1. Button

Android Button represents a push-button. The android.widget.Button is a subclass of TextView class and CompoundButton is the subclass of Button class.

There are different types of buttons in android such as RadioButton, ToggleButton, CompoundButton etc.

Android Button Example with Listener

Here, we are going to create two text fields and one button for the sum of two numbers. If the user clicks the button, the sum of two input values is displayed on the Toast.

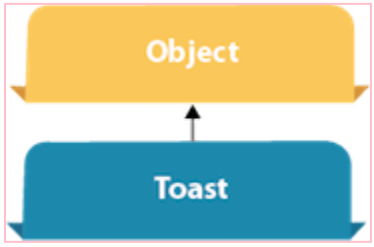
We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

```
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
          //code
        }
});
```

```
<Button
      android:onClick="methodName"
/>
```

2. Toast

Android Toast Example



Android Toast can be used to display information for a short period of time. A toast contains a message to be displayed quickly and disappears after some time.

The android.widget.Toast class is the subclass of java.lang.Object class.

You can also create custom toast as well for example toast displaying images. You can visit the next page to see the code for custom toast.

Toast class

Toast class is used to show notification for a particular interval of time. After some time it disappears. It doesn't block the user interaction.

Constants of Toast class

There are only 2 constants of Toast class which are given below.

Constant	Description
public static final int LENGTH_LONG	displays view for the long duration of time.

public static final int LENGTH_SHORT	displays view for the short duration of time.

Methods of Toast class

The widely used methods of Toast class are given below.

Method	Description
public static Toast makeText(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void setMargin (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Android Toast Example

1. Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show();

Another code:

1. Toast toast=Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT);
2. toast.setMargin(50,50);
3. toast.show();

3. Custom Toast

You are able to create custom toast in android. So, you can display some images like congratulations or loss on the toast. It means you are able to customize the toast now.

```
//Creating the LayoutInflater instance
    LayoutInflater li = getLayoutInflater();
    //Getting the View object as defined in the customtoast.xml file
    View layout = li.inflate(R.layout.customtoast,(ViewGroup)
findViewById(R.id.custom_toast_layout));

    //Creating the Toast object
    Toast toast = new Toast(getApplicationContext());
    toast.setDuration(Toast.LENGTH_SHORT);
    toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
    toast.setView(layout);//setting the view of custom toast layout
    toast.show();
```

4. ToggleButton

Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Android ToggleButton and Switch both are the subclasses of CompoundButton class.

ToggleButton class provides the facility of creating the toggle button.

XML Attributes of ToggleButton class

The 3 XML attributes of ToggleButton class.

XML Attribute	Description
android:disabledAlpha	The alpha to apply to the indicator when disabled.
android:textOff	The text for the button when it is not checked.
android:textOn	The text for the button when it is checked.

Methods of ToggleButton class

The widely used methods of ToggleButton class are given below.

Method	Description
CharSequence getTextOff()	Returns the text when the button is not in the checked state.

CharSequence getTextOn()	Returns the text for when button is in the checked state.
void setChecked(boolean checked)	Changes the checked state of this button.

```
//Getting the ToggleButton and Button instance from the layout xml file
        toggleButton1=(ToggleButton)findViewById(R.id.toggleButton);
        toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
        buttonSubmit=(Button)findViewById(R.id.button);

        //Performing action on button click
        buttonSubmit.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View view) {
                StringBuilder result = new StringBuilder();
                result.append("ToggleButton1 : ").append(toggleButton1.getText());
                result.append("\nToggleButton2 : ").append(toggleButton2.getText());
                //Displaying the message in toast
                Toast.makeText(getApplicationContext(),
result.toString(),Toast.LENGTH_LONG).show();
            }
```

5. CheckBox

Android CheckBox is a type of two state button either checked or unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.

Android CheckBox class

The android.widget.CheckBox class provides the facility of creating the CheckBoxes.

Methods of CheckBox class

There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	Description
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

```
//Getting instance of CheckBoxes and Button from the activty_main.xml file
    pizza=(CheckBox)findViewById(R.id.checkBox);
    coffe=(CheckBox)findViewById(R.id.checkBox2);
    burger=(CheckBox)findViewById(R.id.checkBox3);
    buttonOrder=(Button)findViewById(R.id.button);
```

```
//Applying the Listener on the Button click
buttonOrder.setOnClickListener(new View.OnClickListener(){

    @Override
    public void onClick(View view) {
        int totalamount=0;
        StringBuilder result=new StringBuilder();
        result.append("Selected Items:");
        if(pizza.isChecked()){
            result.append("\nPizza 100Rs");
            totalamount+=100;
        }
        if(coffe.isChecked()){
            result.append("\nCoffe 50Rs");
            totalamount+=50;
        }
        if(burger.isChecked()){
            result.append("\nBurger 120Rs");
            totalamount+=120;
        }
        result.append("\nTotal: "+totalamount+"Rs");
        //Displaying the message on the toast
```

```
        Toast.makeText(getApplicationContext(), result.toString(),
Toast.LENGTH_LONG).show();
    }
```

6. Custom CheckBox

Android provides facility to customize the UI of view elements rather than default.

You are able to create custom CheckBox in android. So, you can add some different images of checkbox on the layout.

```
 cb1=(CheckBox)findViewById(R.id.checkBox3);
    cb2=(CheckBox)findViewById(R.id.checkBox4);
    button=(Button)findViewById(R.id.button);


    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        StringBuilder sb=new StringBuilder("");


        if(cb1.isChecked()){
          String s1=cb1.getText().toString();
          sb.append(s1);
        }


        if(cb2.isChecked()){
          String s2=cb2.getText().toString();
```

```
            sb.append("\n"+s2);


        }
        if(sb!=null && !sb.toString().equals("")){
            Toast.makeText(getApplicationContext(), sb,
Toast.LENGTH_LONG).show();


        }
        else{
            Toast.makeText(getApplicationContext(),"Nothing Selected",
Toast.LENGTH_LONG).show();
        }
       }
```

7. RadioButton

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

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

```
        radioGroup=(RadioGroup)findViewById(R.id.radioGroup);
    }
    public void onclickbuttonMethod(View v){
        int selectedId = radioGroup.getCheckedRadioButtonId();
        genderradioButton = (RadioButton) findViewById(selectedId);
        if(selectedId==-1){
            Toast.makeText(MainActivity.this,"Nothing selected",
Toast.LENGTH_SHORT).show();
        }
        else{
            Toast.makeText(MainActivity.this,genderradioButton.getText(),
Toast.LENGTH_SHORT).show();
        }

    }
}
```

8. Dynamic RadioButton

Instead of creating a RadioButton through drag and drop from palette, android also facilitates you to create it programmatically (dynamically). For creating a dynamic RadioButton, we need to use android.view.ViewGroup.LayoutParams which configures the width and height of views and implements *setOnCheckedChangeListener()* method of *RadioGroup* class.

```
public class MainActivity extends AppCompatActivity {
    RadioGroup rg;
```

```
    RelativeLayout rl;
    RadioButton rb1,rb2;

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

        rg = new RadioGroup(this);
        rl = (RelativeLayout) findViewById(R.id.relativeLayout);
        rb1 = new RadioButton(this);
        rb2 = new RadioButton(this);

        rb1.setText("Male");
        rb2.setText("Female");
        rg.addView(rb1);
        rg.addView(rb2);
        rg.setOrientation(RadioGroup.HORIZONTAL);

        RelativeLayout.LayoutParams params = new
RelativeLayout.LayoutParams((int)
LayoutParams.WRAP_CONTENT,(int)LayoutParams.WRAP_CONTENT);
        params.leftMargin =150;
```

```
        params.topMargin = 100;


        rg.setLayoutParams(params);
        rl.addView(rg);


        rg.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton radioButton = (RadioButton) findViewById(checkedId);

Toast.makeText(getApplicationContext(),radioButton.getText(),Toast.LENGTH_L
ONG).show();
            }
        });
    }
```

9. Custom RadioButton

Rather than the default user interface of android RadioButton, we can also implement a custom radio button. Custom RadioButton makes the user interface more attractive.

```
public class MainActivity extends AppCompatActivity {
    Button button;
    RadioButton genderradioButton;
```

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

        radioGroup=(RadioGroup)findViewById(R.id.radioGroup);
    }


    public void onclickbuttonMethod(View v){
        int selectedId = radioGroup.getCheckedRadioButtonId();
        genderradioButton = (RadioButton) findViewById(selectedId);
        if(selectedId==-1){
            Toast.makeText(MainActivity.this,"Nothing selected",
Toast.LENGTH_SHORT).show();
        }
        else{
            Toast.makeText(MainActivity.this,genderradioButton.getText(),
Toast.LENGTH_SHORT).show();
        }

    }
```

}

10. AlertDialog

Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

Methods of AlertDialog class

Method	Description
public AlertDialog.Builder setTitle(CharSequence)	This method is used to set the title of AlertDialog.
public AlertDialog.Builder setMessage(CharSequence)	This method is used to set the message for AlertDialog.
public AlertDialog.Builder setIcon(int)	This method is used to set the icon over AlertDialog.

```
public class MainActivity extends AppCompatActivity {
    Button closeButton;
    AlertDialog.Builder builder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        closeButton = (Button) findViewById(R.id.button);
        builder = new AlertDialog.Builder(this);
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                //Uncomment the below code to Set the message and title from the
strings.xml file
                builder.setMessage(R.string.dialog_message)
.setTitle(R.string.dialog_title);

                //Setting message manually and performing action on button click
                builder.setMessage("Do you want to close this application ?")
                        .setCancelable(false)
                        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                finish();
                                Toast.makeText(getApplicationContext(),"you choose yes
action for alertbox",
                                Toast.LENGTH_SHORT).show();
```

```
                    }
                })
                .setNegativeButton("No", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        //  Action for 'NO' Button
                        dialog.cancel();
                        Toast.makeText(getApplicationContext(),"you choose no
action for alertbox",
                        Toast.LENGTH_SHORT).show();
                    }
                });
            //Creating dialog box
            AlertDialog alert = builder.create();
            //Setting the title manually
            alert.setTitle("AlertDialogExample");
            alert.show();
        }
    });
}
```

12. Spinner

Android Spinner is like the combox box of AWT or Swing. It can be used to display the multiple options to the user in which only one item can be selected by the user.

Android spinner is like the drop down menu with multiple values from which the end user can select only one value.

Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner.

Android Spinner class is the subclass of AsbSpinner class.

```
public class MainActivity extends AppCompatActivity implements
        AdapterView.OnItemSelectedListener {
    String[] country = { "India", "USA", "China", "Japan", "Other"};


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       //Getting the instance of Spinner and applying OnItemSelectedListener on it
        Spinner spin = (Spinner) findViewById(R.id.spinner);
        spin.setOnItemSelectedListener(this);


        //Creating the ArrayAdapter instance having the country list
        ArrayAdapter aa = new
ArrayAdapter(this,android.R.layout.simple_spinner_item,country);


aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //Setting the ArrayAdapter data on the Spinner
        spin.setAdapter(aa);
```

```
    }


    //Performing action onItemSelected and onNothing selected
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long
id) {
        Toast.makeText(getApplicationContext(),country[position] ,
Toast.LENGTH_LONG).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}
```

13. AutoCompleteTextView

Android AutoCompleteTextView completes the word based on the reserved words, so no need to write all the characters of the word.

Android AutoCompleteTextView is an editable text field, it displays a list of suggestions in a drop down menu from which the user can select only one suggestion or value.

Android AutoCompleteTextView is the subclass of EditText class. The MultiAutoCompleteTextView is the subclass of AutoCompleteTextView class

13. Switch

A Switch is a two-state toggle widget. Users can drag the switch "thumb" back and forth to select either of two options or simply tap the switch to toggle between options.

The text property controls the text of the switch label. The textOn and textOff properties control the text of the thumb. The textAppearance property and the related setTypeface() methods control the typeface and style of the switch label. The switchTextAppearance property and the related setSwitchTypeface() methods control the typeface and style of the thumb text.

XML attributes

android:showText	Whether to draw on/off text.
android:splitTrack	Whether to split the track and leave a gap for the thumb drawable.
android:switchMinWidth	Minimum width for the switch component.
android:switchPadding	Minimum space between the switch and caption text.
android:switchTextAppearance	TextAppearance style for text displayed on the switch thumb.
android:textOff	Text to use when the switch is in the unchecked/"off" state.
android:textOn	Text to use when the switch is in the checked/"on" state.
android:textStyle	Style (normal, bold, italic, bold|italic) for the text.

android:thumb	Drawable to use as the "thumb" that switches back and forth.
android:thumbTextPadding	Amount of padding on either side of text within the switch thumb.
android:thumbTint	Tint to apply to the thumb.
android:thumbTintMode	Blending mode used to apply the thumb tint.
android:track	Drawable to use as the "track" that the switch thumb slides within.
android:trackTint	Tint to apply to the track.
android:trackTintMode	Blending mode used to apply the track tint.
android:typeface	Typeface (normal, sans, serif, monospace) for the text.

Inherited XML attributes

From class android.widget.CompoundButton

From class android.widget.TextView

From class android.view.View

Inherited constants

From class android.widget.TextView

From class android.view.View

Inherited fields

From class android.view.View

Public constructors

Switch(Context context)

Construct a new Switch with default styling.

Switch(Context context, AttributeSet attrs)

Construct a new Switch with default styling, overriding specific style attributes as requested.

Switch(Context context, AttributeSet attrs, int defStyleAttr)

Construct a new Switch with a default style determined by the given theme attribute, overriding specific style attributes as requested.

Switch(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes)

Construct a new Switch with a default style determined by the given theme attribute or style resource, overriding specific style attributes as requested.

Public methods

void	draw(Canvas c) Manually render this view (and all of its children) to the given Canvas.

void	drawableHotspotChanged(float x, float y) This function is called whenever the view hotspot changes and needs to be propagated to drawables or child views managed by the view.
CharSequence	getAccessibilityClassName() Return the class name of this object to be used for accessibility purposes.
int	getCompoundPaddingLeft() Returns the left padding of the view, plus space for the left Drawable if any.
int	getCompoundPaddingRight() Returns the right padding of the view, plus space for the right Drawable if any.
boolean	getShowText()
boolean	getSplitTrack() Returns whether the track should be split by the thumb.
int	getSwitchMinWidth() Get the minimum width of the switch in pixels.
int	getSwitchPadding() Get the amount of horizontal padding between the switch and the associated text.

CharSequence	getTextOff() Returns the text displayed when the button is not in the checked state.
CharSequence	getTextOn() Returns the text displayed when the button is in the checked state.
Drawable	getThumbDrawable() Get the drawable used for the switch "thumb" - the piece that the user can physically touch and drag along the track.
int	getThumbTextPadding() Get the horizontal padding around the text drawn on the switch itself.
BlendMode	getThumbTintBlendMode()
ColorStateList	getThumbTintList()
PorterDuff.Mode	getThumbTintMode()
Drawable	getTrackDrawable() Get the drawable used for the track that the switch slides within.
BlendMode	getTrackTintBlendMode()
ColorStateList	getTrackTintList()

PorterDuff.Mode	getTrackTintMode()
void	jumpDrawablesToCurrentState() Call Drawable.jumpToCurrentState() on all Drawable objects associated with this view.
void	onMeasure(int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height.
boolean	onTouchEvent(MotionEvent ev) Implement this method to handle touch screen motion events.
void	setChecked(boolean checked) Changes the checked state of this button.
void	setShowText(boolean showText) Sets whether the on/off text should be displayed.
void	setSplitTrack(boolean splitTrack) Specifies whether the track should be split by the thumb.
void	setSwitchMinWidth(int pixels) Set the minimum width of the switch in pixels.

void	setSwitchPadding(int pixels) Set the amount of horizontal padding between the switch and the associated text.
void	setSwitchTextAppearance(Context context, int resid) Sets the switch text color, size, style, hint color, and highlight color from the specified TextAppearance resource.
void	setSwitchTypeface(Typeface tf) Sets the typeface in which the text should be displayed on the switch.
void	setSwitchTypeface(Typeface tf, int style) Sets the typeface and style in which the text should be displayed on the switch, and turns on the fake bold and italic bits in the Paint if the Typeface that you provided does not have all the bits in the style that you specified.
void	setTextOff(CharSequence textOff) Sets the text displayed when the button is not in the checked state.
void	setTextOn(CharSequence textOn) Sets the text displayed when the button is in the checked state.
void	setThumbDrawable(Drawable thumb) Set the drawable used for the switch "thumb" - the piece that the user can physically touch and drag along the track.

void	setThumbIcon(Icon icon) Set the drawable used for the switch "thumb" - the piece that the user can physically touch and drag along the track - to the specified Icon.
void	setThumbResource(int resId) Set the drawable used for the switch "thumb" - the piece that the user can physically touch and drag along the track.
void	setThumbTextPadding(int pixels) Set the horizontal padding around the text drawn on the switch itself.
void	setThumbTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the tint specified by setThumbTintList(android.content.res.ColorStateList)} to the thumb drawable.
void	setThumbTintList(ColorStateList tint) Applies a tint to the thumb drawable.
void	setThumbTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the tint specified by setThumbTintList(android.content.res.ColorStateList)} to the thumb drawable.
void	setTrackDrawable(Drawable track) Set the drawable used for the track that the switch slides within.

void	setTrackIcon(Icon icon) Set the drawable used for the track that the switch slides within to the specified Icon.
void	setTrackResource(int resId) Set the drawable used for the track that the switch slides within.
void	setTrackTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the tint specified by setTrackTintList(android.content.res.ColorStateList)} to the track drawable.
void	setTrackTintList(ColorStateList tint) Applies a tint to the track drawable.
void	setTrackTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the tint specified by setTrackTintList(android.content.res.ColorStateList)} to the track drawable.
void	toggle() Change the checked state of the view to the inverse of its current state

Protected methods

void	drawableStateChanged() This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.

int[]	onCreateDrawableState(int extraSpace) Generate the new Drawable state for this view.
void	onDraw(Canvas canvas) Implement this to do your drawing.
void	onLayout(boolean changed, int left, int top, int right, int bottom) Called from layout when this view should assign a size and position to each of its children.
boolean	verifyDrawable(Drawable who) If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying.

Inherited methods

From class android.widget.CompoundButton

From class android.widget.Button

From class android.widget.TextView

From class android.view.View

From class java.lang.Object

From interface android.widget.Checkable

From interface android.view.ViewTreeObserver.OnPreDrawListener

From interface android.graphics.drawable.Drawable.Callback

From interface android.view.KeyEvent.Callback

From interface android.view.accessibility.AccessibilityEventSource

XML attributes

android:showText

Whether to draw on/off text.

Maybe a boolean value, such as "true" or "false".

14. ListView
15. Custom ListView (Adding Images, sub-title)
16. RatingBar
17. WebView
18. SeekBar
19. DatePcker
20. TimePicker
21. Analog clock and Digital clock
22. ProgressBar
23. Vertical Scrollview
24. Horizontal Scrollview’
25. ImageSwitcher
26. Image Slider
27. ViewStub
28. TabLayout
29. TabLayout with FrameLayout
30. SearchView

31.Searchview on Toolbar
32. EditText with TextWatcher

Activity and Intents

1. Activity Life Cycle
2. Implicit Intent
3. Explicit Intent
4. Android Fragments
5. Share App Data

Android Menus

1. Option Menu
2. Context Menu
3. Pop Up Menu

Android Service

Android AlaramManager

Android Storage

1. Android Preferences
2. Internal Storage
3. External Storage

Android SQLLITE

Android Multimedia

Android Speech

Android Telephony