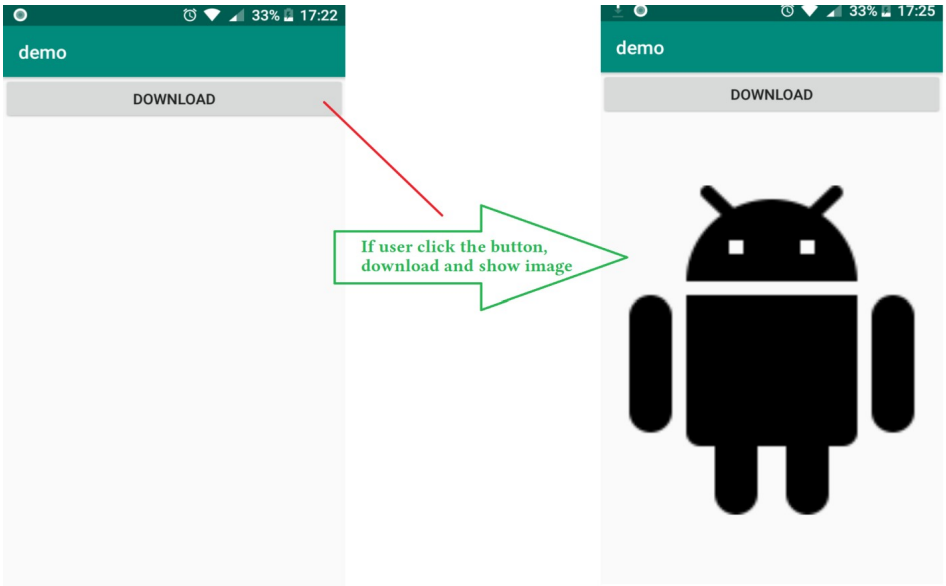
MAIN THREAD

Main Thread
The main thread is started on application start and stays alive during the lifetime of the process.
Main thread is in charge of dispatching events and rendering user interface.

There are many thing we need to be careful when working with main thread:
- If we perform long tasks, such as I/O on main thread, application may lag or thow exception.
- If application has not responded to an input or BroadcastReceiver within 5 seconds, Application Not Responding will appear.
- If UI element is access from outside the main thread, exception *CalledFromWrongThreadException* will be thrown.

Practice
Lets build an app that downloads an image from internet and show it on a view.

The activity include a button, and an image view.
If user clicks download button, app will download image and show it.



Now, Lets implement it.
Add permission to Manifest

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Add button, image view to activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="download"
```

```
    android:onClick="onUserClick"/>

  <ImageView
    android:id="@+id/image_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

</LinearLayout>
```

Implement MainActivity.java

```
package invistd.demo;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class MainActivity extends AppCompatActivity {

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

  void onUserClick(View v) {
    Thread thread_download = new Thread() {
      @Override
      public void run() {
        try {
          //download image data
          URL url = new URL("https://www.materialui.co/materialIcons/action/android_black_144x144.png");
          HttpURLConnection connection = (HttpURLConnection) url.openConnection();
          connection.setRequestMethod("GET");
          connection.connect();
          InputStream inStream = connection.getInputStream();

          //convert data to bitmap
          final Bitmap bitmap = BitmapFactory.decodeStream(inStream);

          final ImageView imageView = findViewById(R.id.image_view);

          //show bitmap on imageView
          runOnUiThread(new Runnable() {
            @Override
            public void run() {
```

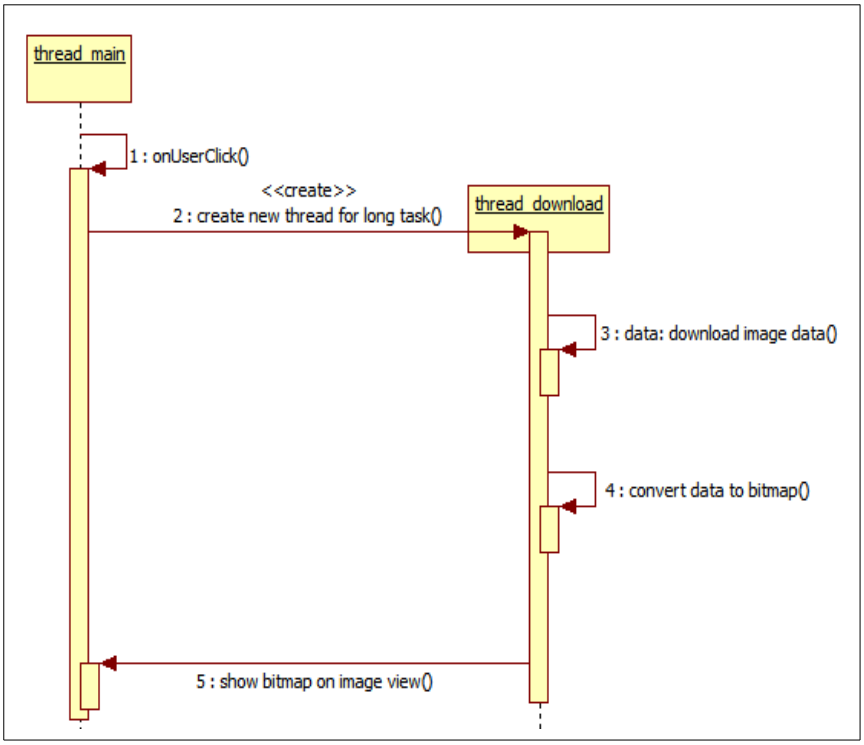
```
                    imageView.setImageBitmap(bitmap);
                }
            });


        } catch (Exception e) {
            Log.e("maxter","exception:" + e.toString());
            e.printStackTrace();
        }
    }
};

thread_download.start();
}
}
```

When user click download button, method *onUserClick* is called.

The method performs a **network task.**
If the network task is performed on main thread, NetworkOnMainThreadException will be thrown.
So, we create a new thread to do this task.

After converted data, we will show the image.

However, image view is an UI element.
If we access an UI element from a non-main thread, CalledFromWrongThreadException will be thrown.
So, we use *runOnUiThread* to delegate the action for main thread.



//continue: native thread and main thread