Practical-7: To implement the concept of multithreading using Android Service class.

Introduction:

Multi-threading is defined as a feature through which we can run two or more concurrent threads of a process. In this a process, the common data is shared among all these threads also known as sub-processes exclusively. In android there are many ways through which multi-threading can be established in the application.

Objective:

- Understanding the basic concept of multithreading.
- Understanding of Handler class in android
- Understanding of Runnable Interface.

Multi-Threading In Android:

Multi-Threading in Android is a unique feature through which more than one threads execute together without hindering the execution of other threads. Multi-Threading in Android is not different from conventional multi-Threading. A class can be thought of as a process having its method as it's sub-processes or threads. All these methods can run concurrently by using feature of Multi-Threading. In android, multi-Threading can be achieved through the use of many in-built classes. Out of them, Handler class is most commonly used.

Handler Class In Android:

Handler class come from the Package android.os.Handler package and is most commonly used for multi-threading in android. Handler class provide sending and receiving feature for messages between different threads and handle the thread execution which is associated with that instance of Handler class. In android class, every thread is associated with an instance of Handler class and it allows the thread to run along with other threads and communicate with them through messages.

Runnable Interface:

Runnable interface is used in multi-threading to be called in a loop when the thread starts. It is a type of thread that executes the statement in its body or calls other methods for a specified or infinite number of times.This runable

interface is used by the Handler class to execute the multi-threading, i.e., to execute one or more thread in specified time.Runnable is an interface which is implemented by the class desired to support multithreading and that class must implements it’s abstract method public void run().Run() method is the core of multithreading as it includes the statement or calls to other methods that the thread needs to be made for multithreading.

```
class ClassName implements Runnable
{
        @override
Public void run()
{
                Body of method
}
}
```

Runnable interface can also be used by using adapter class as explained below:

```
Runnable runnableObject =new Runnable()
{
@override
Public void run()
{
};
}
```

Steps Involved in making project on Multi-Threading:

Step 1: Make a new android project Multi-Threading in Eclipse and create a package named com.nkm.thread in it.

Step 2:
Listing 1: Create an XML file main.xml if not already created in layout folder and paste the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <RelativeLayout
        android:id="@+id/firstlayout"
        android:layout_width="fill.parent"
        android:layout_height="wrap.content"
        android:gravity="center"
        android:layout_marginTop="80dp">

        <TextView
            android:id="@+id/display"
            android:layout_width="wrap.content"
            android:layout_height="wrap.content"
            android:text="Button will appear after 10 seconds" />
    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/secondlayout"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/firstlayout"
        android:gravity="center">


         <TextView
             android:id="@+id/timer"
             android:layout_width="wrap.content"
             android:layout_height="wrap.content"
             android:gravity="center.horizontal"
             android:text="12"
             android:layout_marginTop="80dp"
             android:textSize="36dp"/>
     </RelativeLayout>

     <RelativeLayout
         android:id="@+id/thirdlayout"
         android:layout_width="fill.parent"
         android:layout_height="wrap.content"
         android:layout_below="@+id/secondlayout"
         android:gravity="center">

         <Button
             android:id="@+id/clickme"
             android:layout_width="wrap.content"
             android:layout_height="wrap.content"
             android:text="Click_me"
             android:visibility="false"
             android:layout_marginTop="100dp"/>
     </RelativeLayout>

</RelativeLayout>
```

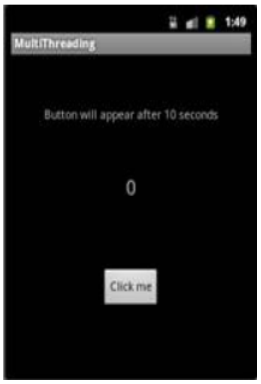
Step 3:
Listing 2: Make a Java file named MultiThreadingActivity if not already created in com.nkm.thread package and paste the following code:

```
package com.nkm.thread;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.Button;
import android.widget.TextView;

public class MultiThreadingActivity extends Activity {
        Handler hand = new Handler();
        Button clickme;
        TextView timer;

        @Override
        public void onCreate(Bundle_savedInstance) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                timer = (TextView) findViewById(R.id_timer);
                clickme = (Button) findViewById(R.id_clickme);
                hand.postDelayed(run, 1000);
        }

        Runnable run = new Runnable() {
                @Override
                public void run() {
                        updateTime();

                }
        };


       public void updateTime() {
              timer.setText("" + (Integer.parseInt(timer.getText().toString()) - 1));
              if (Integer.parseInt(timer.getText().toString()) == 0) {
                     clickme.setVisibility(0);
              } else {
                     hand.postDelayed(run, 1000);
              }
       }
}
```

So, on the whole it creates a thread that executes in a loop until certain condition met.

Output-1

Output-2



Output-3