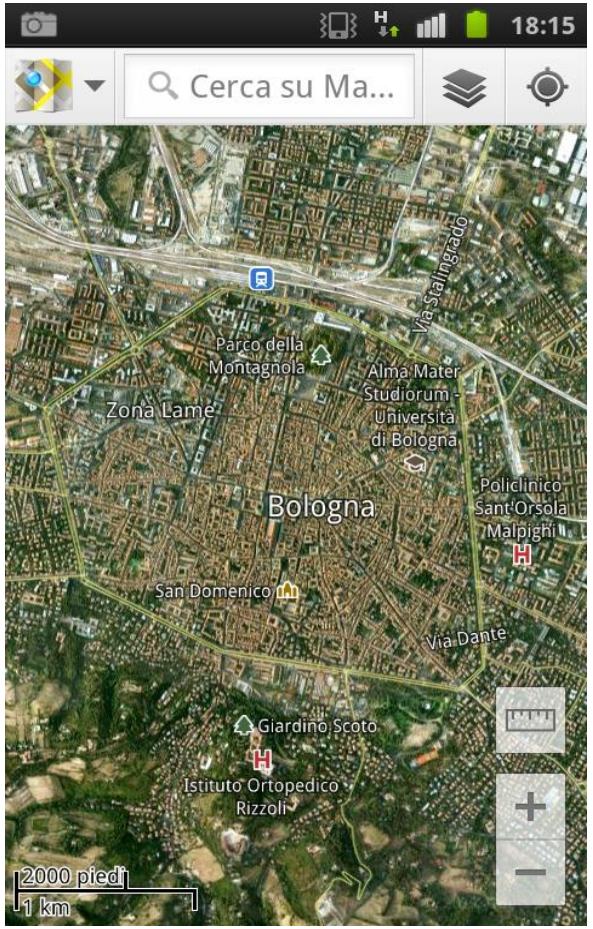
CIT 4404 Mobile App Development
CIT 4313 Responsive User-Interface
Topic3: Basic Android Application Components

Dr. Fullgence Mwakondo
Institute of Computing and Informatics
Technical University of Mombasa
mwakondo@tum.ac.ke

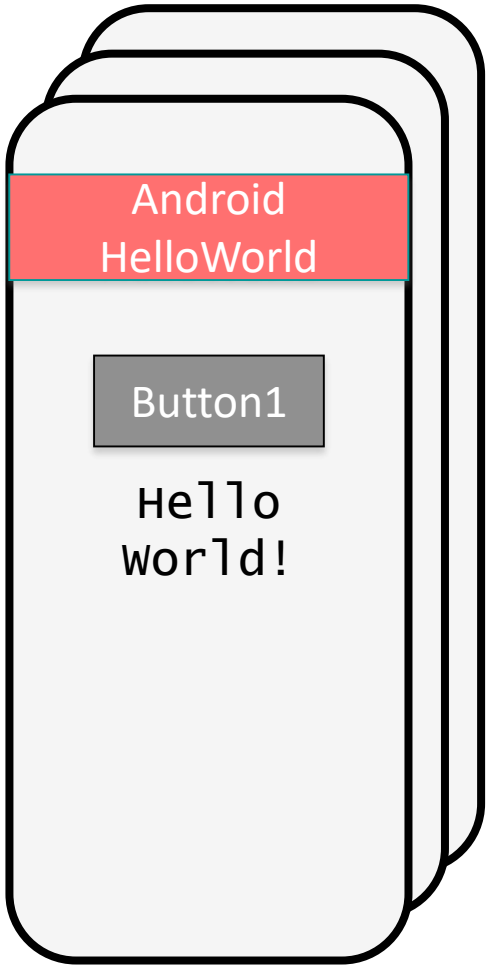
Android Applications Design



APPLICATION **COMPONENTS**

- **Activities**
- **Intents**
- **Services**
- **Content Providers**
- **Broadcast Receivers**

Android Components: Activities



- An **Activity** is a code that creates a **single screen** with a **User Interface** of an **Application** .
- An Application can be composed of *multiples screens* (Activities).
- The **Home Activity** is shown when the user launches an application.
- Different activities can exhange information one with each other.

Android Components: Activities

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

PROGRAMMATIC APPROACH

MainActivity.java

Example:

```
Button button=new Button (this);
TextView text= new TextView();
text.setText("Hello world");
```

Android Components: Activities

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

DECLARATIVE APPROACH

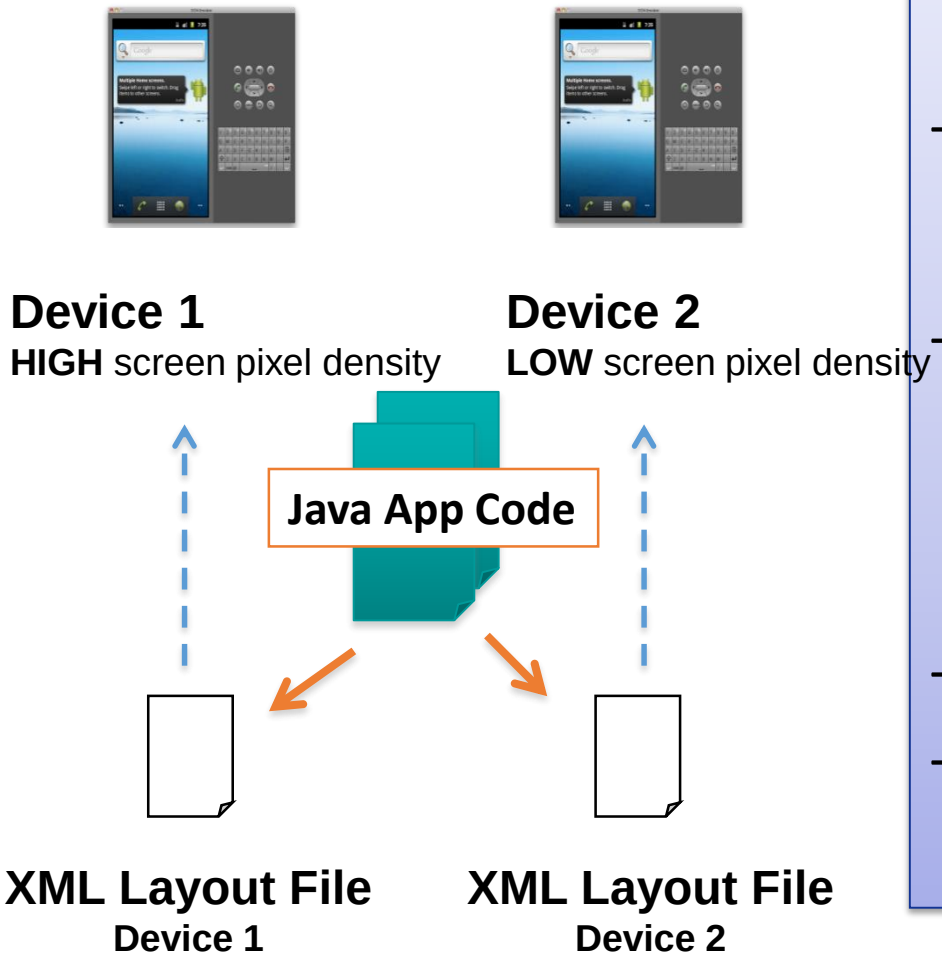
activity_main.xml

Example:

```
< TextView android.text=@string/hello”
android:textcolor=@color/blue
android:layout_width=“fill_parent”
android:layout_height=“wrap_content” />
< Button android.id=“@+id/Button01”
android:textcolor=“@color/blue”
android:layout_width=“fill_parent”
android:layout_height=“wrap_content” />
```

Android Components: Activities

EXAMPLE



- Build the **application layout** through XML files (like HTML)
- Define **two** different XML **layouts** for two different devices
- At **runtime**, Android detects the current device configuration and loads the appropriate resources for the application
- **No need to recompile**!
- Just add a new XML file if you need to support a new device

Android Components: Activities

- *Android applications typically use both the approaches*!

DECLARATIVE APPROACH

XML Code

Define the Application **layouts** and **resources** used by the Application (e.g. labels).

PROGRAMMATIC APPROACH

Java Code

Manages the **events**, and handles the **interaction** with the user.

Android Components: Activities

- **Views** can generate **events** (caused by human interactions) that must be managed by the Android-developer.

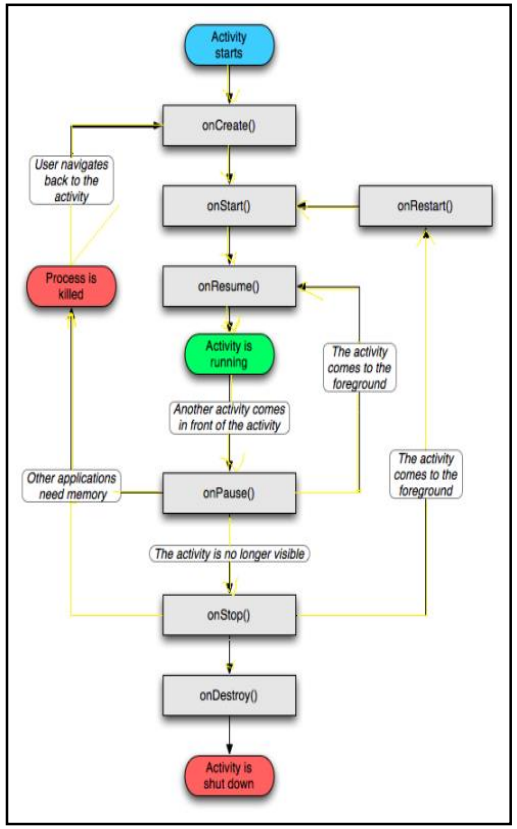
Button

TextEdit

Example

```
public void onClick(View arg0) {
        if (arg0 == Button) {
                // Manage Button events
        }
}
```

Android Components: Activities



- The **Activity Manager** is responsible for creating, destroying, managing activities.
- Activities can be on different **states**: *starting*, *running*, *stopped*, *destroyed*, *paused*.
- Only one activity can be on the **running** state at a time.
- Activities are organized on a **stack**, and have an event-driven life cycle (details later …)

Android Components: Activities

- Main difference between Android-programming and Java (Oracle) -programming:
  - **Mobile devices have constrained resource capabilities!**
- Activity lifetime depends on **users' choice** (i.e. change of visibility) as well as on **system constraints** (i.e. memory shortage).
- Developer must implement **lifecycle methods** to account for state changes of each Activity …

Android Components: Activities

```
public class MyApp extends Activity {

      public void onCreate() { ... }
      public void onPause()  { ... }
      public void onStop()   { ... }
      public void onDestroy(){ ... }
      ….
}
```

Called when the Activity is **created** the first time.

Called when the Activity is **partially visible**.

Called when the Activity is **no longer visible**.

Called when the Activity is **dismissed**.

Android Components: UI screen components

- A typical user interface of an android application consists of action bar and content area.
  - 1. Main Action Bar
  - 2. View Control
  - 3. Content Area
  - 4. Split Action Bar

Android Components: Example on Activities programmatic approach

```
Public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button button= new Button(this);
        button.setText("CANCEL THIS OPERATION");

        TextView text = new TextView(this);
        text.setText("HELLO BMCS");

        setContentView(button);
    }
}
```

Android Components: Example on Activities Declarative approach

```
Public class MainActivity extends AppCompatActivity {

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

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="30dp"
        android:text="OK"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <Button
        android:id="@+id/buttonDELETE"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="201dp"
        android:text="DELETE"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Android Components: Views for Activities

- An activity consists of views. A view is just a widget that appears on the screen such as:
  - TextView – used to display text to the user
  - EditText – textview that allows users to edit its content
  - Button – push button widget
  - ImageButton – a button that displays an image
  - ToggleButton – displays checked/unchecked states using light indicator
  - CheckBox – special button with two states: checked and unchecked
  - RadioButton – special button with two states: checked and unchecked

Android Components: Views Attributes

- All Views and ViewGroups have
- a unique identifier integer assigned at compile time, mapped to a user-specified variable: android:id=”@+id/my_button”
- Size defined in width and height.
- view must define width and height relative to the parent.
  - wrap_content sizes view to its content.
  - match_parent makes view as big as its parent ViewGroup allows.
- Views are rectangles with left and top coordinates. Can get location with getLeft() and getTop() Defined relative to the parent.

Android Components: Accessing views for Activities - programmatically

- Accessing views xml layout file programmatically in Activities.

```
Public class MainActivity extends AppCompatActivity {
    private Button mybuttonDELETE;
    private TextView mytextview;
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);

         mybuttonDELETE =(Button)findViewById(R.id.buttonDELETE);
         mytextview = (TextView)findViewById(R.id.textview);
       }
}
```

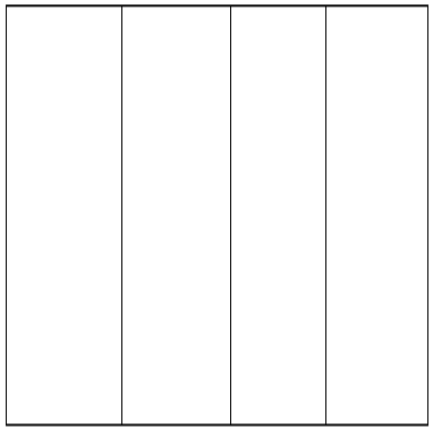
Android Components: ViewGroups for Activities

- One or more views can be grouped together into one GroupView e.g. Layouts, RadioGroups
- TYPES OF LAYOUT

- There are many types of layout. Some of which are listed below:
  - Linear Layout
  - Absolute Layout
  - Table Layout
  - Frame Layout
  - Relative Layout
  - Grid Layout
  - Constraint Layout

Android Components: ViewGroup : LinearLayout

- Arranges the views objects sequentially as they are inserted from top to bottom(vertically) and left to right (horizontally)
- Has a property orientation whose values are:
  - Vertical {LinearLayout.VERTICAL}
  - Horizontal {LinearLayout.HORIZONTAL}
- Has a method used to set the property
  - *linearlayout*.setOrientation(value)



Android Components: Example on LinearLayout programmatic approach

```
Public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button mybutton =  new Button(this);
        mybutton.setText("WELCOME TO TUM RENTAL APP! PRESS THE BUTTON");
        TextView mytextview = new TextView(this);
        mytextview.setText("WHAT IS YOUR NAME?");
        EditText myedittext = new EditText(this);

        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);

        mylayout.addView(mytextview);
        mylayout.addView(myedittext);
        mylayout.addView(mybutton);

        setContentView(mylayout);
    }
}
```

Android Components: Example on LinearLayout declarative approach

a) Actvity Java file

```
Public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mylinearlayout);
    }
}
```

b) Layout xml file

Layout has two widgets, which have no constraints on each other's size or location

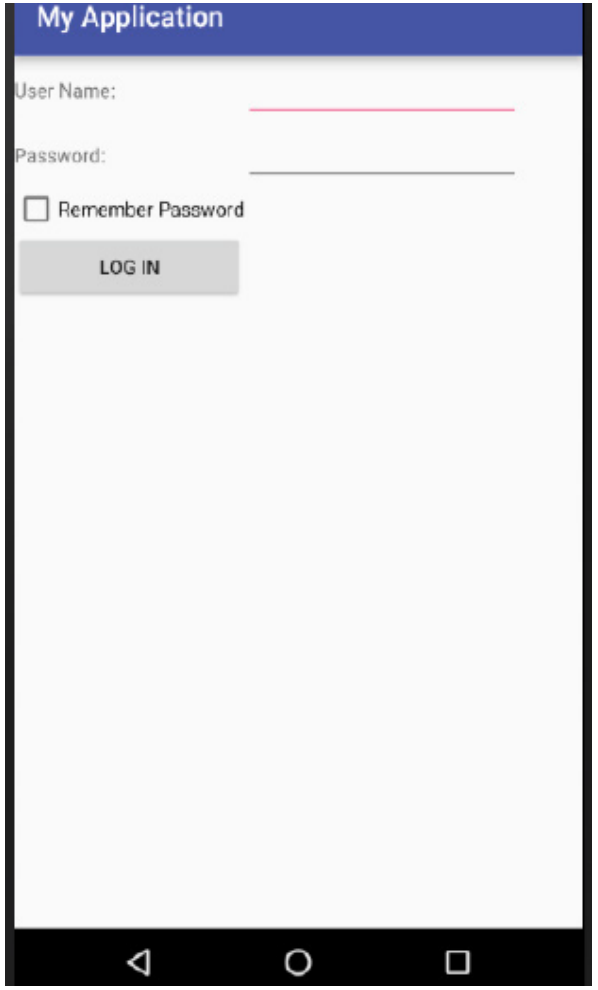
Text box containing a set string

Button with a set string.

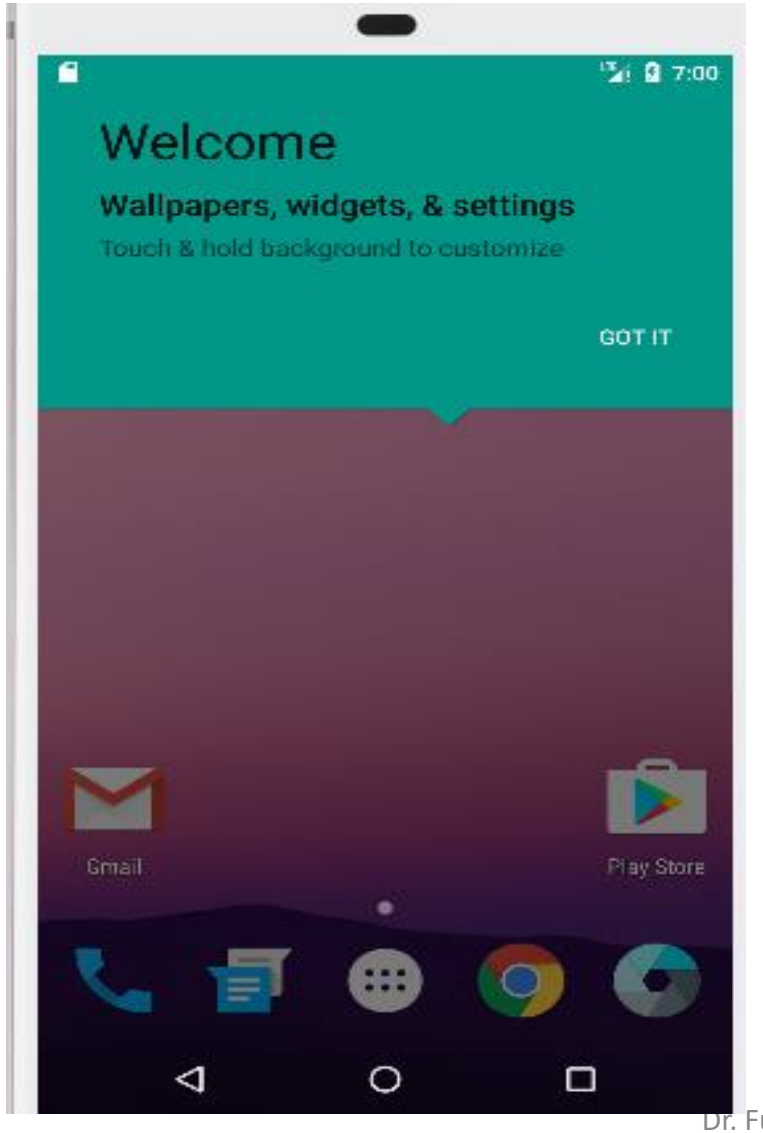
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Android Components: ViewGroup : TableLayout

- A layout that resembles a spreadsheet arranges the views objects inside the cells i.e. groups views into rows and columns
- Width of columns is determined by the largest cell in that column
- Has to be used with TableRow
  - *linearlayout*.setOrientation(value)



Android Components: Example on TableLayout programmatic approach




```
Public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText("WELCOME TO TUM RENTAL APP! PRESS THE BUTTON");
        TextView mytextview = new TextView(this);
        mytextview.setText("WHAT IS YOUR NAME?");
        EditText myedittext = new EditText(this);

        TableLayout mylayout = new TableLayout(this);
        TableRow firstrow = new TableRow(this);
        TableRow secondrow = new TableRow(this);

        mylayout.addView(firstrow);
        mylayout.addView(secondrow);
        firstrow.addView(mytextview);
        firstrow.addView(myedittext);
        secondrow.addView(mybutton);
        setContentView(mylayout);
    }
}
```

Android Components: Example on TableLayout Declarative approach

a) Actvity Java file

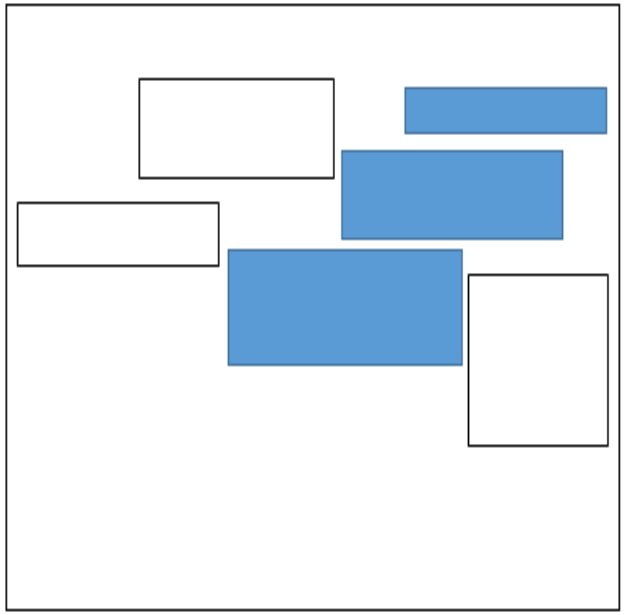
```java
Public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mytablelayout);
    }
}
```

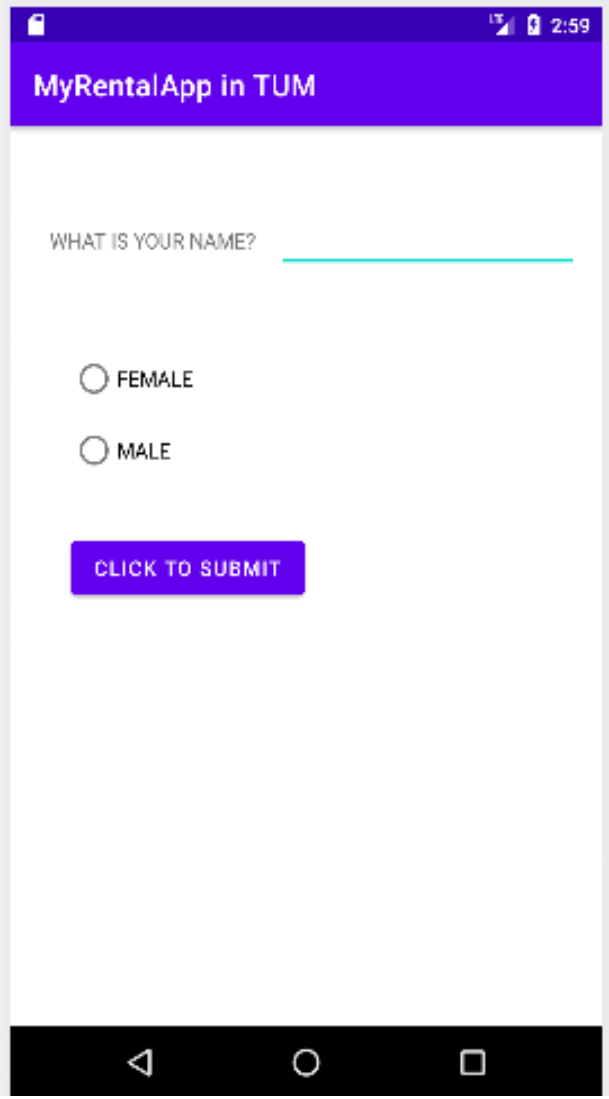
b) Layout xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/what_is_your_name" />
        <EditText
            android:id="@+id/editTextTextPersonName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="" />
    </TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/button" />
    </TableRow>
</TableLayout>
```

Android Components: ViewGroup: ConstraintLayout

- Allows position/size specification based on spatial relationships between views.
- All views move together as screen size changes.
- Has a number of properties
- Use wrap_content, match_parent.
- Automatically adjusts based on size and orientation of screen.
- Easiest to create in Android Studio Layout Editor (Visual Designer)
- Because it is complex





Android Components: Example on ConstraintLayout Declarative approach

a) Actvity Java file

```
Public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myconstraintlayout);
    }
}
```

b) Layout xml file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:text="@string/what_is_your_name2"

        app:layout_constraintBaseline_toBaselineOf="@+id/editTextTextPersonName2"
        app:layout_constraintEnd_toStartOf="@+id/editTextTextPersonName2"
        app:layout_constraintStart_toStartOf="parent" />
    <EditText
        android:id="@+id/editTextTextPersonName2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="53dp"
        android:layout_marginEnd="1dp"
        android:ems="10"
        android:labelFor="@id/editTextTextPersonName2"
        android:importantForAutofill="no"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView3"
        app:layout_constraintTop_toTopOf="parent" />
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="42dp"
        android:layout_marginTop="145dp"
        android:layout_marginEnd="43dp"
        app:layout_constraintEnd_toEndOf="@+id/textView3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <RadioButton
            android:id="@+id/radioButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/female" />
        <RadioButton
            android:id="@+id/radioButton3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/male" />
    </RadioGroup>
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="42dp"
        android:layout_marginTop="31dp"
        android:text="@string/click_to_submit"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/radioGroup" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

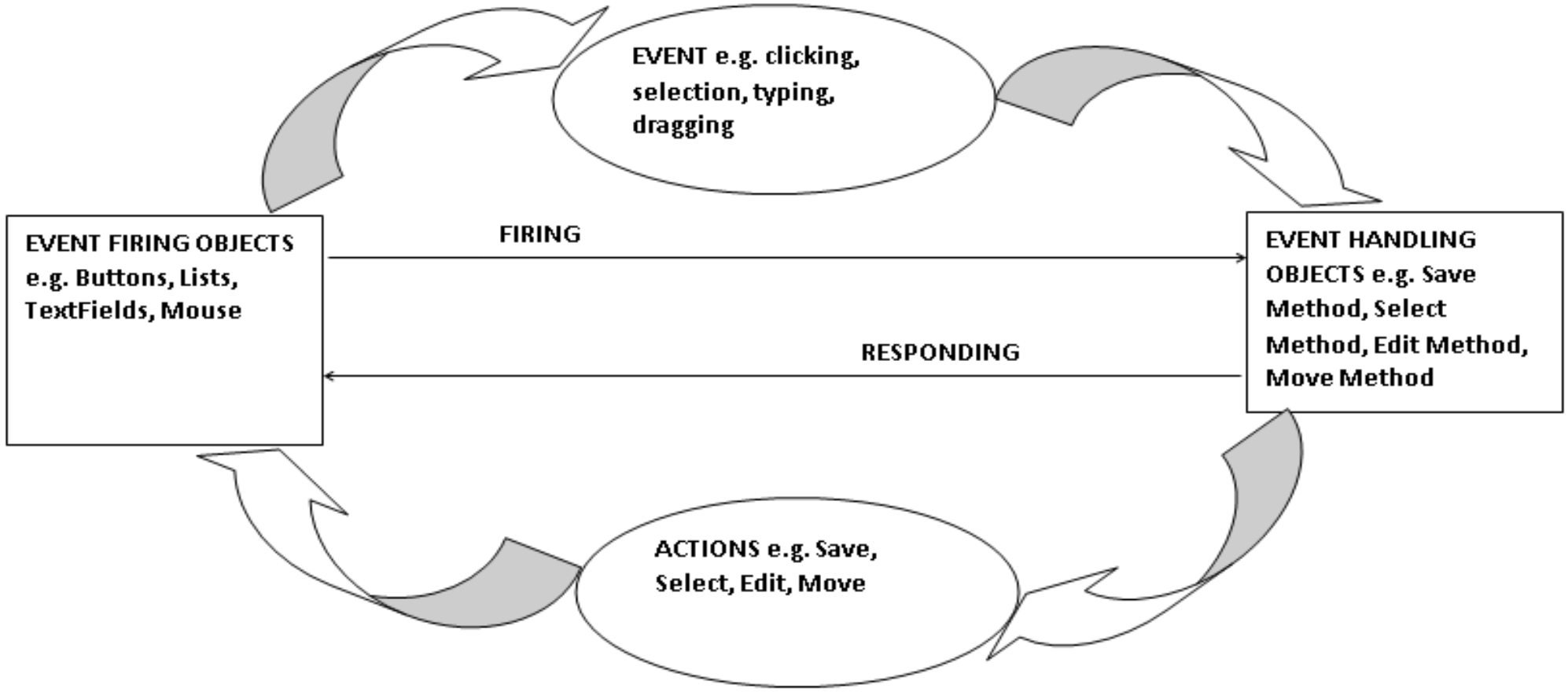
Android Components: - Responsive UI Design

- Android defines two characteristics for each screen:
- Screen Size (physical size) :ie. Small, Normal, Large, XLarge
- Screen Density (density of pixels (dpi) on screen): i.e. MDPI (~160dpi), HDPI (~240dpi), XHDPI (~320dpi), XXHDPI (~480dpi), XXXHDPI (~640dpi)
- Apps are compatible with all screen sizes and densities automatically, but this may not create a good UX.
- Create specialized layouts, optimize images for density.
- Avoid hard-coded layout sizes: Use wrap_content, match_parent.
- Automatically adjusts based on size and orientation of screen.

Android Components: Handling View Events

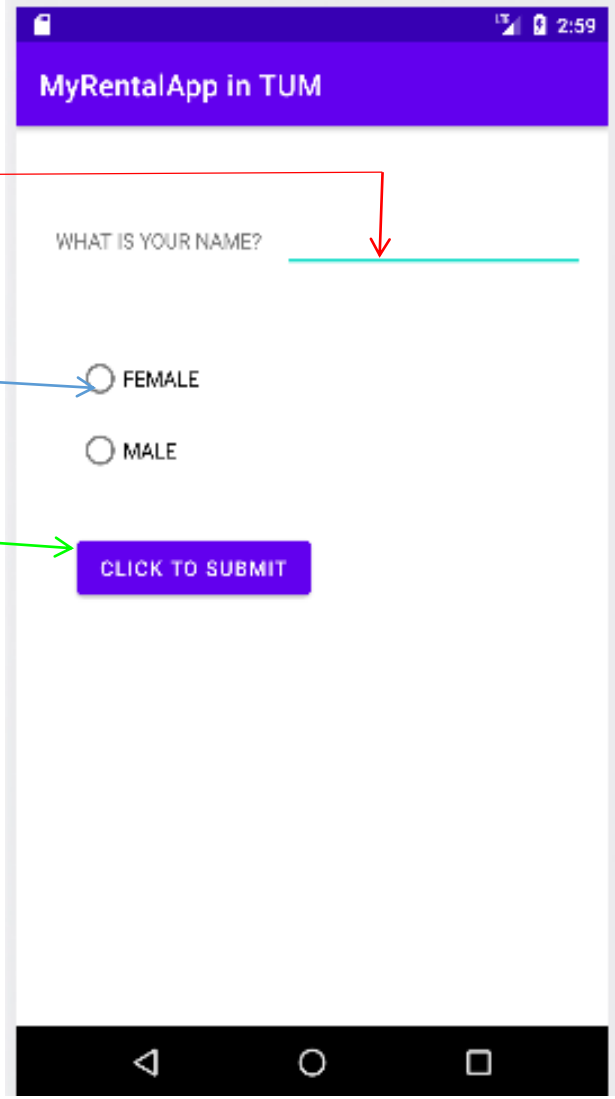
- Event is an occasion (in time) that is marked with a special occurrence for ease of remembrance (marriage…wedding, birth…party, death…funeral, Independence…celebration, etc.)
- Ordinarily, events are generated by nature or user actions in the time space and responded to with occurrences.
- In Programming, user actions on GUI components (Buttons, Lists, TextFields, Mouse…) generate events (clicking, selection, typing, dragging…).
- GUI components actions of event generation is called event firing.
- Event handling constitutes responding to the event with an action.

Android Components: Handling View Events



Android Components: Handling View Events

- Views in android that generate events are:
  - EditText
  - Checkbox
  - RadioButton
  - ToggleButton
  - Button
- Events generated are:
  - **Click**
  - **Select**
  - Drag
  - Press
  - **Touch**
  - **Type**



Android Components: Handling View Events programmatic approach

```
Public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

         Button mybutton= new Button(this);
         mybutton.setOnclickListener(new View.onClickListener(){
         @Override
             public void onClick(View v) {
               //code for what to do here            }
        });
        setContentView(mybutton);
    }
}
```

Android Components: Example on Handling View Events programmatic approach

```
Public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

         Button mybutton= new Button(this);
         mybutton.setOnclickListener(new View.onClickListener(){
         @Override
            public void onClick(View v) {
              //code for what to do here
              Toast.makeText(this, "THIS IS SAVE BUTTON",Toast.LENGTH_SHORT).show();
             }
        });
        setContentView(mybutton);
    }
}
```

Android Components: Handling View Events

Declarative approach

a) Actvity Java file

```java
Public class MainActivity extends AppCompatActivity {

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

- Format
- EventName="ListenerName"

b) Layout xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/androi
    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"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/buttonOK"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="30dp"
        android:onClick="okClick"
        android:text="OK"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

      <Button
        android:id="@+id/buttonCANCEL"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="29dp"
        android:layout_marginEnd="71dp"
        android:onClick="cancelClick"
        android:text="CANCEL"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Android Components: Example on Handling View Events

Declarative approach

a) Actvity Java file

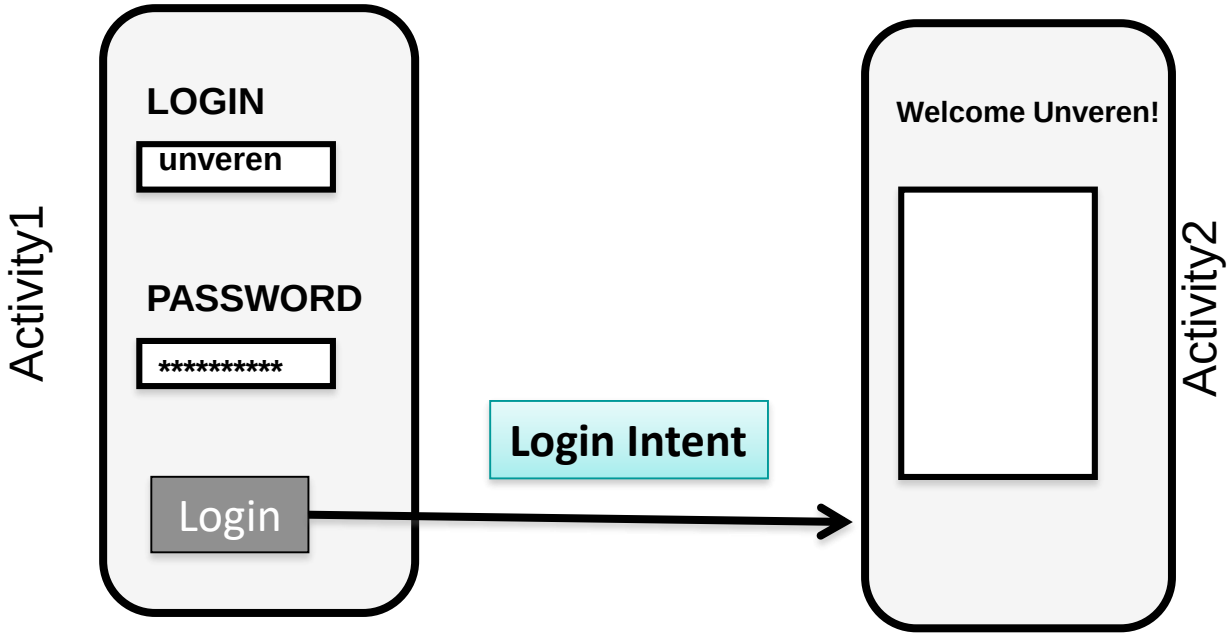
```java
Public class MainActivity extends AppCompatActivity {

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

b) Layout xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/buttonOK"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="30dp"
        android:onClick="okClick"
        android:text="OK"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="UnknownId" />
    <Button
        android:id="@+id/buttonCANCEL"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="29dp"
        android:layout_marginEnd="71dp"
        android:onClick="cancelClick"
        android:text="CANCEL"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/buttonDELETE"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="201dp"
        android:onClick="deleteClick"
        android:text="DELETE"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/buttonSAVE"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="201dp"
        android:layout_marginEnd="70dp"
        android:onClick="saveClick"
        android:text="SAVE"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

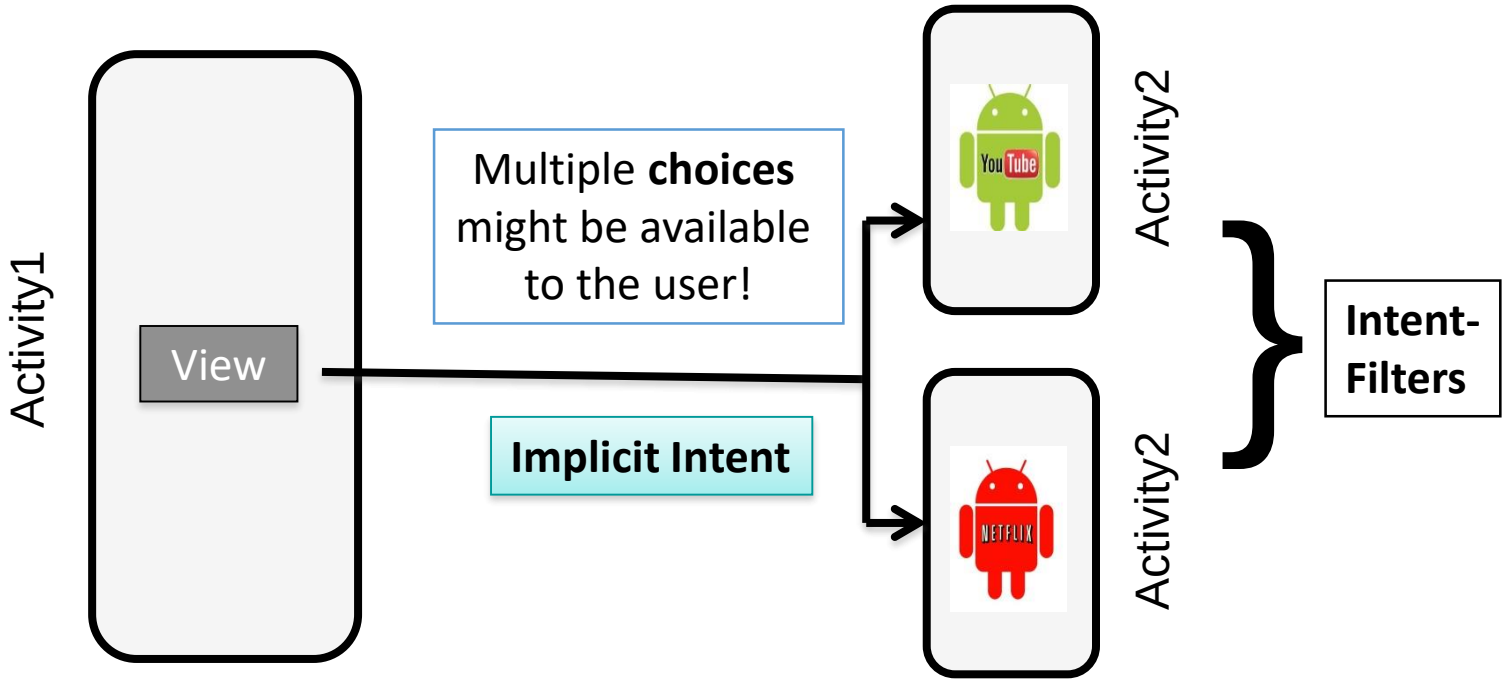
Android Components: Intents

- **Intents**: a code that enables switching or navigating between android app components i.e to pass data or asynchronous **messages** to activate core Android components (e.g. Activities).
- **Explicit** Intent → The component *(e.g. Activity1)* specifies the <u>destination</u> of the intent *(e.g. Activity 2)*.



Android Components: Intents

- **Implicit** Intent → The component *(e.g. Activity1)* specifies the <u>type</u> of the intent *(e.g. "View a video")*.
- Intents are used to link actvities to form a complete Android Application and also allow us to pass data to/between activity instances



Android Components: Switching Activity using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  ……————————————————————————————————————————————————————————.
  setContentView(mylayout);
  ————————————————————————————————————————————————————————————
  //Declare and initialize an Intent Variable
  Intent myIntent=new Intent(this,Activity2.class);

  //Switch to Activity2
  StartActivity(myIntent);
 }
}
```

```
Public class Activity2 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   ————————————————————————————————————————————————————————————..
   setContentView(mylayout);
  }
}
```

Android Components: Example on Switching Activity using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText(“DISPLAY ACTIVITY 2! PRESS THE BUTTON");
        mybutton.setOnclickListener(new View.onClickListener(){
           @Override
           public void onClick(View v) {
              DisplayActivity2();
            }
        });
        TextView mytextview = new TextView(this);
        mytextview.setText(“HELLO STUDENTS! WELCOME TO INTENT
DEMONSTRATION");

        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);
        mylayout.addView(mytextview);
        mylayout.addView(mybutton);
        setContentView(mylayout);
    }
   public void DisplayActivity2(){
      Intent myIntent=new Intent(this,Activity2.class);
      StartActivity(myIntent);
   }
}
```

```
Public class Activity2 extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button mybutton =  new Button(this);
        mybutton.setText("WELCOME TO TUM RENTAL APP! PRESS THE BUTTON");
        TextView mytextview = new TextView(this);
        mytextview.setText("WHAT IS YOUR NAME?");
        EditText myedittext = new EditText(this);

        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);

        mylayout.addView(mytextview);
        mylayout.addView(myedittext);
        mylayout.addView(mybutton);

        setContentView(mylayout);
    }
}
```

Android Components: Passing Data between Activities using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  …………………………………………………………………………………………….
  setContentView(mylayout);
  ……………………………………………………………………………………………
 //Declare and assign a variable some DATA
  DataType myVariable1 = DATA;
 //Declare and initialize an Intent Variable
  Intent myIntent1=new Intent(this,Activity2.class);
 //Put DATA variable in the Intent using putExtra
   myIntent1.putExtra("MYVARIABLE",myVariable1);
 //Switch to Activity2
  StartActivity(myIntent1);
 }
}
```

```
Public class Activity2 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   ……………………………………………………………………………………………..
   setContentView(mylayout);
  //Declare and initialize an Intent Variable
  Intent myIntent2=new Intent();
  //Declare and assign a variable to receive DATA
  DataType myVariable2 = getIntent().
getStringExtra("MYVARIABLE");
  }
}
```

Android Components: Example on Passing Data between Activities using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    private String myname = “MWAKONDO”;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText(“DISPLAY ACTIVITY 2! PRESS THE BUTTON");
        mybutton.setOnclickListener(new View.onClickListener(){
           @Override
           public void onClick(View v) {
              DisplayActivity2();
            }
        });
        TextView mytextview = new TextView(this);
        mytextview.setText(“HELLO STUDENTS! WELCOME TO INTENT
DEMONSTRATION");

        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);
        mylayout.addView(mytextview);
        mylayout.addView(mybutton);
        setContentView(mylayout);
    }
   public void DisplayActivity2(){
      Intent myIntent=new Intent(this,Activity2.class);
      myIntent.putExtra(“MYDATA”,myname);
      StartActivity(myIntent);
   }
}
```

```
Public class Activity2 extends AppCompatActivity {
    private String myvariable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText("WELCOME TO TUM RENTAL APP! PRESS THE BUTTON");
        TextView mytextview = new TextView(this);
        mytextview.setText("WHAT IS YOUR NAME?");
        EditText myedittext = new EditText(this);
        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);
        mylayout.addView(mytextview);
        mylayout.addView(myedittext);
        mylayout.addView(mybutton);

        setContentView(mylayout);
        myvariable = getIntent().getStringExtra(“MYDATA”);
        Toast.makeToast(this,myvariable, TOAST.LENGTH_SHORT).show()
    }
}
```

Android Components: Returning Results from Activities using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  ………………………………………………………………………………….
  setContentView(mylayout);
  ………………………………………………………………………………….
 //Declare and assign a variable some DATA
  DataType myVariable1 = DATA;
 //Declare and initialize an Intent Variable
  Intent myIntent1=new Intent(this,Activity2.class);
 //Put DATA variable in the Intent using putExtra
   myIntent1.putExtra(“MYVARIABLE”,myVariable1);
 //Switch to Activity2
  StartActivity(myIntent1);
 }
}
```

```
Public class Activity2 extends AppCompatActivity {
    @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   ………………………………………………………………………………..
   setContentView(mylayout);
  //Declare and initialize an Intent Variable
  Intent myIntent2=new Intent();
  //Declare and assign a variable to return DATA
  DataType myVariable2 = DATA VALUE;
  //Set data to pass back
  myIntent2.setData(uri.parse(myvariable2);
  setResult(RESULT_OK,myIntent2);
  finish();
  }
}
```

Android Components: Example on Returning Results from Activities using Intent programmatic approach

```
Public class Activity1 extends AppCompatActivity {
    private String myname = “MWAKONDO”;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText(“DISPLAY ACTIVITY 2! PRESS THE BUTTON");
        mybutton.setOnclickListener(new View.onClickListener(){
           @Override
           public void onClick(View v) {
              DisplayActivity2();
            }
        });
        TextView mytextview = new TextView(this);
        mytextview.setText(“HELLO STUDENTS! WELCOME TO INTENT
DEMONSTRATION");

        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);
        mylayout.addView(mytextview);
        mylayout.addView(mybutton);
        setContentView(mylayout);
    }
   public void DisplayActivity2(){
      Intent myIntent=new Intent(this,Activity2.class);
      myIntent.putExtra(“MYDATA”,myname);
      StartActivity(myIntent);
   }
}
```

```
Public class Activity2 extends AppCompatActivity {
    private String myvariable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button mybutton =  new Button(this);
        mybutton.setText("WELCOME TO TUM RENTAL APP! PRESS THE BUTTON");
        TextView mytextview = new TextView(this);
        mytextview.setText("WHAT IS YOUR NAME?");
        EditText myedittext = new EditText(this);
        LinearLayout mylayout = new LinearLayout(this);
        mylayout.setOrientation(LinearLayout.VERTICAL);
        mylayout.addView(mytextview);
        mylayout.addView(myedittext);
        mylayout.addView(mybutton);

        setContentView(mylayout);
        myvariable = getIntent().getStringExtra(“MYDATA”);
        Toast.makeToast(this,myvariable, TOAST.LENGTH_SHORT).show()
    }
}
```