rigorous development and testing. It's at this point that you need the marketing and advertising team to help popularize the app. The official release date for the app should represent the climax of the application marketing efforts to this time.

The marketing team should create buzz around the app with articles and write-ups by influential journalists and bloggers. The team also needs to announce to anyone who showed interest in the app before its launch. Another way to promote the release is via email blast and mentions on your social media channels. The rationale is to get ratings and downloads, as well as to create some momentum.

A robust marketing campaign is just as important as the app itself. Therefore, make sure that you never underestimate the marketing bit of the process.

5. Maintenance and Updates Phase

The lifecycle of the application development does not end with releasing your product. It's important that after your product gets in the hands of the users, you continue providing necessary updates and maintenance.
This is important because it avoids a situation where the app retention rate drops. The other name for the maintenance and updates phase is the post-development phase.

Conclusion

App development does not end at launch. Once your app is downloaded by users, there will be an outpouring of feedback which you will need to incorporate in future versions.
It might seem like a tough thing especially if for those developing an app for the first time. However, as one gets used to it, the development will be more predictable.

Self-directed learning
READING

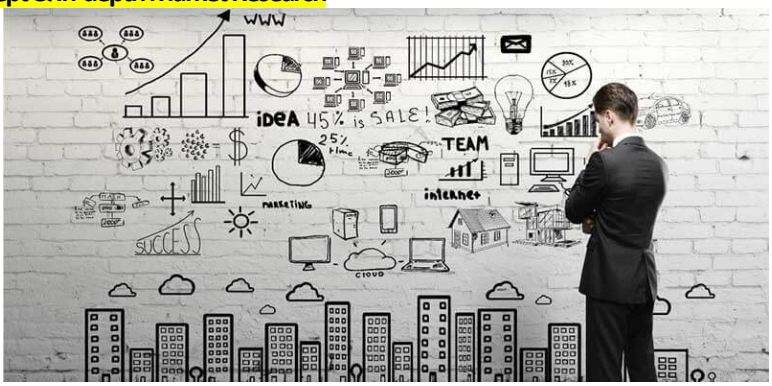
An Overview of Different Stages of Mobile App Development Lifecycle

June 5, 2017 by Mehul Rajput
Source: https://www.mindinventory.com/blog/an-overview-of-different-stages-of-mobile-app-development-lifecycle/

Based on one survey smart phone users spend their 90% of mobile time on apps. Fortune 500 companies generate 42% of all mobile sales as well as average time people spend on their mobile app is also increased by 21%. Many other surveys predict that the era of mobile app is growing swiftly with the development of more innovative apps.

When looking for developing a new mobile app, one should make a foolproof strategy to make the app development lifecycle proficient and develop a popular mobile app. From identifying unique app concept to its successful implementation, a mobile app development company requires comprehending the recent trends, tools and technology to market an app using winning strategy. Let's take a look at the different stages of mobile app development lifecycle:

1. Comprehend the App Concept & In-depth Market Research



Finding an innovative and unique concept for a new app leads you to make it popular and successful. The first phase of mobile app development is very important as it is associated with your target market. There are several aspects that require discovering in this first stage of app lifecycle. You need to make a list of questions and find answers to get a definite outcome.

- First of all decide the purpose of your app.
- Know what achievement you are expecting from your app?
- What are your target markets/end users?
- List out your key competitors, what best they offer to users/what users expecting more from their app? How best your app stays competitive to defeat your rivals or how you can make your app unique compared to your rivals?
- Selection of right mobile app platform
- How much time app development requires? When will you expect to launch your app?
- What is your app budget?
- Will you choose in-house app development or hire a mobile app development company?
- Select a team of professionals work on the project as well as assigning work to team members based on their proficiency.
- Which factors you will consider making your app marketing successful and boost conversion?

You need to make profound research and analysis to get answers of all such questions. Without detail analysis of all such aspects, you should not move ahead to start your development process.

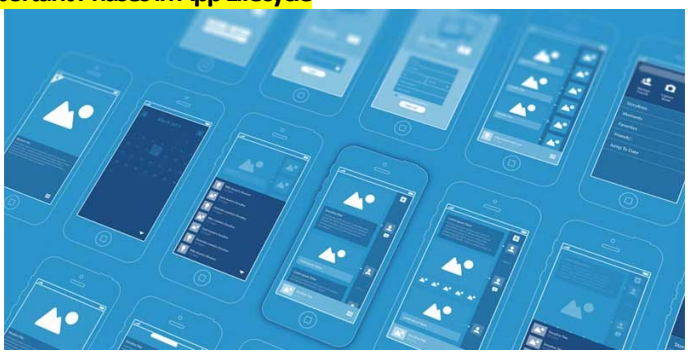
2. Set up Key Objectives of Your App

This is the planning phase of your app lifecycle, which is very significant. You need to make decisions about all features and functionalities require in your app.

- Also make sure that your app is capable to accomplish your end-users demands
- What features your app proffers to make it unique?
- What factors make your app more advanced?

Without offering your users bunch of common features, bestow some out-of-the-box features attract them to use your app. When you develop an app, always consider the recent technology trends helps in making an app quickly such as Kotlin – Latest Android Programming Language. It helps proffering excellent user experience to your target audience.

3. Wireframes & Prototyping – Important Phases in App Lifecycle



This stage is very important as it gives you a rough idea about the look & feel of your app. In this phase, you need to ensure key features you wish to include in your app.

There are several things you need to consider before start wireframing such as comprehending your key objectives, screen orientation, better representation of content, use of well- organizing content, use of blocks, and many more help you in making your wireframe effective.

With the help of foolproof wireframe, you can make your designing and development process more efficient and fast. You should also focus on the opportunities to include in your project as well as factors that enhance user experience.

Wireframe helps you to guide for the backend structure that you require supporting the app such as APIs, servers, data diagram, data integrations as well as other services such as push notifications. Any technical limitation found in the backend development process can be overcome in this initial phase.
App Prototyping helps you in making the development process easier as it offers an opportunity to evaluate design ideas, collect feedbacks, realize dead links, and focus on several other important aspects. You can use online prototyping tools helps to make your app development process easier.

Read also: ***The Difference Between Wireframe, Mockup and Prototype***

4. Developing the App – Adopt an Agile Methodology

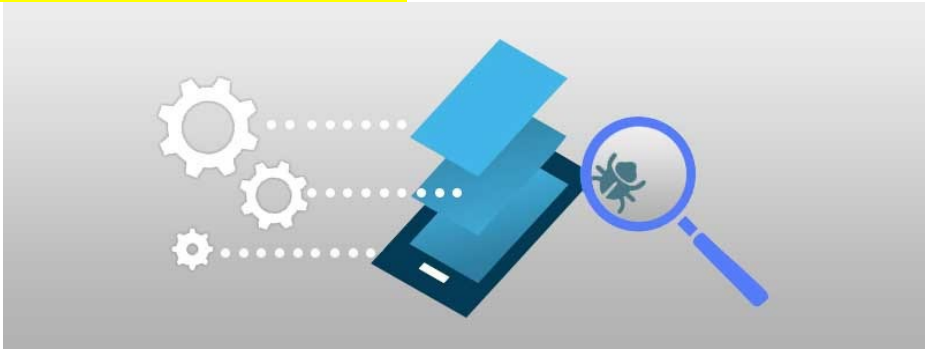
It involves several processes. If you are not using backend as a service (BaaS) provider or an app development platform, you need to setup the databases, storage solutions, servers and APIs for your app backend.

You should select an Agile Methodology for mobile app development as it enables to make the changes, and implement new features efficiently as well as keep evolving with the need of changing trends.

You require setup an account for a developer in the app store that you choose for your app launch. It requires some days to get started hence, if you don't have a developer account, take the steps accordingly. You should consider app store guidelines before starting development to prevent your app from rejection by the app store.
In this stage, your designers will make the skins for your app that let you understand the look of your app. These actual screens represent the interface that your users finally utilize to make interaction with your app. Therefore, in this stage, you should ensure that the design includes all the feedback and ideas as defined in early phase. In this stage, UI/UX details make the difference as it impacts the appearance and functionality of an app that users have to experience.

Read also: ***Top Things To Consider Before Developing Mobile Apps***

5. App Testing & Debugging – Require Foolproof Methodology



This is one of the most important phases in order to make your app seamless. Ensure different testing such as usability testing, interface checking, compatibility, service, as well as performance, security checks, operational testing, and several more. Building well-planned testing structure helps you to reduce your time and money in the development process.

6. Launch the App

When you found your app ready for launch, you can submit it in the app store or the play store. It takes time to get approval from the app store around 4-6 days, while Android apps need 3 hours to get approval.

App Store Optimization and App Marketing

Once your app gets approval, start focusing on app store optimization and app marketing to create awareness of your app within your defined target audience.

Make research to find targeted keywords to use in your app title, description before you submit your app. With effective landing page that comprises app name, icon, screenshots, promotional videos, contact information, link to download app and other information helps in promoting your app efficiently. Remember, your app marketing and promotion is equally important as its development in order to enhance its reach and explore potential markets.

Conclusion

In order to make mobile app development process efficient and trouble-free, app developers should comprehend the app development lifecycle precisely. It helps you in many ways to make your app seamless, feature-rich and functional compared to your rivals apps.

Efficiently developed app reduces chances of rejection by the app store. From discovering concept to app launch, every phase has its own importance in the app development lifecycle; hence, mobile app developers need to comprehend the importance of each phase before initiating further.

Activity

Source: https://developer.android.com/reference/android/app/Activity.html

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with windowIsFloating set) or embedded inside of another activity (using ActivityGroup).

There are two methods almost all subclasses of Activity will implement:

- onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI, and using findViewById(int) to retrieve the widgets in that UI that you need to interact with programmatically.
- onPause() is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the ContentProvider holding the data).

To be of use with Context.startActivity(), all activity classes must have a corresponding <activity> declaration in their package's AndroidManifest.xml.

Activity Lifecycle

Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

- If an activity is in the foreground of the screen (at the top of the stack), it is *active* or *running*.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is *paused*. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

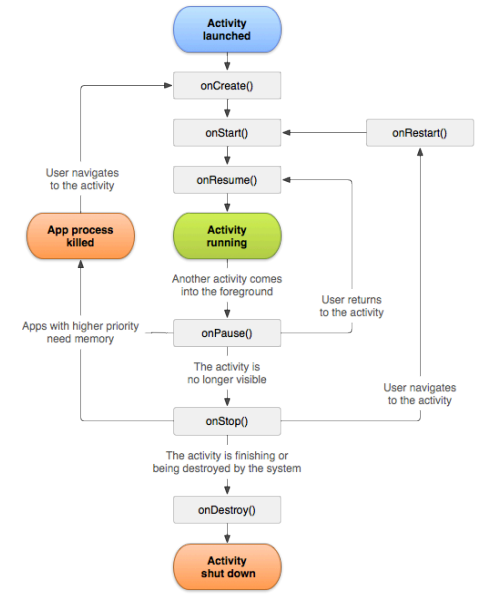
The entire lifecycle of an activity is defined by the following Activity methods. All of these are hooks that you can override to do appropriate work when the activity changes state. All activities will implement onCreate(Bundle) to do their initial setup; many will also implement onPause() to commit changes to data and otherwise prepare to stop interacting with the user. You should always call up to your superclass when implementing these methods.

```
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();
}
```

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.

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(). For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread 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. For example, you can register a BroadcastReceiver in onStart() to monitor for changes that impact your UI, and unregister it in onStop() when the user no longer sees what you are displaying. The onStart() and onStop() methods can be called multiple times, as the activity becomes visible and hidden 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.



Configuration Changes

If the configuration of the device (as defined by the Resources.Configuration class) changes, then anything displaying a user interface will need to update to match that configuration. Because Activity is the primary mechanism for interacting with the user, it includes special support for handling configuration changes.

Unless you specify otherwise, a configuration change (such as a change in screen orientation, language, input devices, etc) will cause your current activity to be *destroyed*, going through the normal activity lifecycle process of onPause(), onStop(), and onDestroy() as appropriate. If the activity had been in the foreground or visible to the user, once onDestroy() is called in that instance then a new instance of the activity will be created, with whatever savedInstanceState the previous instance had generated from onSaveInstanceState(Bundle).

This is done because any application resource, including layout files, can change based on any configuration value. Thus the only safe way to handle a configuration change is to re-retrieve all resources, including layouts, drawables, and strings. Because activities must already know how to save their state and re-create themselves from that state, this is a convenient way to have an activity restart itself with a new configuration.

In some special cases, you may want to bypass restarting of your activity based on one or more types of configuration changes. This is done with the android:configChanges attribute in its manifest. For any types of configuration changes you say that you handle there, you will receive a call to your current activity's onConfigurationChanged(Configuration) method instead of being restarted. If a configuration change involves any that you do not handle, however, the activity will still be restarted and onConfigurationChanged(Configuration) will not be called.

Debugging you app with Android Virtual Devices (AVD) & Logcat (DDMS)

1. Define test cases for your application

A test case, in software engineering, is a **set of conditions** under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do. The mechanism for determining whether a software program or system has passed or failed such a test is known as a **test oracle**. In some settings, **an oracle could be a requirement or use case**, while in others it could be a **heuristic**.

It may take **many test cases** to determine that a software program or system is considered sufficiently scrutinized to be released.

2. Use Dalvik Debug Monitor Server (DDMS) and Logcat

+ Dalvik Debug Monitor Server (DDMS):

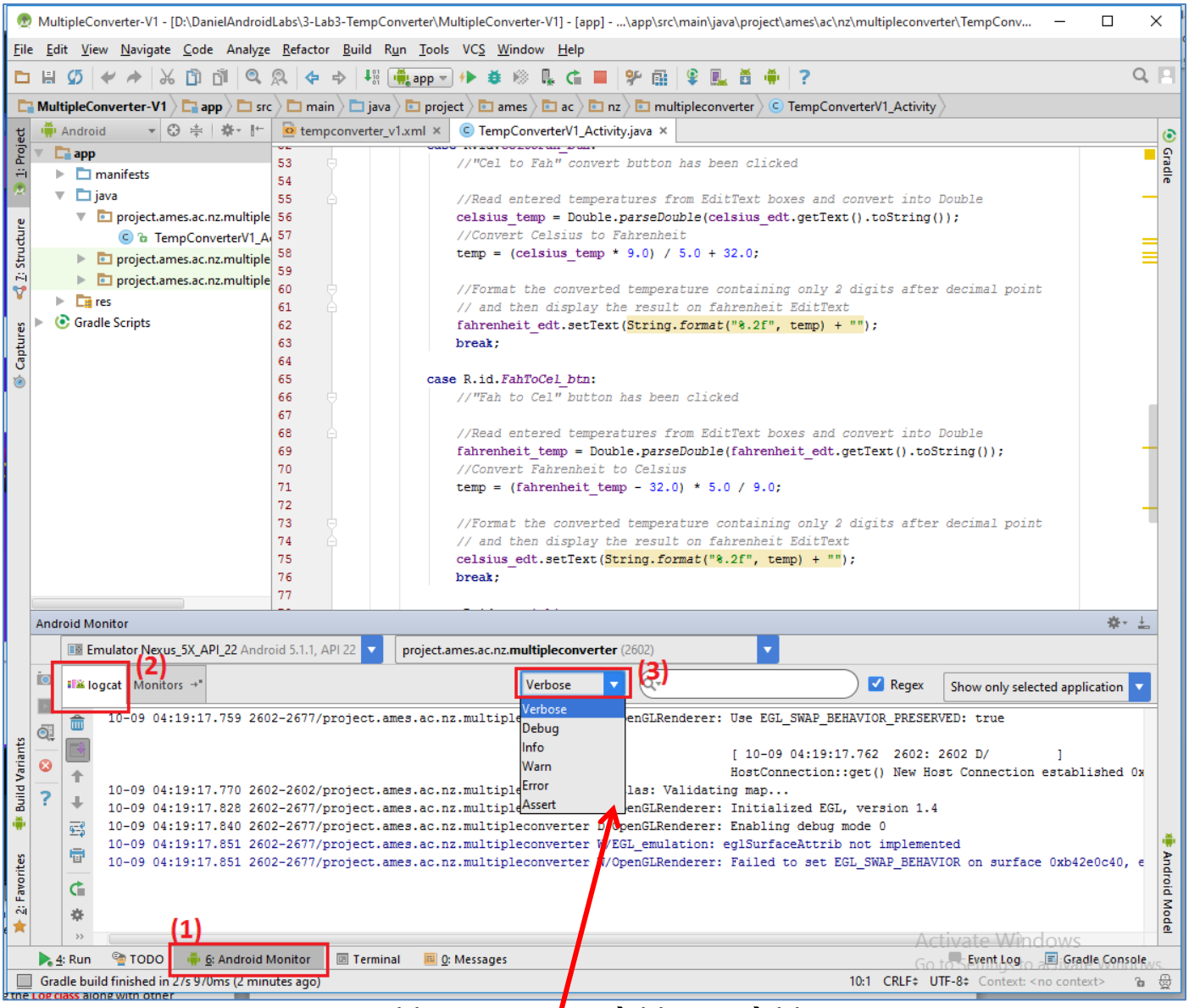
Android Studio includes a **debugging tool called the Dalvik Debug Monitor Server (DDMS),** which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more.
For more information on accessing logcat from DDMS, instead of the command line, see Using DDMS at:
http://android.magicer.xyz/tools/debugging/ddms.html

The **Android logging system** provides a mechanism for **collecting and viewing system debug output**. Logs from various applications and portions of the system are collected in a series of circular buffers, **which then can be viewed and filtered by the logcat command.** You can use logcat from an ADB shell to view the log messages.

+ Using Logcat:

LogCat is integrated into DDMS, and outputs the messages that you print out using the **Log class** along with other system messages such as stack traces when exceptions are thrown.
In order to view Logcat window, follow the steps:

(1) Click **"Android Monitor**" at the bottom bar of the Android Studio IDE;
(2) Choose "**logcat**" tab;
(3) Select filter "**Verbose**"



Logcat monitor: (1) Android Monitor → (2) Logcat → (3) Filter: Verbose

Log is a logging class that you can utilize in your code to print out messages to the LogCat. Common logging methods include:

- v(String, String) (verbose)
- d(String, String) (debug)
- i(String, String) (information)
- w(String, String) (warning)
- e(String, String) (error)

Logging messages on LogCat

For example:

```
Log.i("MyActivity", "Get item number " + position);
```

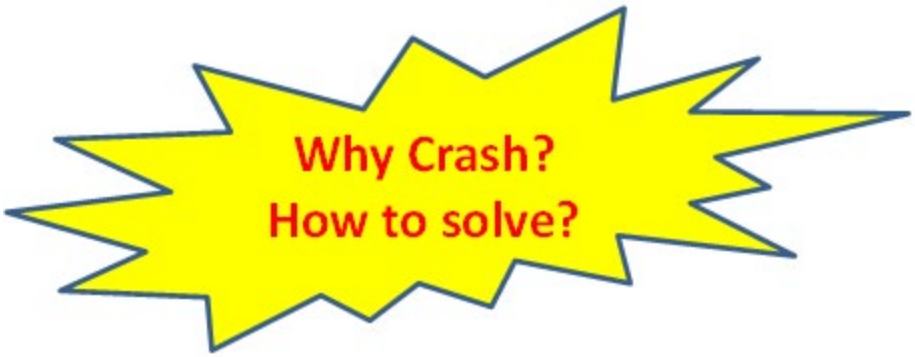
The LogCat will then output something like:

```
I/MyActivity(1557): Get item number 1
```

App crash

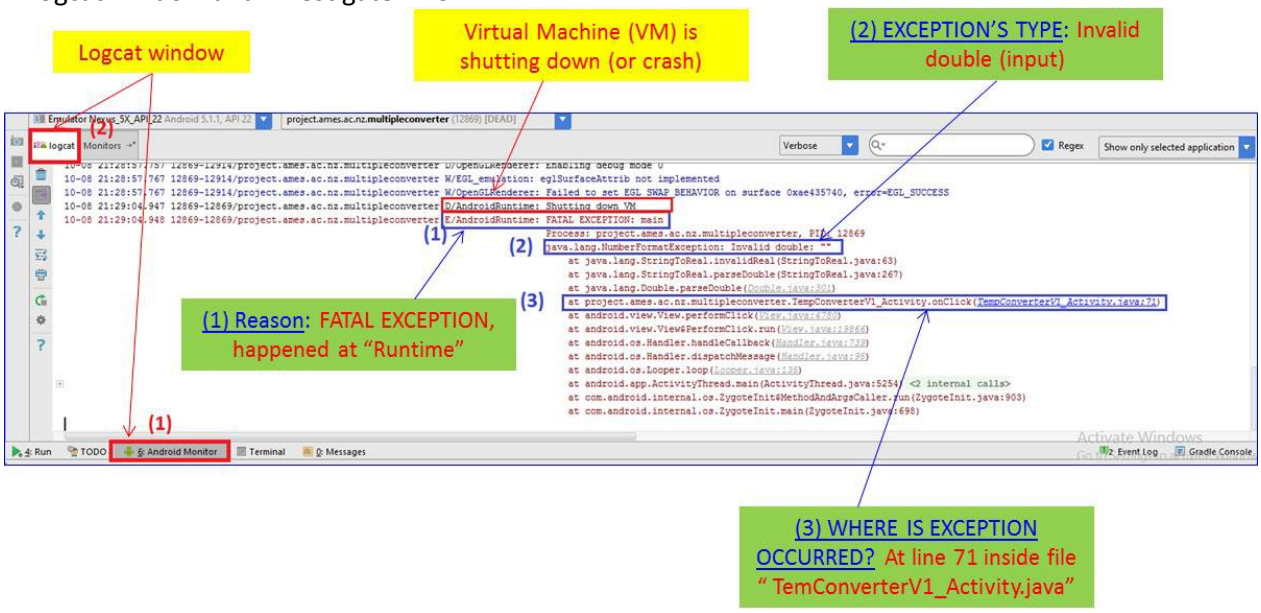
What's Exception? How to catch & handle Exception

Read more: Java – Exceptions: https://www.tutorialspoint.com/java/java_exceptions.htm



1. Why? Crash due to Exceptions

+ Open Logcat window and investigate Error:



Logcat window: show the error why the application is crashed.

2. What's Exception?

An **exception** (or **exceptional event**) is a problem that arises during the execution of a program. When an **Exception** occurs, the normal flow of the program is disrupted and the program/Application **terminates abnormally**, which is not recommended, therefore, **these exceptions are to be handled**.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs:

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Based on these, we have **three categories of Exceptions**. You need to understand them to know how exception handling works in Java.

1. **Checked exceptions** − A checked exception is an exception that occurs at the compile time, these are also called as **compile time exceptions**. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions. For example, if you use FileReader class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a FileNotFoundException occurs, and the compiler prompts the programmer to handle the exception.
2. **Unchecked exceptions** − An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation. For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an ArrayIndexOutOfBoundsExceptionexception occurs.
3. **Exception Hierarchy -** All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class. Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example: JVM is out of memory. Normally, programs cannot recover from errors.

3. How to catch "Exceptions" to avoid CRASH? – TRY … CATCH

A method **catches an exception** using a combination of the **try** and **catch keywords**. A **try/catch block** is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following:

Syntax:

```
try {
    //Try block:
    //Protect codes that possibly generate an exception
}
catch (ExceptionName e) {
  //Catch block:
  //statements to handle any exceptions
}
```

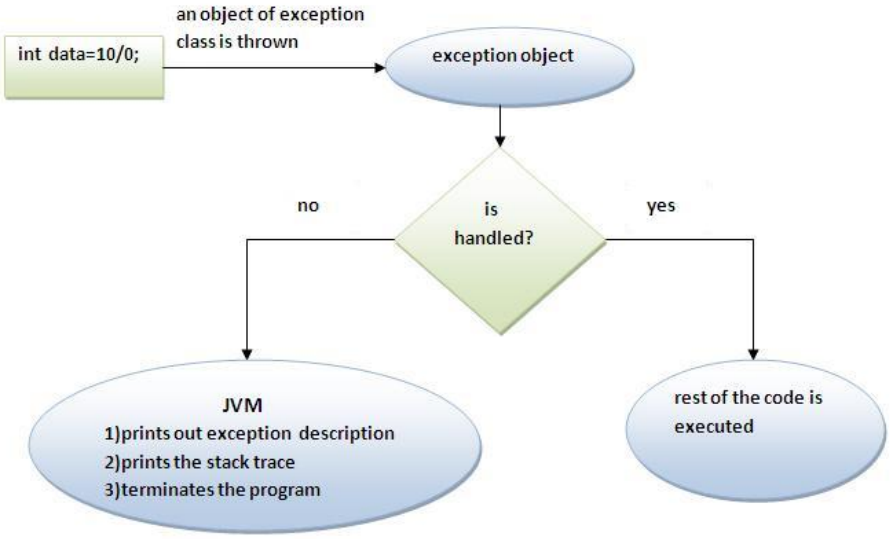
The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the **catch block (or blocks)** that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

Internal working of java try-catch block: The JVM firstly checks whether the exception is handled or not. If exception is not handled, JVM provides a default exception handler that performs the following tasks:

- Prints out exception description.
- Prints the stack trace (Hierarchy of methods where the exception occurred).
- Causes the program to terminate.

But if exception is handled by the application programmer, normal flow of the application is maintained i.e. rest of the code is executed.



Internal working of java try-catch block: http://www.javatpoint.com/try-catch-block

4. Read more:

1. Android Resources Organizing & Accessing: https://www.tutorialspoint.com/android/android_resources.htm
2. Event handling: https://www.tutorialspoint.com/android/android_event_handling.htm
3. Java – Exceptions: https://www.tutorialspoint.com/java/java_exceptions.htm
4. DDMS: Logcat & Logging: https://developer.android.com/studio/profile/ddms.html
5. Logcat Command-line tool: https://developer.android.com/studio/command-line/logcat.html