UNIT:4

Advanced UI Programming

SYLLABUS

- 4.1 Event driven Programming in Android (Text Edit, Button clicked etc.)
- 4.2 Activity Lifecycle of Android

Procedural vs. Event-Driven Programming

- *Procedural programming* is executed in procedural order.
- In event-driven programming, code is executed upon activation of events.

EVENTS

- An *event* can be defined as a type of signal to the program that something has happened.
- The event is generated by external user actions such as mouse movements, mouse clicks, and keystrokes, or by the operating system, such as a timer

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.

4.1 Event driven Programming in Android

Register a Listener

- Using an Anonymous Inner Class
- Activity class implements the Listener interface.
- Using Layout file.

Event Listeners Registration Using an Anonymous Inner Class

```
// Create an anonymous implementation of OnClickListener
private OnClickListener myListener = new OnClickListener()
 {
    public void onClick(View v)
 {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button obj_button = (Button)findViewById(R.id.button1);
    // Register the onClick listener with the implementation above
   obj_button.setOnClickListener(myListener);
    ...
}
```

Registration Using the Activity Implements Listener Interface

```
public class ExampleActivity extends Activity implements
    OnClickListener {
        protected void onCreate(Bundle savedValues) {
            ...
            Button obj_button =
    (Button)findViewById(R.id.button1);
            obj_button.setOnClickListener(this);
        }

        // Implement the OnClickListener callback
        public void onClick(View v) {
          // do something when the button is clicked
        }
        ...
    }
```

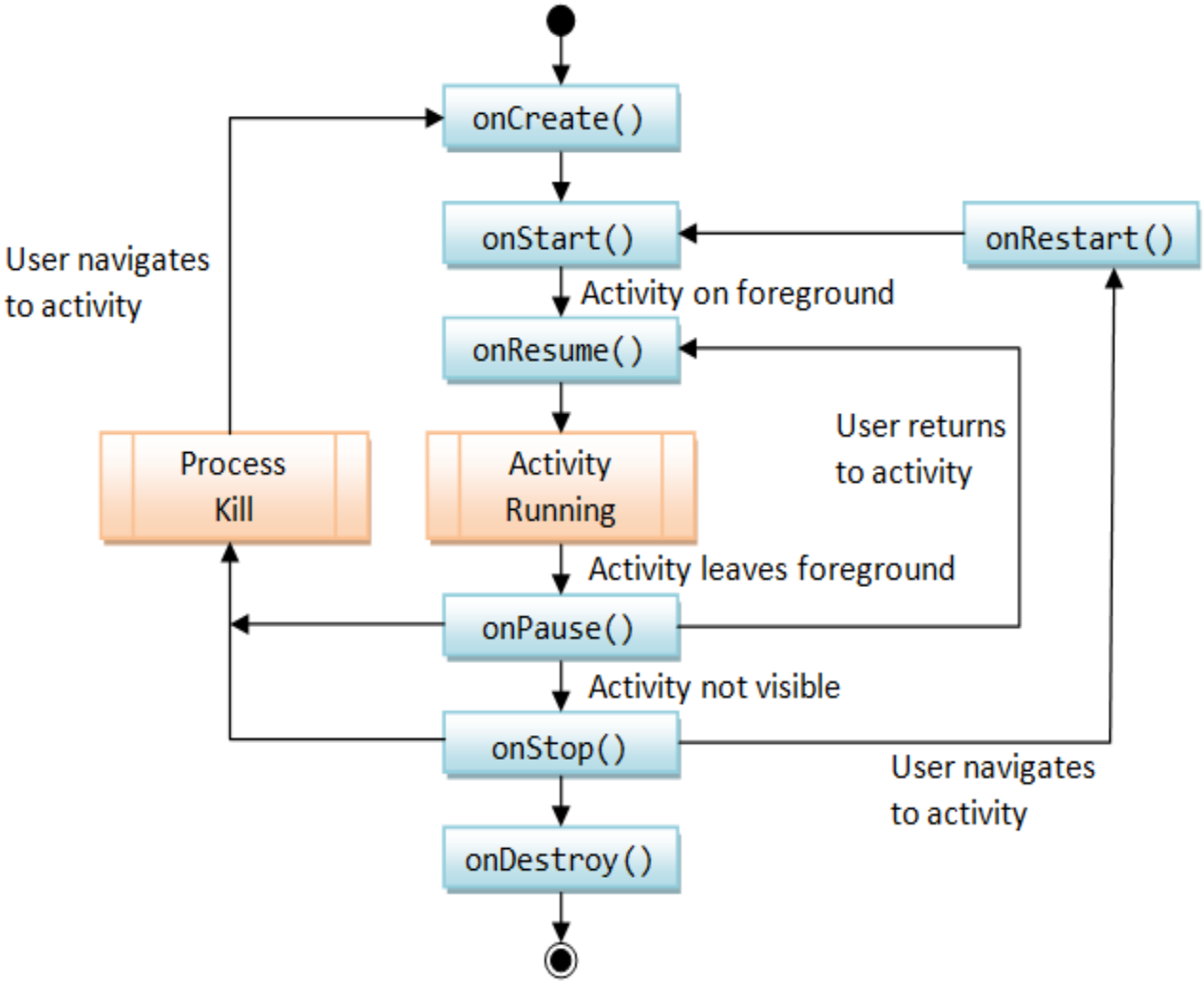
Registration Using Layout file activity_main.xml

```
public class MainActivity extends Activity
{
   @Override
  protected void onCreate(Bundle savedInstanceState)
 {       super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  Button obj_button = (Button)findViewById(R.id.button1);
    }
 //--- Implement the event handler for the button.
  @Override
 public void Myhandler(View v)
 {                    }
}
```

In XML file:

```
<Button     android:id="@+id/button1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
  android:text="click_me"
 android:onClick="Myhandler"/>
```

4.2 Activity Lifecycle of Android



EXAMPLE

```
public class MainActivity extends Activity
{

    protected void onCreate(Bundle savedInstance
   State)
 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("lifecycle","onCreate invoked");
    }
```

```
    protected void onStart()
{
        super.onStart();
         Log.d("lifecycle","onStart invoked");
    }

    protected void onResume()
{
        super.onResume();
         Log.d("lifecycle","onResume invoked");
    }
```

```
    protected void onPause()
{
        super.onPause();
         Log.d("lifecycle","onPause invoked");
    }
    protected void onStop()
{
        super.onStop();
         Log.d("lifecycle","onStop invoked");
    }
```

```
    protected void onRestart()
{
        super.onRestart();
        Log.d("lifecycle","onRestart invoked");
    }
    protected void onDestroy()
 {
        super.onDestroy();
        Log.d("lifecycle","onDestroy invoked");
    }
}
```

IMPORTANT QUESTIONS.

1. What is Event Driven Programming? Explain in brief.
2. Write a short note on event listener and event handler.
3. How to register listener Explain in brief.
4. Give example of activity life cycle.
5. Explain exception handling concept.