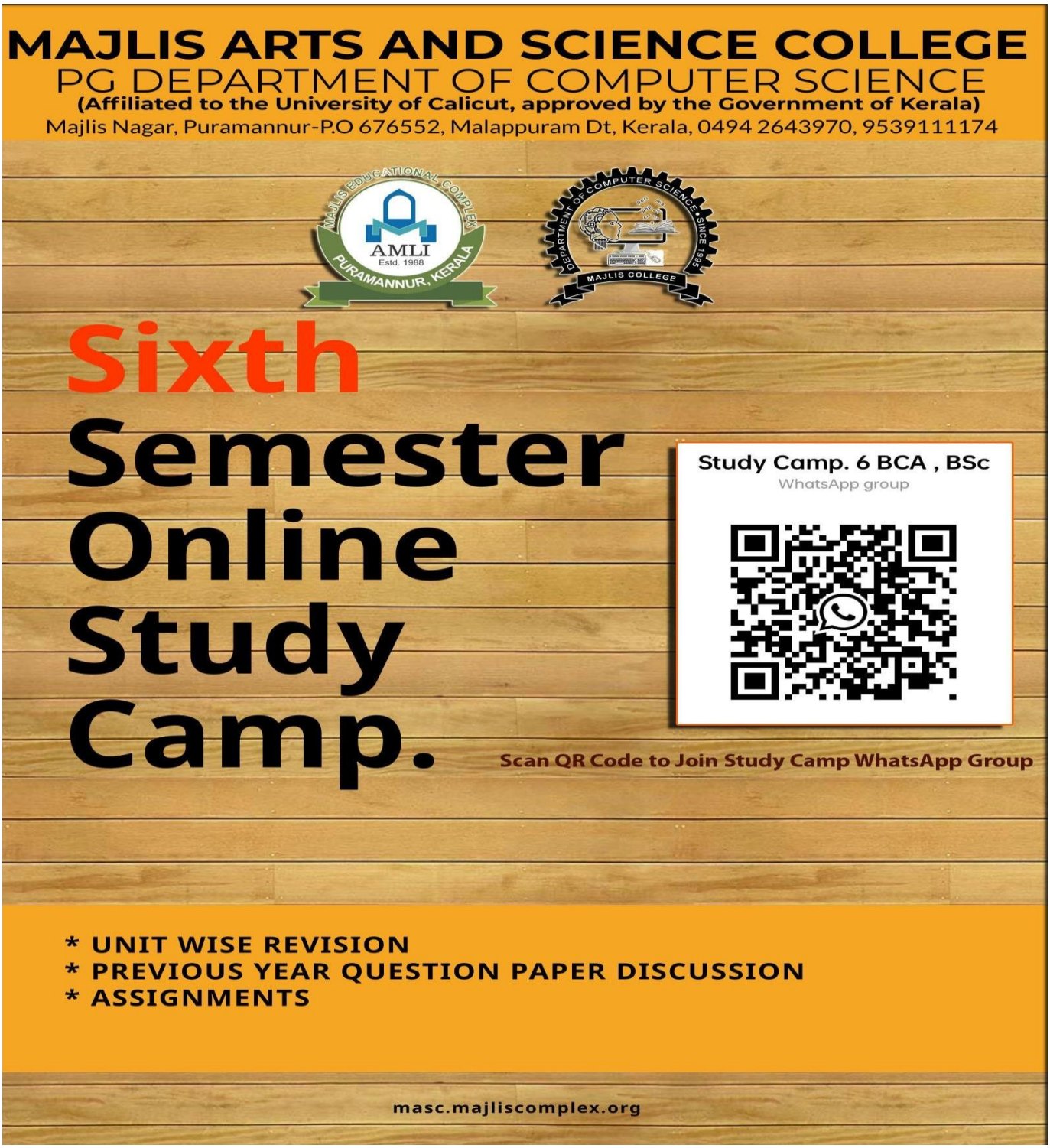


ANDROID PROGRAMMING

MODULE 3

MODULE 3

SYLLABUS

User interfaces development in android - building UI completely in code, UI using XML, UI in XML with code, Android's common controls - Text controls, button controls, checkbox control, radio button controls, image view, date and time controls, map view control.

understanding adapters, adapter views, list view, grid view, spinner control, gallery control, styles and themes.

Understanding layout managers - linear layout manager, table layout manager, relative layout manager, frame layout manager, grid layout manager.

Very Short Answer Questions (1 mark each)

1. Which controls are used to select date and time?

   Ans: Date picker and Time picker
2. Which method is called to clear all radio button within the radio group?

   Ans: clearCheck()
3. List any 5 basic controls in android?

   Ans:
   - TextView.
   - EditText.
   - AutoCompleteTextView.
   - Button.
   - ImageButton.
   - ToggleButton.
   - CheckBox.
   - RadioButton.
4. What is view in android?

   Ans : View is the basic building block of UI(User Interface) in android. View refers to the android. It can be an image, a piece of text, a button or anything that an android application can display.

5. What is the difference between android:gravity and android:layout_gravity?

Ans :android:layout_gravity attribute is used by child views to tell their parent how they want to be placed inside it, while android:gravity is used by the parent layout to tell the child views how they should be placed inside it.

Paragraph Questions

6. How to implement MapView in android?

Ans: Step 1. Download Android Studio

Follow the guides to download and install Android Studio.

Step 2. Install the Google Play services SDK

Add Google Play services to Android Studio.

Step 3. Create a Google Maps project

Step 4. Set up a Google Maps API key

Step 5. Look at the code

By default, the XML file that defines the app's layout is at res/layout/activity_maps.xml

Step 6.Connect an Android device

Step 7. Deploy and run your app

7. Show the syntax of button in xml.

Ans: A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.Depending on whether you want a button with text, an icon, or both, you can create the button in your layout in three ways:1}With text, using the Button class:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

2)With an icon, using the ImageButton class:

```
<ImageButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/button_icon"
   android:contentDescription="@string/button_icon_desc"
   ... />
```

3)With text and an icon, using the Button class with the android:drawableLeft attribute:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/button_text"
   android:drawableLeft="@drawable/button_icon"
   ... />
```

8. **What is the purpose of directory layout?**

   Ans : Within an Android project structure, the most frequently edited folders are: src - Java source files associated with your project. ... res - Resource files associated with your project. All graphics, strings, layouts, and other resource files are stored in the resource file hierarchy under the res directory.

9. **What is layout? Give brief description of any 3 layouts.**

   Ans : layouts are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using View/ViewGroup objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.

   Android **LinearLayout** is a view group that aligns all children in either vertically or horizontally.

   Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

Android **RelativeLayout** enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

An Absolute Layout lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.

Frame Layout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.

Android ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

Android GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter.

10. **Explain spinner control and its purpose by giving suitable example**.

Ans: Android Spinner is a view similar to the dropdown list which is used to select one option from the list of options. It provides an easy way to select one item from the list of items and it shows a dropdown list of all values when we click on it. The default value of the android spinner will be the currently selected value and by using Adapter we can easily bind the items to the spinner objects.

```
<?xml version="1.0" encoding="utf-8"?>
<!--Constraint layout which contain Spinner widget-->
<android.support.constraint.ConstraintLayoutxmlns: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="com.geeksforgeeks.Spinner.MainActivity">
<Spinner
android:id="@+id/coursesspinner"
android:layout_height="50dp"
android:layout_width="160dp"      android:layout_marginEnd="10dp"
```

android:layout_marginStart="10dp" android:layout_marginBottom="10dp"
android:layout_marginTop="10dp"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"/></android.support.constraint.ConstraintLayout>

Essay Questions

10.Briefly explain the following layout managers

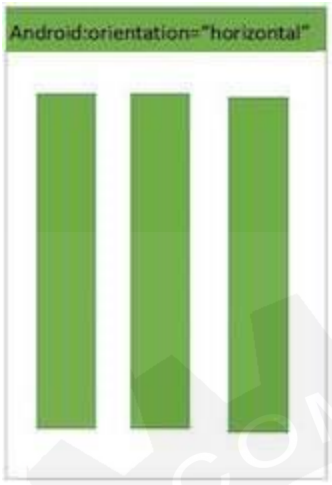
a. Linear layout
b. Table layout
c. Relative layout
d. Grid layout

Ans: **Linear Layout**

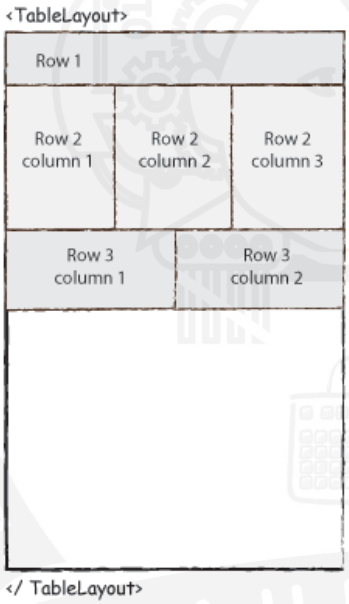
LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute. All children of a LinearLayout are stacked one after the other, so a vertical list will only have one child per row, no matter how wide they are, and a horizontal list will only be one row high (the height of the tallest child, plus padding). A LinearLayout respects *margin*s between children and the *gravity* (right, center, or left alignment) of each child.

```
<?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:paddingLeft="16dp"
   android:paddingRight="16dp"
   android:orientation="vertical" >

<EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/to" />
    </LinearLayout>
```



b. **Table layout**-Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.



android:id-This is the ID which uniquely identifies the layout.

android:collapseColumns-This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.

android:shrinkColumns-The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.

android:stretchColumns-The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5. Following will be the content of **res/layout/activity_main.xml** file –

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <TableRow
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
      android:text="Time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="1" />

    <TextClock
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textClock"
      android:layout_column="2" />

  </TableRow>
```

c. Relative layout

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

android:id-This is the ID which uniquely identifies the layout.

android:gravity-This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

android:ignoreGravity-This indicates what view should not be affected by gravity.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:paddingLeft="16dp"
  android:paddingRight="16dp" >

  <EditText
    android:id="@+id/name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:hint="@string/reminder" />

  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/name">

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Button"
      android:id="@+id/button" />

  </LinearLayout>

</RelativeLayout>
```

d. Grid Layout

In Android GridLayout, we can specify the number of columns and rows that the grid will have. We can customize the GridLayout according to our requirements, like setting the size, color or the margin for the Layout.

So, a GridLayout basically places its children in a rectangular grid. This grid has a set of a number of thin lines that separate the view area into cells. Suppose you have a grid of **N** columns, then we will have **N+1** grid indices that would be starting from **0.**

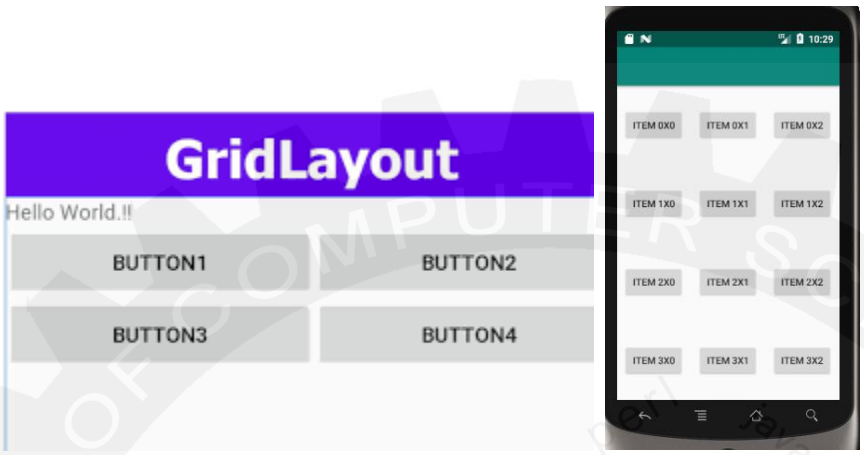
```
<GridLayout
xmlns:android= “http://schemas.android.com/apk/res/android”
android: id= “@+id/GridLayout”
android: layout_width= “match_parent”
android: layout_height= “match_parent”
</GridLayout>
```

To add children to Android GridLayout, we can specify them in the **<GridLayout>** tag. To do it, you can consider the following code:

```
<GridLayout
    xmlns:android=     “http://schemas.android.com/apk/res/android”
    android: id= “@+id/GridLayout”
    android: layout_width= “match_parent”
    android: layout_height= “match_parent”
    android:column= “1”
    android:row= “1”
        <RadioButton
            android:id= “@+id/but1”
            android:layout_column= “0”
            android:layout_row= “0”
            android:text= “Radio Button”
        >
        <RadioButton
            android:id= “@+id/but2”
            android:layout_column= “0”
            android:layout_row= “1”
            android:text= “Radio Button”
```

>

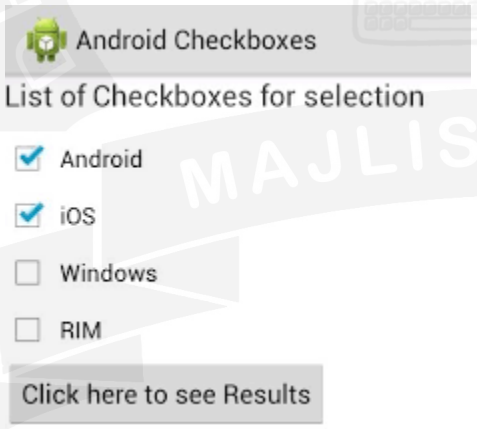
</GridLayout>



11. What are checkboxes and radio buttons? Write a sample code to show the use of check boxes and radio buttons in android.

Ans: **a.Check Box**

Checkboxes allow the user to select one or more options from a set. Typically, you should present each checkbox option in a vertical list.



create each checkbox option, create a CheckBox in your layout. Because a set of checkbox options allows the user to select multiple items, each checkbox is managed separately and you must register a click listener for each one. When the user selects a checkbox, the CheckBox object receives an on-click event.

To define the click event handler for a checkbox, add the android:onClick attribute to the <CheckBox> element in your XML layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
```

```
            // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

b.Radio button

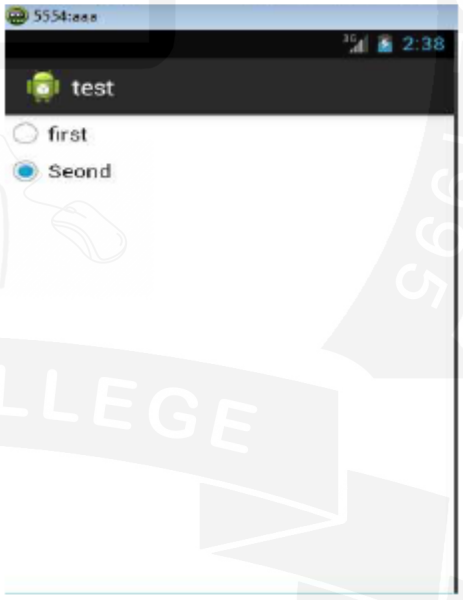
RadioButton is a two states button which is either checked or unchecked. If a single radio button is unchecked, we can click it to make checked radio button. Once a radio button is checked, it cannot be marked as unchecked by user.

RadioButton is generally used with *RadioGroup*. RadioGroup contains several radio buttons, marking one radio button as checked makes all other radio buttons as unchecked.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <RadioGroup
        android:id="@+id/rd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:id="@+id/r1"
            android:text="first"/>
        <RadioButton
            android:id="@+id/r2"
            android:text="Seond"/>

    </RadioGroup>

</LinearLayout>
```




```
Rg=(RadioGroup)findViewById(R.id.rd);
Rd1=(RadioButton)findViewById(R.id.r1);
Rg.setOnCheckedChangeListener()
```