Introduction to Android Development

Android is one of the most popular operating systems with over 88% of phones running Android.

XML & Java Code
An XML file is used to build layouts in Android. Layouts are static pages (screens) which can be manipulated using Java code (programmatically).
Declaring UIs in XML enables us to better separate the presentation of our application from the code that controls its behaviour.

Installing Android Studio
Android Studio is an IDE that makes it easy for us to write & build Android applications very conveniently.
We can create emulators (virtual Android phones), see previews and build UI layouts using drag & drop features.

You can go to https://developer.android.com/studio to download Android studio.

The Layout Editor
The layout editor is used to quickly build layouts by dragging UI elements which is easier to write xml by hand.
We can setup different attributes easily using the design mode in layout editor.
Chapter 1 - Creating our first App

In this chapter we will create our first Android app. I don't expect you to know anything but will walk you through the steps to build your first Android App!

What is an APK?
An APK is a collection of different files (like code, audio, video, etc.) compiled and bundled into a single file.

What is an AVD?
AVD stands for Android Virtual Device.
AVD is an emulator configuration that simulates a physical Android device.

Android UI layouts
View are the base class for widgets
View = Basic building blocks
View Group holds View and View Group
(Container)

Just like a box can hold objects and more boxes

Diagram:
```
View Group
  ↓
View Group  View  View
  ↓
View  View
```
XML vs Java in Android

XML is the skeleton code which describes the UI layout. Java drives this XML.

Views

When submit is clicked:
- change the screen and store the data
- Java

Important Notes

- Android Studio is a resource hungry program. You need to have patience while using it!
- Sometimes Android Studio might download files from the internet. So keep your WiFi/Hotspot ready.
- In very rare cases, your firewall might block Android Studio. In that case, you will need to...
- If your computer is slow, use USB debugging to use your phone as an AVD replacement.
- If you are using an old PC, make sure that the virtualization is turned on.

Creating our Unit Converter App.

Step 1: Click on create a new project
Step 2: Click on Empty activity

Step 3: Click Next

Step 4: Type the name of the app & click Finish

You can change Apk location from here
The R-Java file

Android R-Java file contains resource IDs for all the resources. We can use it to access views from our Java file.

```java
button = findViewById(R.id.mybutton);
```

- function to get view
- button in XML has an ID of "my-button"

Adding Event Listeners

We can add click listeners by using `setOnCheckedChangeListener` method as follows:

```java
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Action Here
    }
});
```

This is performed when the button is clicked.
**android : onClick attribute**

The `onClick` attribute can be set for the `button` element in XML layout.

```
android: onClick = "sendMessage"    → In XML
```

```
public void sendMessage(View view) {
    // code here
}
```

→ In Java
Chaper 1 - Practice Set

1. Create an Android app which is capable of displaying "Good morning" to the user.

2. Create an App which is capable of adding two numbers entered by the user.

3. Create an App which displays the multiplication table of a given number.
Chapter 2 - Java Refresher

Java is an amazing object oriented programming language. We will use Java to create Android apps.

The main method

The Java program starts executing from here

```java
public static void main(String[] args) {
  // code
}
```

Printing to the console

The following code prints "Hello World" to the console:

```java
System.out.println("Hello World");
```

Variables in Java

Variables are buckets in memory

- String actionNow; → String is used to store seq of characters
- int marks; → int is used to store numbers
- int value = 7;

Type → Name → Value

Comments

// This code will be ignored by the Java compiler
// This is a comment
/* This is a
   multiline comment */
Strings in Java

String = sequence of characters

String name = "Harry";

→ Strings are immutable & cannot be changed

Printing Strings

We can concatenate string like this:

System.out.println("Hello, " + name);

String methods

String name = "Harry";

name.length()
name.toLowerCase()
name.trim()

Other data types to store numbers

We can store numbers using:

- byte → -128 to 127
- short → \(-2^{15}/2\) to \(2^{15}/2 - 1\)
- int → \(-2^{31}/2\) to \(2^{31}/2 - 1\)
- float → used to store decimal values (4 bytes) [ex. 10.1f]
- long → \(-2^{63}/2\) to \(2^{63}/2 - 1\)
- double → decimal values of 8 bytes [ex. 7.88d]
Booleans
true or false values
boolean isGood = true;

Operators in Java
These are the types of operators in Java:

1. Arithmetic operators → a+b, a*b, a/b, a%b
2. Assignment operators → a=3, a+=3
3. Relational Operators → a<b, a>=3
4. Logical Operators → (a>3) && (b<7)
5. Unary Operators → a++, b--
6. Bitwise Operators → ~, <<, &,

If - else conditionals
We can use If - else to execute instructions when a condition is true

if (a>3) {
  // code
  3
} else if (a>1) {
  // code
  3
} else {
  // code
  3

We can use if inside an if, if inside an else, and booleans inside if (as conditions)
Loops in Java

1) for loop: The syntax of a for loop looks like this:
   
   ```java
   for (initialize; check boolean exp; update) {
       // code;
   }
   ```

2) while loop: The syntax looks like this:
   
   ```java
   while (boolean) {
       // code
   }
   ```
   → stops when boolean becomes false

3) do while loop: do-while loop is guaranteed to execute at least once.
   
   ```java
   do {
       // code
   } while (condition);
   ```
   → Note the Semicolon

Functions in Java

We can use functions to separate logic like this:

```java
public static void harrysMethod(int a) {
    // code
}
```
Object Oriented Programming

In OOPs, class is a blueprint for creating objects.

```
Class
  \-- Obj 1
  \-- Obj 2
  \-- Obj 3
```

Syntax:
```
public class Harry {
    public static void thisMethod() {
        // code
    }
}
```

The objects can be created like this:
```
Harry h = new Harry();
```

Inheritance in Java

In Java, we can create classes from another class like this:
```
public class Programmer extends Employee {
    private String language;
    // code
}
```

```
Employee \rightarrow Programmer
```
Arrays in Java
We can store collection of similar items in an Array

```
[ 7, 10, 11, 21, 88 ]
```

```java
int[] harry = { 1, 5, 9, 21, 3; 
System.out.println(harry[0]);
```

Java Collections Framework
The collections framework in Java allows us to enjoy features like resizable arrays.

Arraylist is a class inside collections framework for creating resizable arrays.

```java
ArrayList harry = new ArrayList();
```

For Arraylist & other collection methods, visit the java docs.

Iterating through Arraylists
We can iterate through Arraylists like this:

```java
for (object o : harry) 
    System.out.println(“object: “ + o);
```

Here harry is an ArrayList
How to view Java Docs

Java has a very beautiful documentation where all the details from the API are listed.

You can navigate to Google and search for the docs where you can further search for the Java API you are interested in.
Chapter 2 - Practice Set

1. Write a Java program to print multiplication table of a given number.

2. Write a Java program to add two numbers.

3. Write a Java program to find out day of the week given the number.

4. Write a Java program to calculate income tax on a given income considering Indian laws.

5. Write a Java program to sum first n even numbers using for loops.

6. Write a program to print the following pattern:

   * * * * *
   * * * * *
   * * * *
   * * *
   * *
Chapter 3 - Activities & Layouts

What is an Activity?
A single, focused thing that the user can do.
In layman's terms ⇒ Activity = Screen

Activity has layouts
⇒ Defines how views are displayed

Types of Layouts in Android
If you go to the palette, you will find a lot of layouts.
We can use any of these (LinearLayout, ConstraintLayout etc.)

Linear Layout
Arranges its collection of views in a straight horizontal or vertical row.

⇒ gravity attribute can be used for alignment

Horizontal

Vertical

Required attributes of a LinearLayout
→ Layout width ⇒ can be match_parent/wrap_content/some_value_dp
→ Layout height ⇒ can be match_parent/wrap_content/some_value_dp
→ Orientation ⇒ can be horizontal or vertical

Note ⇒ Never close your Emulator on every run when you change the app.
Emulator is a resource hungry program & will take time to launch.
RelativeLayout

Elements are aligned relative to each other
Attributes like android:layout_alignParentTop are used to declare positions of views.

The RelativeLayout is available in legacy tab in Palette

Legacy tab in Palette
Some View layouts in Android Studio are replaced with better counterparts.

ListView is replaced with RecyclerView
GridView is replaced with ConstraintLayout
TabHost is replaced with TabLayout
RelativeLayout is replaced with ConstraintLayout

Important Notes

→ Component view shows certain warning sometimes. Correcting these by following good practices is recommended.
→ You can resize the preview as per the device of your liking in Design view itself.
→ Android Studio uses Gradle as the build system.
→ Project structure of Android is very simple & straightforward.
→ You don’t have to remember all the attributes of every View

Strings .xml file
This file contains string resources in XML format

<string name="app">Harry </string>

<string name="username">Alice</string>
We can use `<b>` bold `</b>` and `<i>` Italic `</i>` tags inside the text in `strings.xml`. Other HTML Tags are ignored. Escape sequence characters like `\n` are also allowed.

ScrollView
ScrollView is a ViewGroup used to create Scrollable Views. It contains just one child which can be greater than the screen height and can be scrolled.

ScrollView offers many attributes which is used to customize it.

Horizontal ScrollView
It is very similar to ScrollView but instead of vertical scrolling, it provides functionality for horizontal scrolling.

Logging in Android
We can log a message to the console using:

```java
Log.d("Tag", "log this message");
```

Similarly we can use log.i for info, log.e for error etc.

Constraint Layout
Constraint layout allows us to position widgets by applying constraints.

Constraints work exactly like a spring!

Cornerers can be used to resize the widget.
We can add a baseline constraint on widgets inside the constraint layout by right clicking > Show Baseline & finally adding the baseline constraints.
Chapter 3 - Practice Set

1. Create a layout which looks like this:

   ![Diagram](image1.png)
   - Image Views
   - Text Views

2. Create a layout which looks like this:

   ![Diagram](image2.png)
   - Scrollable Images

3. Create a layout which shows a bulk text scrollable vertically.

   ![Diagram](image3.png)

4. Create a layout which looks like a contact us page of a company.
Chapter 4 - Multi-Screen Apps

An Activity = Screen

One activity in an app is specified as the "main" activity which is shown to the user when the app is launched.

Intent
Whenever a new activity starts, previous activity is stopped but the system preserves the activity in a stack. This way when a new activity starts, that new activity is pushed onto the back stack and takes user focus.

```
Backstack
Screen 1 --> Screen 2 --> Screen 3
```

An activity is started with an Intent. An intent is a message from one activity to another activity. We can pass information from one activity to another using intents.

Types of Intents
An Intent can be of two types:

1. Implicit Intent: Target Component not known.
   
2. Explicit Intent: Target of the intent is known.
   
   - Fully classified class name of a specific activity known.
Our First Multi-Screen App

We will now create an app which has an EditText on the first activity. User enters a message and clicks a button to send this message to second activity.

We will now create this App using the following steps:

1. Create a new app
2. Design the layout for the first app
3. Create button click listener
4. Create the second activity
5. Add proper links & metadata in Manifest.xml
6. Add an Intent using the following code:

   ```java
   Intent intent = new Intent(this, SecondActivity.class);
   startActivityForResult(intent);
   ```

   This is an explicit Intent

Sending data across activities

We can send data across activities using intent extras. Intent extras are key/value pairs in a Bundle. A Bundle is a collection of data stored as key/value pairs.
Syntax for Intent.putExtra looks like this:

    intent.putExtra("key", "value");

We can get this message from another activity using:

    Intent intent = getIntent();
    String message = intent.getStringExtra("key");

Implicit Intents

In implicit intent, we initiate an activity without knowing which app or activity will handle the task.
Ex: Take a photo, open this URL etc.

Activities with matching intent filters opt in to perform the action.

Creating an App with Implicit Intent

Following code starts an activity to open a URL

    String url = "Some URL" // e.g. https://codewithmanny.com
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        // cannot handle intent
    }


Similarly we can handle intents with other actions. For eg: open a location, share text etc. See android docs for more.
Chapter 4 - Practice Set

1. Create a layout in Android which looks like this

   ![Diagram of a layout]

   Explore how to get this full screen theme

   buttons

2. Create an android app capable of opening your favorite website on a single click

3. Create a quiz app using the layout in 1

4. Create an App capable of sending Email to the user. The app should take Email, Subject, and message as input
Chapter 5 - ListView & RecyclerView

In Android, a scrollable list of items is implemented using a ListView. The data is populated into the list using an Adapter.

Adapter converts an Array/Arraylist into View items.

Data Source ← Adapter ← View

Array ← ArrayList
Adapter → ArrayList

ArrayAdapter
ArrayAdapter is used to display a set of items in a ListView.

ArrayAdapter < String > ad = new ArrayAdapter(this, R.layout.list_item, 5Arr);  // context  
ListView.setAdapter(ad);

This sets the content of 5Arr to ListView

Custom Array Adapter
We can create custom Array Adapters by creating a model and a class (eg: MyAdapter) which extends ArrayAdapter.
We can pass our string array to MyAdapter like this:

MyAdapter ad = new MyAdapter(this, 5Arr);
ListView.setAdapter(ad)
Why use ListViews

It will be very hectic to create a scrollable view where data is coming from a DataSource. Just imagine how hectic would it be to otherwise populate data into a View to show it to the user. Hence Listviews are used due to the following reasons:

- User & Developer Friendly
- Optimized to some extent
- Easy to customize
- Simple lists can be created in few lines of code.

Adding onClick listener to items

We can override onItemClickListener method to add click listeners as shown below:

```java
listView.setOnItemClickListener(new OnItemClickListener()
```

Built-in XML layouts

Android provides you a list of built-in layouts to be used within ListViews.

Ex:  
- android.R.layout.simple_list_item_1
- android.R.layout.simple_list_item_2
- etc.

Thanks Android for providing free layout!
RecyclerView

It's simply a better version of list views.

In ListViews → Memory is directly proportional to the number of items.

RecyclerView solves this problem by recycling the existing views hence saving up on memory.

We simply update the Adapter in RecyclerView to an Adapter which is capable of Recycling the Views.

**ListViews vs RecyclerView**

- **ListView**
  - User keeps scrolling which adds views & hence more memory is consumed.
  - Views are recycled when a user scrolls resulting into Performance boost.

- **RecyclerView**
  - Old views (scrolled)
  - New View
  - Recycled View
How to Implement a RecyclerView

A RecyclerView can be implemented just like ListViews using an Adapter. All the major changes are done to the Adapter. Android Docs has a page which can be used as a starter template for implementing RecyclerViews.
Chapter 5 - Practice Set

1. Create an app using ListView to display the contents of a String array with onClick listeners.

2. Create an app using RecyclerView from Android Studio.

3. Create an app using RecyclerView to display contents of the “Contact” array which looks like this:

   public class Contact {
     private int sno;
     private String phNo;
     private String name;
   }
Chapter 6 - Media Player

Media is a crucial component of Android.

Media = Audio + Video + Images

The MediaPlayer class
This class provides APIs for playing variety of Media types. We can simply create an instance, add a music & play it.

Adding music to the android "raw" directory
We can add an mp3 file to our res/raw folder. If the raw folder is not present, you need to create one.

Playing our first music
We can create an instance of MediaPlayer and play our first music by adding following lines inside onCreate:

```java
MediaPlayer = MediaPlayer.create(this, R.raw.music);
MediaPlayer.start();
```

Similarly we can use mediaPlayer.pause() to pause the music from the mediaPlayer.

Playing music from the web
We can play music from the web using the setDataSource method of mediaPlayer.
The following steps plays music from the web:

Step 1: Create a new mediaPlayer instance
mediaPlayer = new MediaPlayer();

Step 2: Set the datasource
mediaPlayer.setDataSource("https://audio.source.com")

Step 3: Add android.permission.INTERNET to manifest

Step 4: Add android:usesCleartextTraffic="true" in the application tag of manifest (AndroidManifest.xml)

Step 5: Add OnPreparedListener to the media player & override required methods.

Step 6: Run mediaPlayer.prepareAsync(); method to start preparing the media player.

Adding SeekBar
Android SDK provides a SeekBar widget which allows developers to add a progress bar for media execution. Further, we can add OnSeekBarChangeListener() to take actions when the seekbar is changed. This can be done like this:

SeekBar.setMax(mediaPlayer.getDuration());
SeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBar...
Playing Videos

We can use the same MediaPlayer class to play videos in Android.

In order to display videos we use a SurfaceView like this:

```java
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.my_video);
SurfaceHolder holder = surfaceView.getHolder();
holder.addCallback(new SurfaceHolder.Callback() {
    // override methods here

    void onWindowInsert(SurfaceHolder holder, int type) {
    // your implementation
    }

    void onWindowChange(SurfaceHolder holder, int format, int frameRate)
    // your implementation
    });
```
Chapter 6 - Practice Set

1. Create an Android App which plays your favorite song on loop.

2. Add a seekbar in 1 for you to seek to a specific part in your favorite song.

3. Create an Android app which plays three favorite files of a user from web server.

4. Create an App which plays two videos simultaneously on one Android screen.

5. Stop one when another Video is played from your app in 4.
Chapter 7 - Storing Data

Saving data makes apps more lively.

Data persistence = Storing & saving the data

In android, we can save data using these common ways:

- Text File → Not Efficient
- Shared Preferences → Somewhat efficient
- Database → Highly Efficient

Shared Preferences

Shared Preferences allow us to save and retrieve data in the form of key-value pair.

We can access Shared preferences like this:

```java
SharedPreferences sp = getSharedPreferences(MESSAGE, MODE_PRIVATE);
SharedPreferences.Editor ed = sp.edit();
ed.putString("text", "This is me");
ed.apply();
```
sP. get String ("text", "default Value");

// Retrieve Data

Working with Database

We can use SQLite database to work with databases in Android.

We can create a class extending from SQLiteOpenHelper and use it for CRUD operations.
Chapter 7 - Practice Set

1. Store the HiScore of a game using Shared Preferences and fetch it into a TextView.

2. Create a database to store data of Employees using SQLiteOpenHelper class.

3. Add an Entry to the database created in 2.

4. Fetch an Entry from the database created in 2.
Project - Music Player

Create a music player iSenseit and add pause/play functionality to it. Your music player should be capable of reading the SD card and fetch all the songs into a view such that user can play the song of their choice.