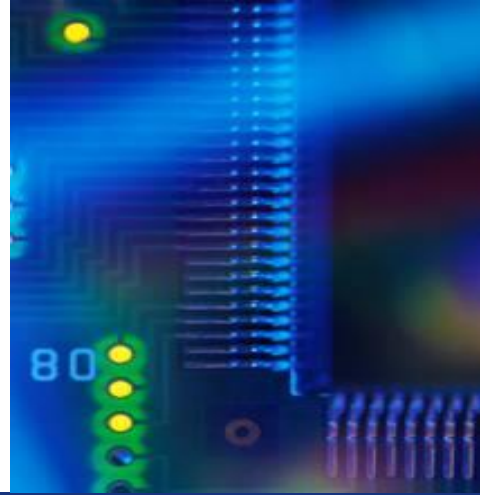
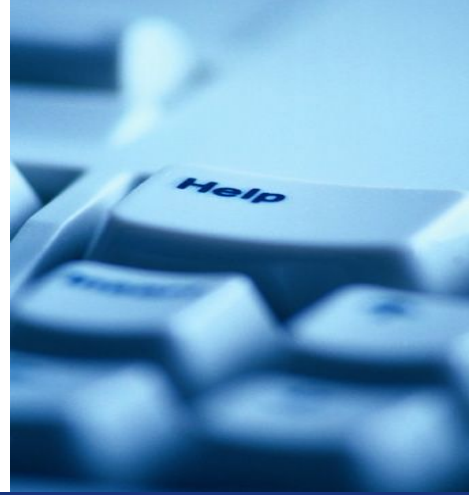




**UNIVERSITAS
BUDI LUHUR**



FAKULTAS TEKNOLOGI INFORMASI

Mobile Programming

[PG119/ 3 SKS]

Riskiana Wulan, MKom

Review Before UTS

Pengumuman

Libur (Cuti Bersama) Natal dan Tahun Baru

2024-12-24 (Week 13)
CUTI BERSAMA

2024-12-31 (Week 14)
CUTI BERSAMA

Kuliah Pengganti (ONLINE)

2024-12-13 (Jumat, jam 09.45 sampai selesai)
2024-12-20 (Jumat, jam 09.45 sampai selesai)

Layout

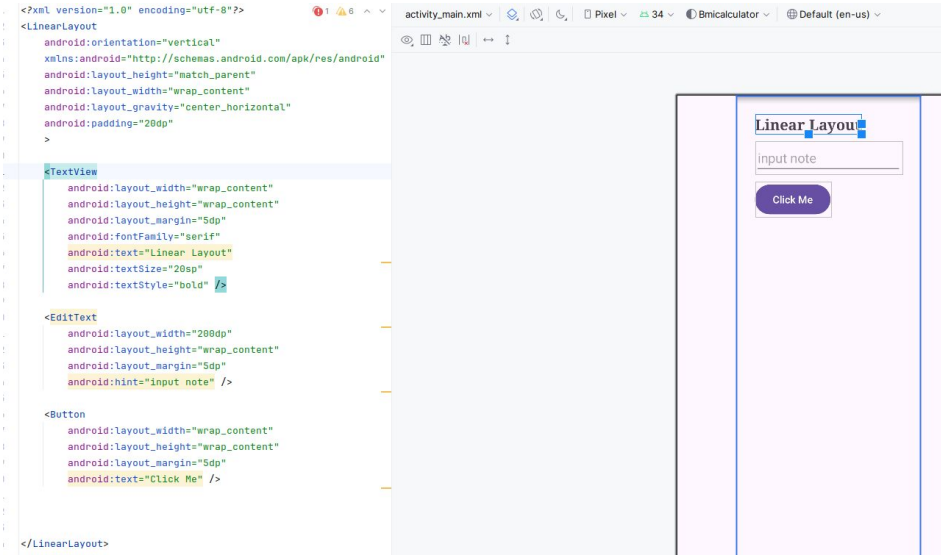
1. Linear Layout

Description: Arranges child views in a single column (vertical) or a single row (horizontal).

Orientation: Either vertical or horizontal.

Usage: Use `android:orientation="vertical"` for a vertical layout and `android:orientation="horizontal"` for a horizontal layout.

Sample vertical layout



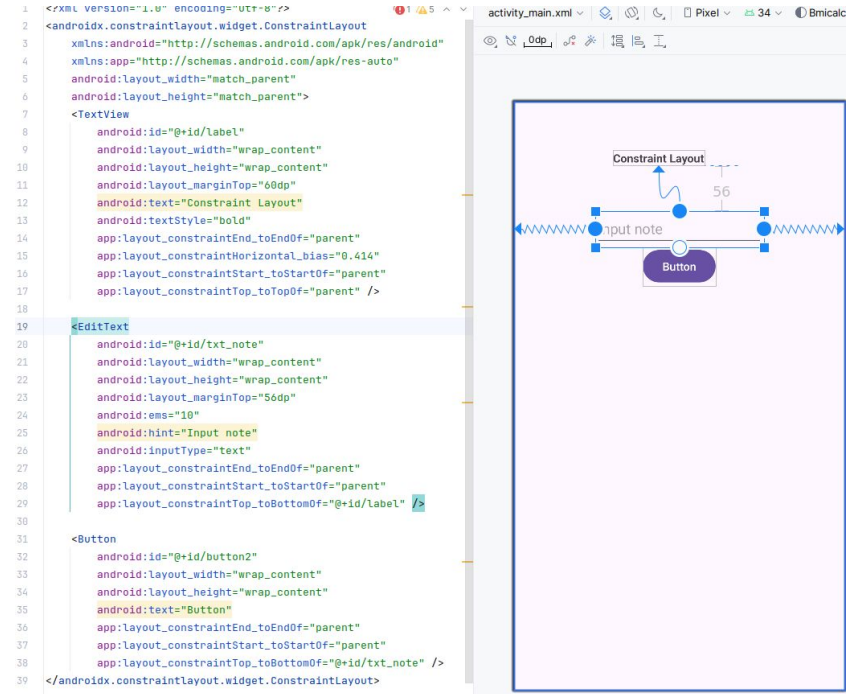
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="wrap_content"
    android:layout_gravity="center_horizontal"
    android:padding="20dp"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:fontFamily="serif"
        android:text="Linear Layout"
        android:textSize="28sp"
        android:textStyle="bold" />
    <EditText
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:hint="input note" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Click Me" />
</LinearLayout>
```

The visual preview on the right shows a vertical arrangement of three components: a bold serif text label "Linear Layout", a text input field with the placeholder "input note", and a purple button labeled "Click Me".

2. Constraint Layout

Description: Allows positioning and sizing of views in a flexible way using constraints.

Usage: Ideal for complex layouts where views are linked with constraints to each other or the parent.



The screenshot displays the Android Studio interface for editing an XML layout file named `activity_main.xml`. The left pane shows the XML code, and the right pane shows the visual design of the layout.

XML Code:

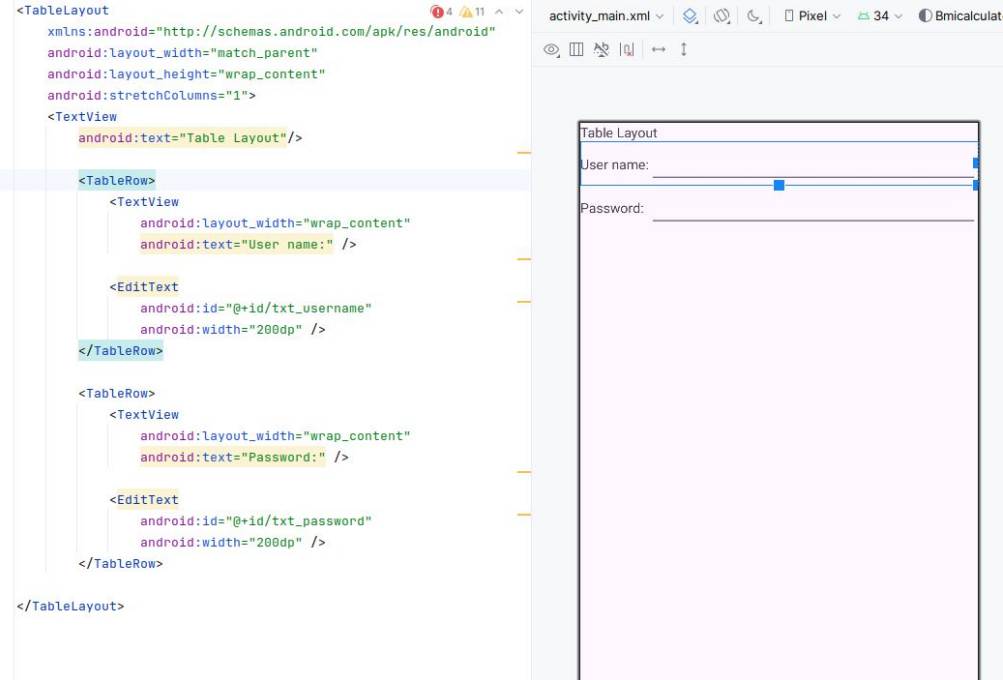
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:text="Constraint Layout"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.414"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/txt_note"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="56dp"
        android:ems="10"
        android:hint="Input note"
        android:inputType="text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/label" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txt_note" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Visual Design: The design view on the right shows a visual representation of the layout. It features a light blue background. At the top, a blue circle with the text "Constraint Layout" is positioned. Below it, a text input field with the hint "Input note" is shown. At the bottom, a blue button with the text "Button" is displayed. The components are connected by blue lines representing constraints, indicating their relative positions and sizes within the layout.

3. Table Layout

Allows you to organize views in a table format, with rows and columns. Each row in a `TableLayout` is defined by a `TableRow` element.

Within each `TableRow`, you can place views (like `TextView`, `Button`, etc.), and they will be organized in columns.



The screenshot displays the XML code for an `activity_main.xml` file in Android Studio. The code defines a `TableLayout` with two rows. The first row contains a `TextView` with the text "Table Layout". The second row contains two views: a `TextView` with the text "User name:" and an `EditText` field for the username. The third row contains two views: a `TextView` with the text "Password:" and an `EditText` field for the password. The visual preview on the right shows the layout as it will appear on a device screen, with the text and input fields arranged in a table format.

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1">
    <TextView
        android:text="Table Layout"/>
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:text="User name:" />
        <EditText
            android:id="@+id/txt_username"
            android:width="200dp" />
        </TableRow>
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:text="Password:" />
        <EditText
            android:id="@+id/txt_password"
            android:width="200dp" />
        </TableRow>
    </TableLayout>
```



View Component (part 2)

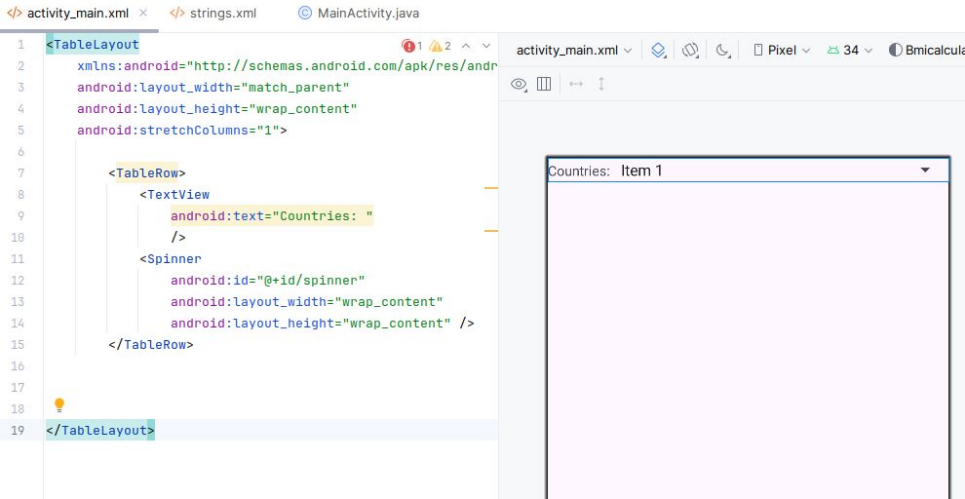
View Component

1. Spinner

Spinner provides a dropdown menu to select one item from a list.

Useful for selecting from a small set of items.

activity_main.xml

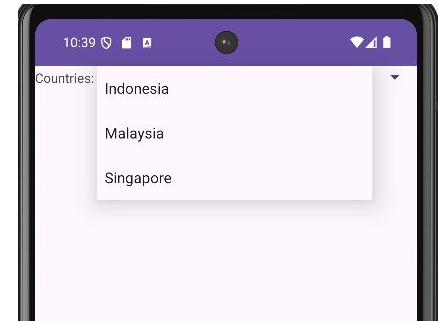


MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    Spinner spinner;

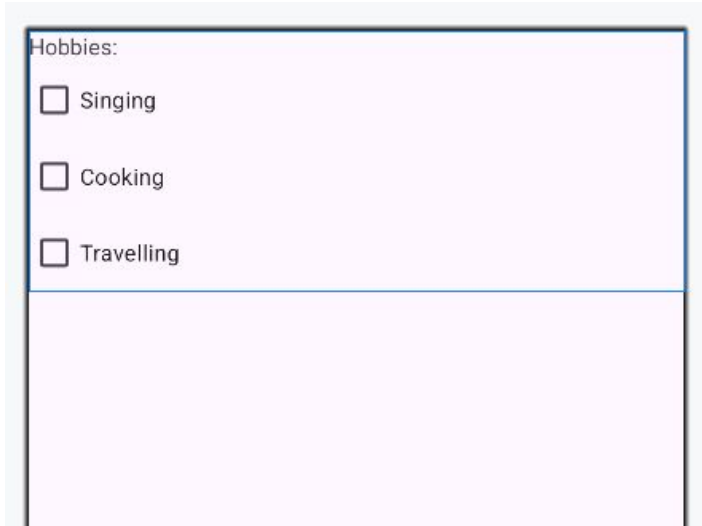
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        spinner = findViewById(R.id.spinner);

        String[] countries = {"Indonesia", "Malaysia", "Singapore"};
        ArrayAdapter<String> adapter = new ArrayAdapter<>(context: this, android.R.layout.simple_spinner_item, countries);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
    }
}
```



2. Checkbox

CheckBox allows users to select or deselect an option.
Multiple checkboxes can be checked at the same time.



```

<? activity_main.xml x <? strings.xml © MainActivity.java
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   android:layout_width="match_parent"
3   android:layout_height="wrap_content"
4   android:orientation="vertical"
5   android:stretchColumns="1">
6
7   <TextView
8     android:layout_width="wrap_content"
9     android:layout_height="wrap_content"
10    android:text="Hobbies: " />
11
12   <CheckBox
13     android:id="@+id/chk_singing"
14     android:layout_width="wrap_content"
15     android:layout_height="wrap_content"
16     android:text="Singin">
17
18   </CheckBox>
19
20   <CheckBox
21     android:id="@+id/chk_cooking"
22     android:layout_width="wrap_content"
23     android:layout_height="wrap_content"
24     android:text="Cooking">
25
26   <? />
27   <? />
28   <CheckBox
29     android:id="@+id/chk_travelling"
30     android:layout_width="wrap_content"
31     android:layout_height="wrap_content"
32     android:text="Travelling">
33
34   </CheckBox>
35
36   </LinearLayout>
37

```

3. Radio Button

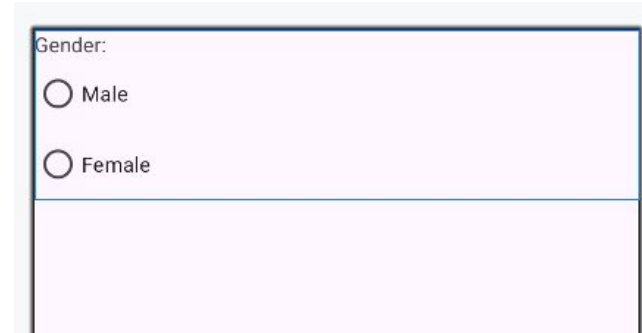
RadioButton allows single selection within a group.
Must be used within a RadioGroup for mutual exclusivity.

```
<?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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gender: " />

    <RadioGroup
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:id="@+id/radio_button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male">
        </RadioButton>
        <RadioButton
            android:id="@+id/radio_button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female">
        </RadioButton>
    </RadioGroup>

</LinearLayout>
```



4. Edit Text

```
<EditText
    android:id="@+id/txt_fullname"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:hint="full name" />
```

Handling User Input

```
EditText fullname;
fullName = findViewById(R.id.txt_fullname);
String fullnameValue = fullname.getText().toString();
```

findViewById: Finds the view component (EditText, Button, etc) by their IDs to interact with them in Java code.

getText().toString(): Retrieves the input text as a String (important for passing user input as data).

Handling button

```
Button btnSave = findViewById(R.id.btnSave);
btnSave.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Add your logic
    }
});
```

Handling Button

Atau bisa dengan menambahkan nama method pada onClick attribute

```
public void handleSave(View v) {
//add your logic here
}
```

```
<Button
    android:id="@+id/btn_save"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:onClick="handleSave"
    android:text="Calculate" />
```

onClick	handleSave	
outlineProvider		
overScrollMode		
> padding	[?, ?, ?, ?]	
paddingHorizontal		
paddingVertical		
password		
phoneNumber		

ACTIVITY and INTENT (Recap)

Activity

Definition: A single screen in an Android app with a user interface.

Purpose: Manages UI components and handles interactions for a screen in the app.

Lifecycle Overview: *onCreate, onStart, etc.*

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

Intent

Definition: An Intent is a messaging object you can use to request an action from another app component.

Types of Intents:

- **Explicit Intent:** Used to start a specific activity.
- **Implicit Intent:** Used to perform an action without specifying the exact app or component (e.g., open email, viewing a web page).

Intent: Creates an explicit intent targeting `SecondActivity`.

putExtra: Passes additional data (message) to the new activity.

startActivity: Starts the new activity.

Using Explicit Intents to Start Another Activity

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

        String resultString = "This is result";
        Intent resultIntent = new Intent(MainActivity.this,
SecondActivity.class);
        resultIntent.putExtra("result", resultString);
        startActivity(resultIntent);
    }
}
```

Receiving Data in Another Activity

```
public class SecondActivity extends AppCompatActivity {

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

        resultText = findViewById(R.id.txt_result);
        backButton=findViewById(R.id.btn_back);

        Intent intent = getIntent();
        String result = intent.getStringExtra("result");
        resultText.setText(result);

    }
```

getIntent(): Retrieves the intent that started this activity.

getStringExtra: Extracts the data passed from MainActivity.