PEMROGRAMAN APLIKASI PERANGKAT BERGERAK (MOBILE)

Background Task

K Candra Brata

andra.course@gmail.com

Please Wait...

Connecting to server...

25%

25/100

Thread

http://developer.android.com/guide/components/processes-and-threads.html

Concurrency

- ❑ **Concurrency** is the ability to run several parts of a program or several programs in parallel, which means at the same time.
- ❑ Running multiple tasks at the same time means they are running **asynchronously**.
- ❑ If time consuming tasks can be performed asynchronously or in parallel, this improve the overall performance and the interactivity of your program.
- ❑ **Java supports concurrency by allowing programs to create multiple <u>threads</u>.**

Thread

- A **Thread** is a concurrent unit of execution.

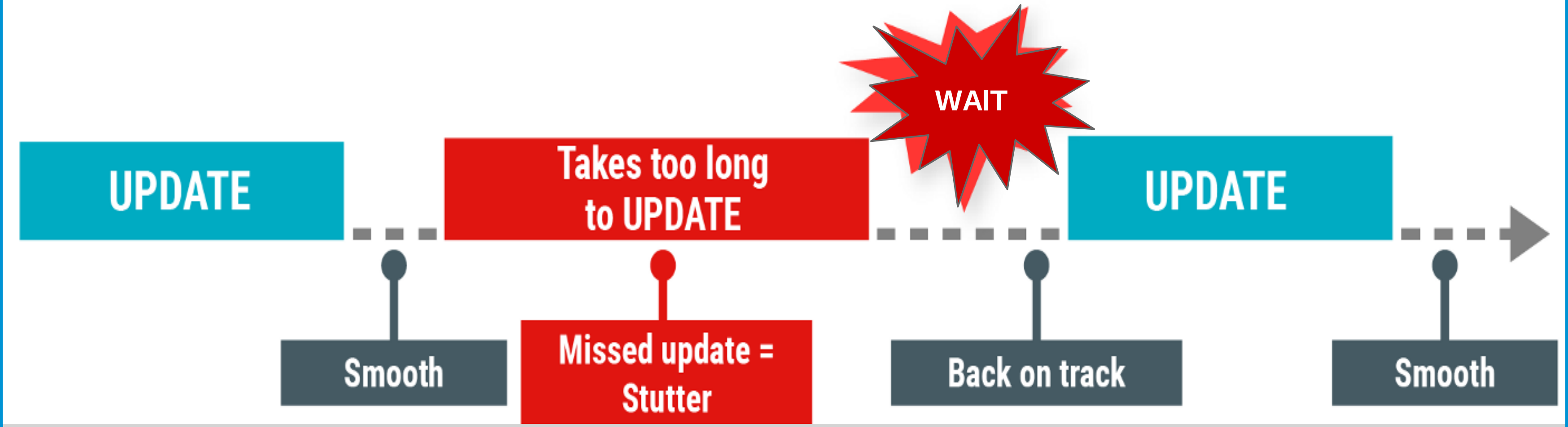
- When you first start your Android Activity, the **main thread**, which is also called the **UI thread**, is automatically created.
- The one single UI thread is in charge of dispatching and managing all the event-driven activities in the main layout, and this includes the drawing events.

- For instance, if you touch a button on screen, the UI thread dispatches the touch event to the button's handler, sets its pressed state and posts an invalidate request to the event queue. When a **Handler** is triggered, it runs on the UI thread and dequeues the request and notifies the component to redraw itself.

The main thread

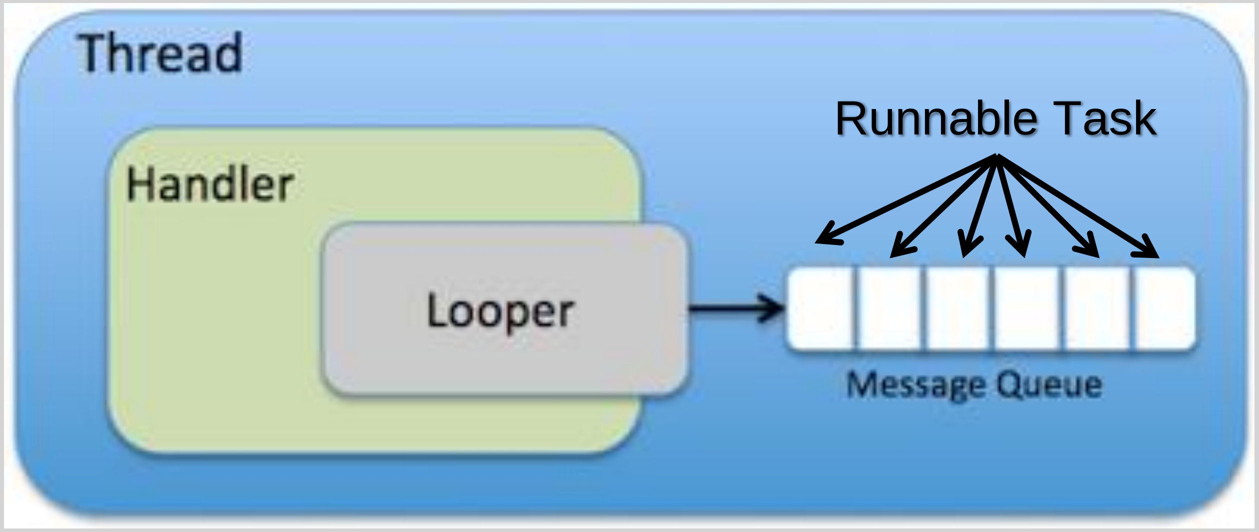
The Main thread must be fast !!

- ❑ IF hardware updates screen every 16 milliseconds
- ❑ UI thread has 16 ms to do all its work
- ❑ If it takes too long, app stutters or hangs



Thread

- ❑ Android collects all events in a queue and processed an instance of the Looper class.



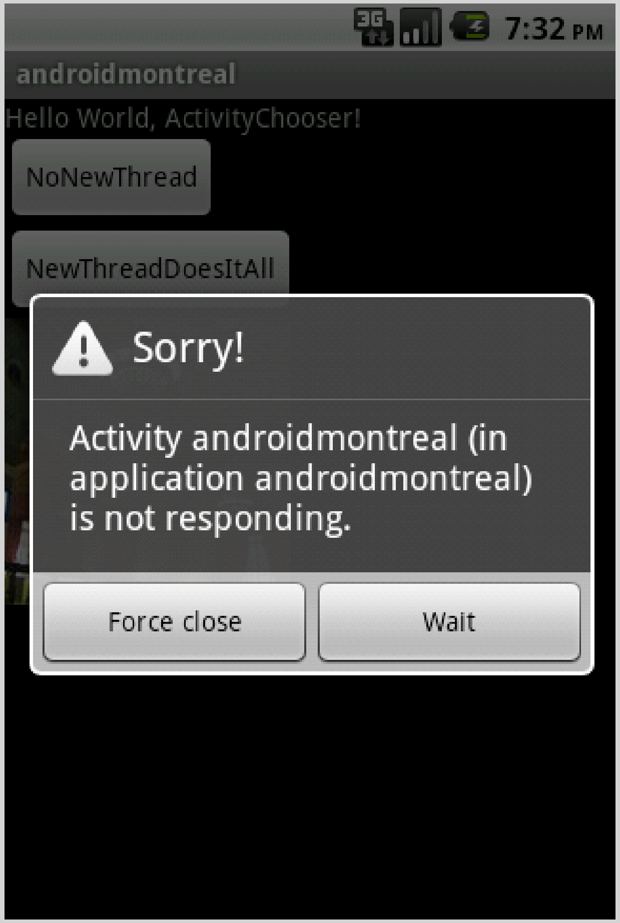
- ❑ **If the programmer does not use any concurrency constructs, all code of an Android application runs in the main thread and every statement is executed after each other.**

Why should I care about Threading?

Android will show an **"ANR"** error if a View does not return from handling an event within 5 seconds.
(if the UI thread is blocked by some code that running in the "main thread“, prohibits UI events from being handled .)

This means that any long-running code **should run** in a **background thread**.

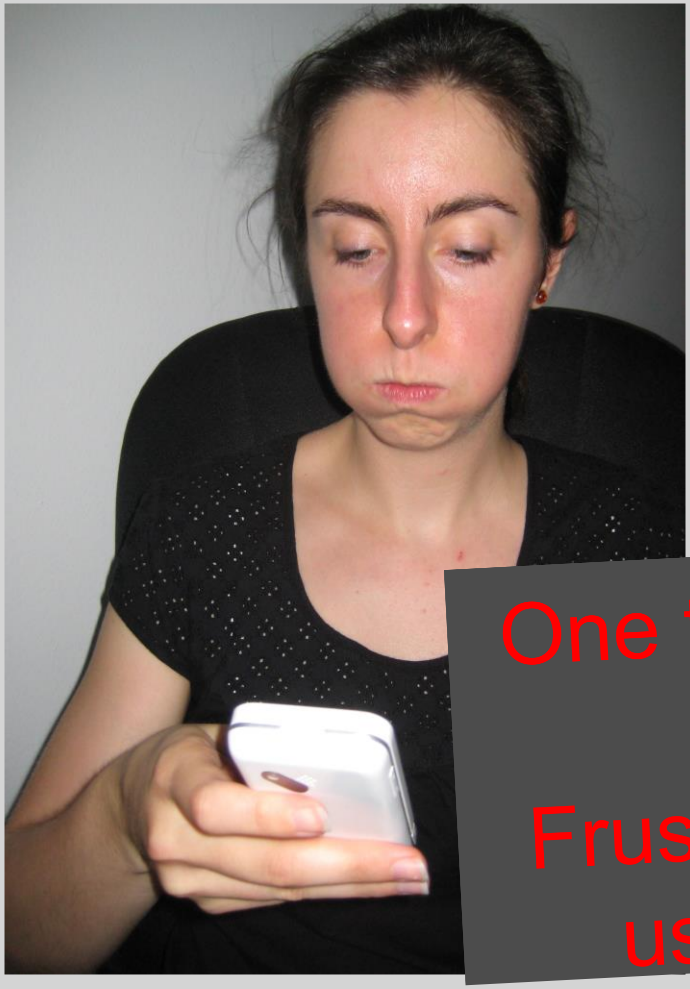
Why should I care about Threading?



Hello World, ActivityChooser!

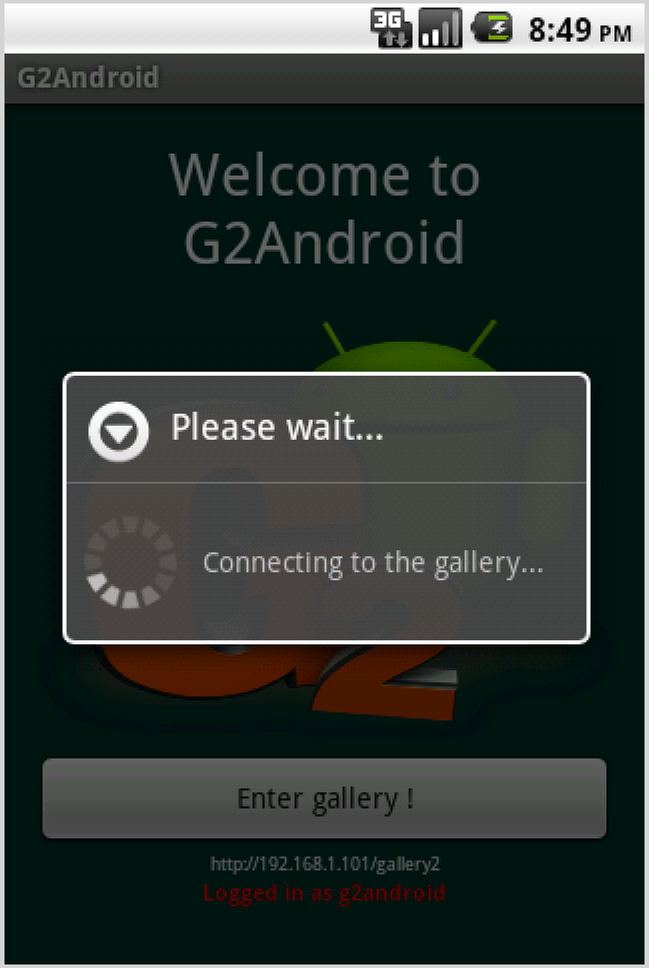
Sorry!

Activity androidmontreal (in application androidmontreal) is not responding.



One thread = Frustrated user !

Why should I care about Threading?



Welcome to G2Android

Please wait...

Connecting to the gallery...

http://192.168.1.101/gallery2

Logged in as g2android

Many threads

=

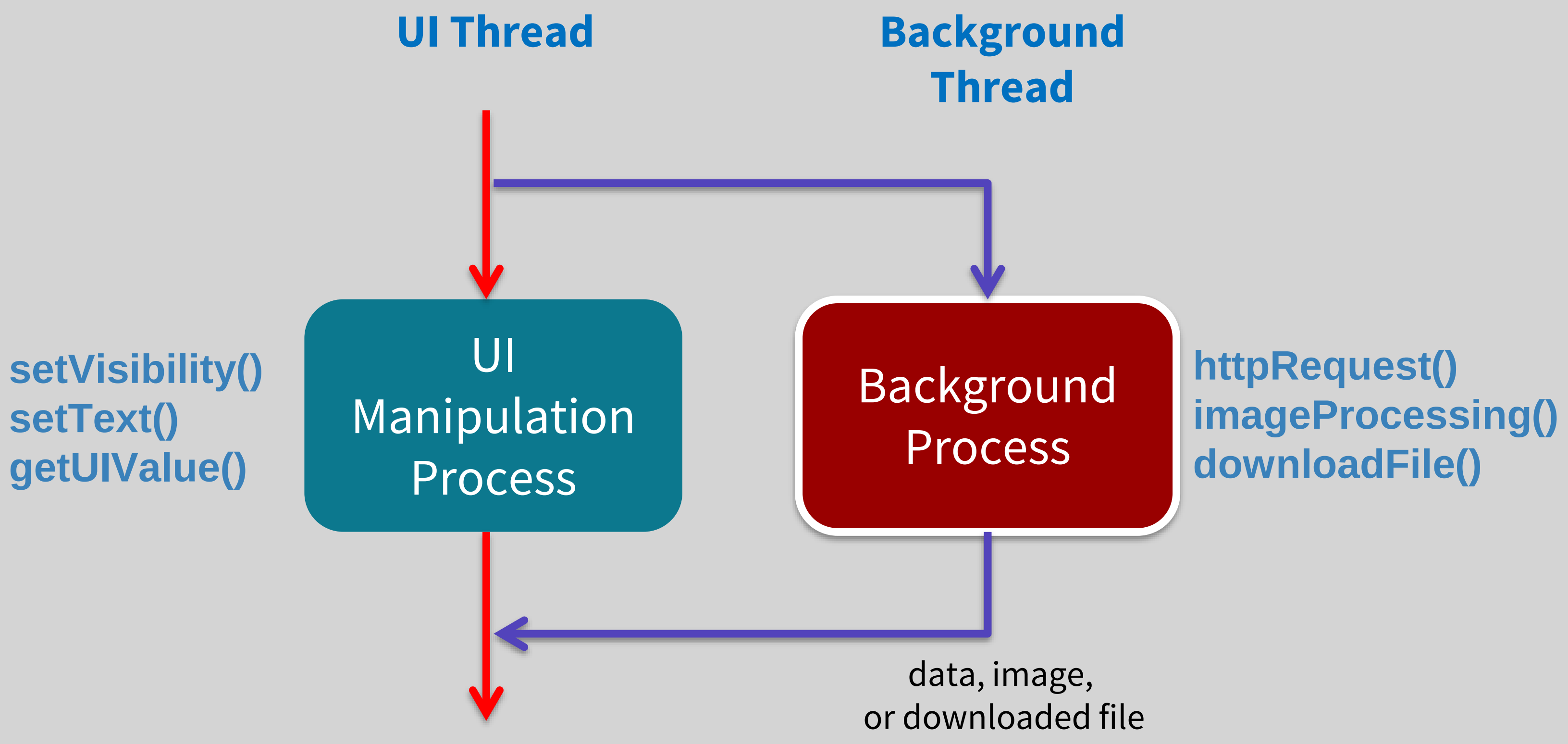
Happy user

Two rules for Android threads

- **Do not block the UI thread**
  - Complete all work in less than 5 seconds for each screen
  - Run slow non-UI work on a non-UI thread

- **Do not access the Android UI toolkit from outside the UI Thread / Main Thread**
  - Do UI work only on the UI thread

UI Thread and Background Basic Process



Once a Thread has finished its process, I can not be re-started

Thread

- ❑ There are two ways to execute code in a new thread.

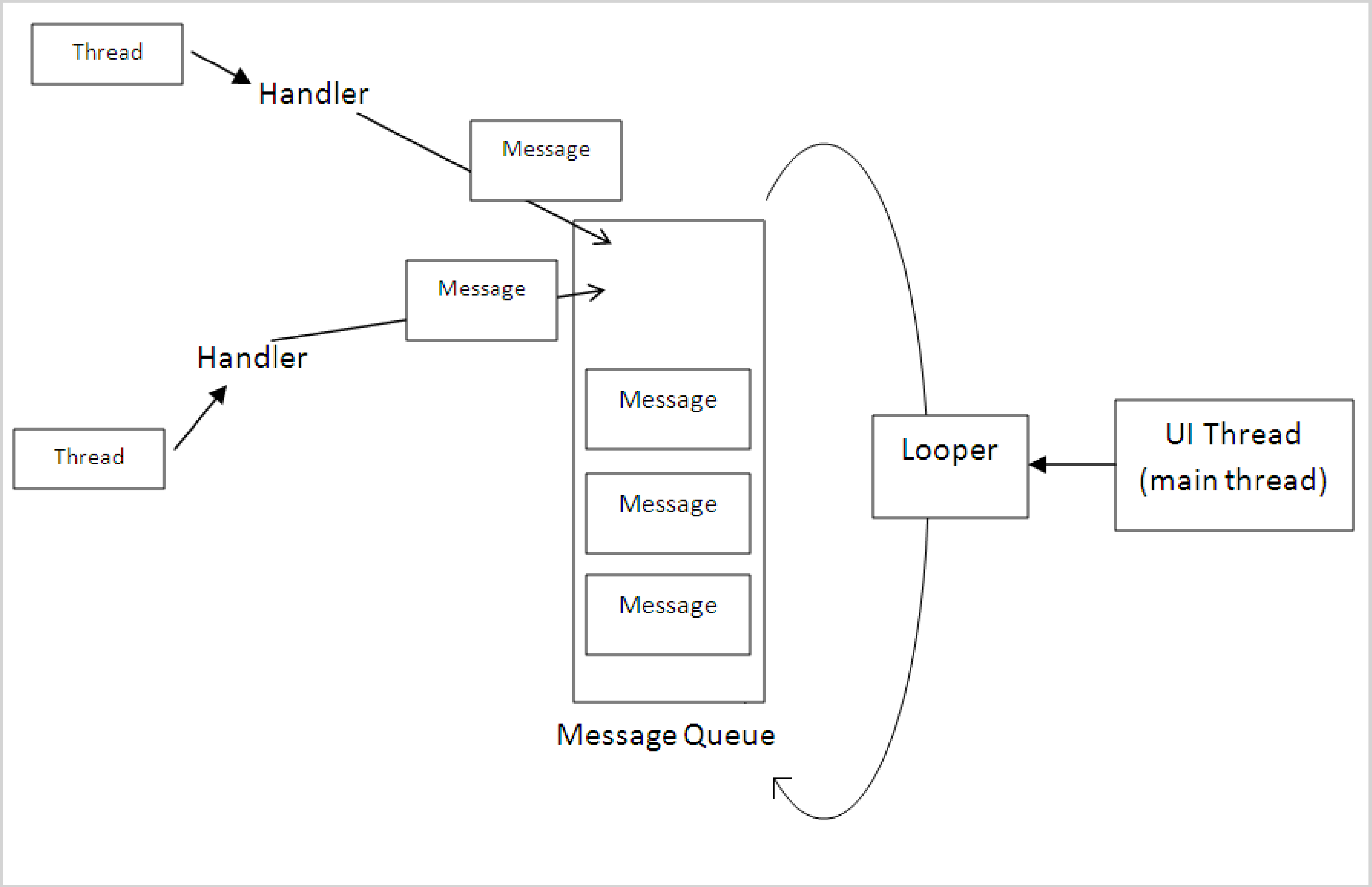
1. Subclass Thread and overriding its **run()** method, or
2. Construct a new Thread and pass a **Runnable** param to the constructor.

- ❑ In either case, the **start()** method must be called to actually execute the new Thread.

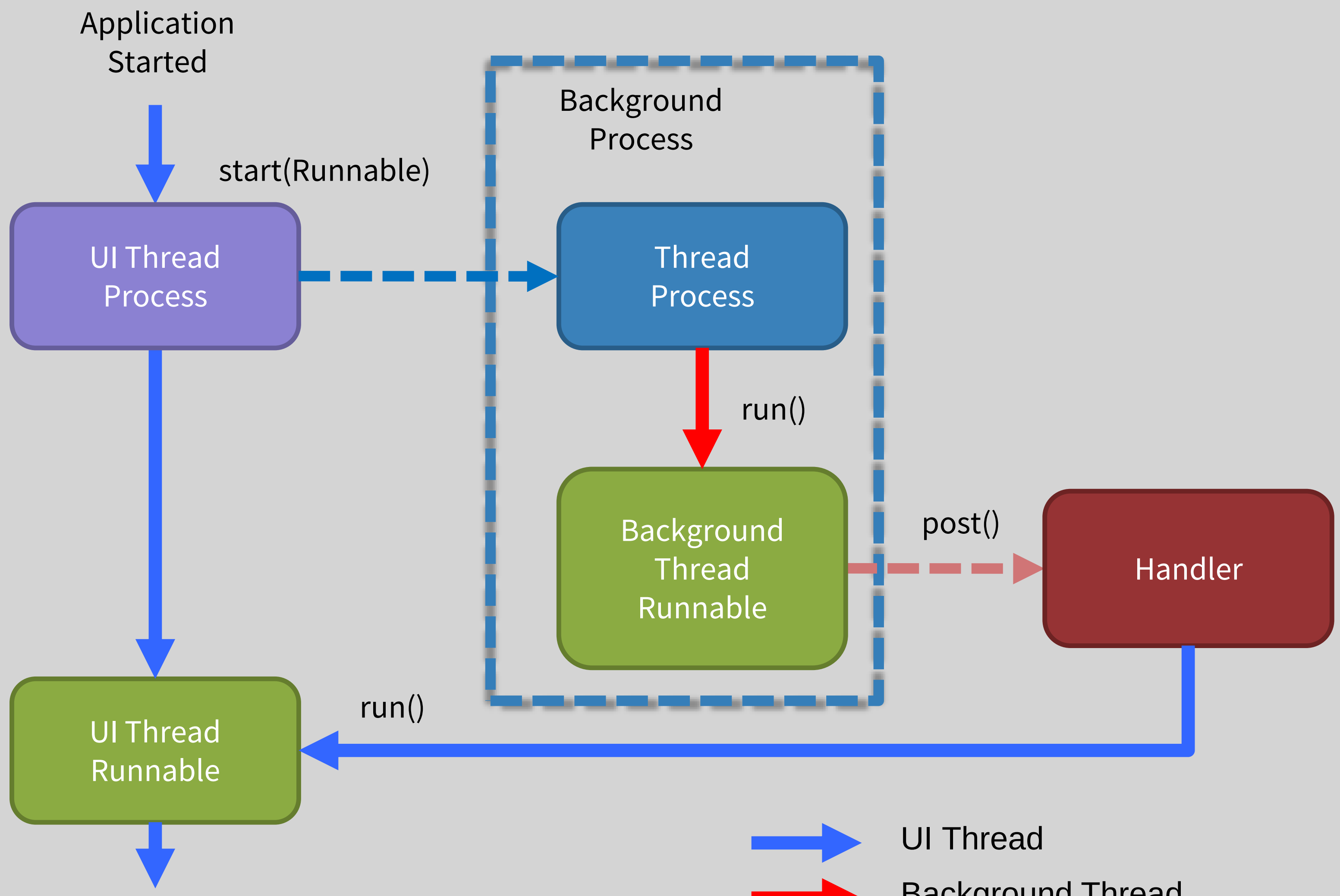
Android Handler

- ❑ **Do not access to the UI toolkit outside of the Main Thread.**
- ❑ **Background threads are not allowed to modify UI elements.**
- ❑ You need to pass data/information to mainThread
- ❑ An **Android Handler** allows you to send and process targeted Messages on the Android Activity’s **main thread**.
- ❑ Android offers **several ways to access the UI thread** from other threads.
  1. Handler.post(Runnable)
  2. Activity.runOnUiThread(Runnable)
  3. The View class allows you to post objects of type **Runnable** via the **post()** method.
     - View.post(Runnable)
     - View.postDelayed(Runnable, long)

Android Handler



Regular Thread Runnable



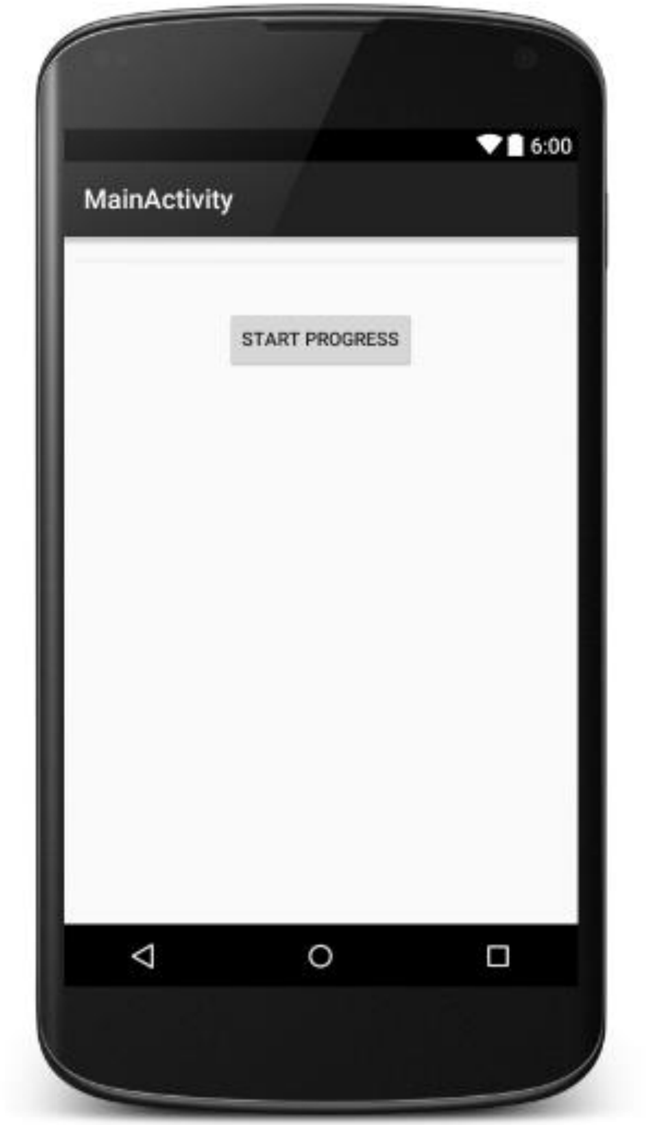
activity_main.XML

```
<?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:orientation="vertical" >

   <ProgressBar
      android:id="@+id/progressBar1"
      style="?android:attr/progressBarStyleHorizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:indeterminate="false"
      android:max="10"
      android:padding="4dip" >
   </ProgressBar>

   <TextView
      android:id="@+id/textView1"
      android:layout_gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="" >
   </TextView>
   <Button
      android:id="@+id/button1"
      android:layout_gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="startProgress"
      android:text="Start Progress" >
   </Button>

</LinearLayout>
```



MainActivity.Java

```
public class MainActivity extends Activity implements View.OnClickListener {

    private ProgressBar progress;
    private TextView text;
    private Button btn;
    private Thread bgthread;

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

        progress = (ProgressBar) findViewById(R.id.progressBar1);
        text = (TextView) findViewById(R.id.textView1);
        btn = (Button) findViewById(R.id.button1);

        btn.setOnClickListener(this);

    }
```

MainActivity.Java

```
@Override
public void onClick(View v) {
    // Buat Thread baru setiap kali tombol start progress di klik
    // Setiap kali thread akan dijalankan, harus dibuat baru,
    // Thread yang sudah finish/terminated tidak bisa dijalankan kembali
    if (bgthread == null || bgthread.getState() == Thread.State.TERMINATED) {

        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                try {

                    for (int i = 0; i <= 10; i++) {
                        final int value = i;
                        // Simulating something timeconsuming
                        Thread.sleep(1000); // in milisecond

                        progress.post(new Runnable() {
                            @Override
                            public void run() {
                                text.setText("Updating "+value+"/10");
                                progress.setProgress(value);
                            }
                        });
                    }

                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        };
        bgthread = new Thread(runnable);
        bgthread.start();

    }
}
```

AsyncTask

Simplifying Android Thread process management

AsyncTask

Complex operations that require frequent UI updates need complicated threads code. To remedy this problem, Android 1.5 and above offers a new utility class, called **AsyncTask**.

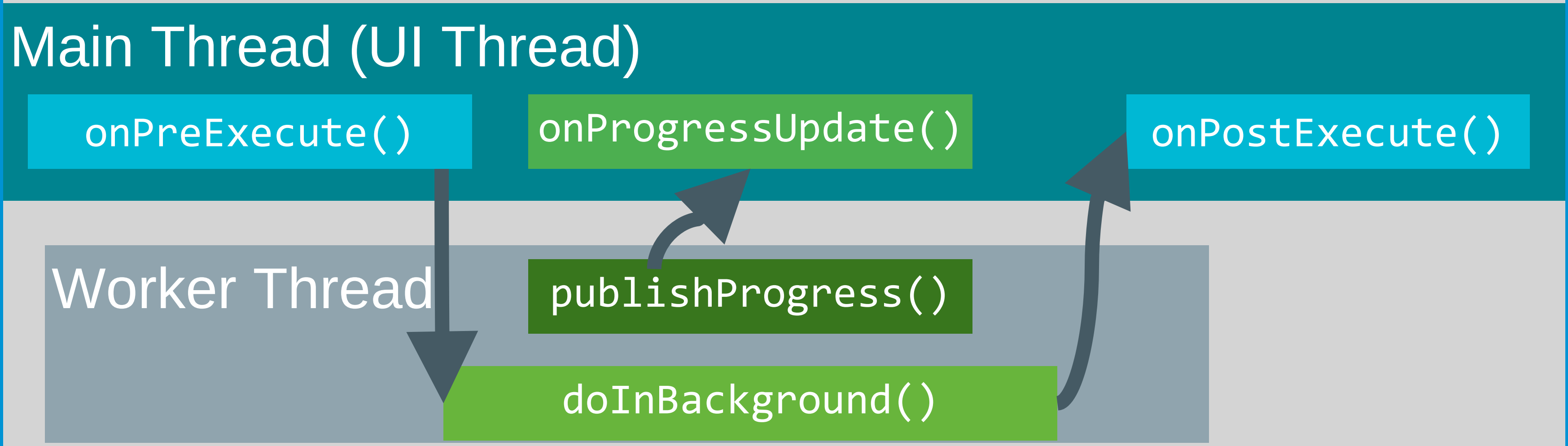
The goal of **AsyncTask** is to take care of thread management.

AsyncTask instance has to be created on the UI thread and can be executed only once.

Use AsyncTask to implement basic background tasks.

AsyncTask Basic Process

- ❑ Create a class that extends **AsyncTask** Class.
- ❑ To start the new thread, call the AsyncTask's **execute()** method
- ❑ When execute is called, Android does the following:
  1. runs **onPreExecute()** in the main (UI) thread.
  2. runs **doInBackground()** in a background thread.
  3. runs **onPostExecute()** in the main (UI) thread.



AsyncTask prototype sample

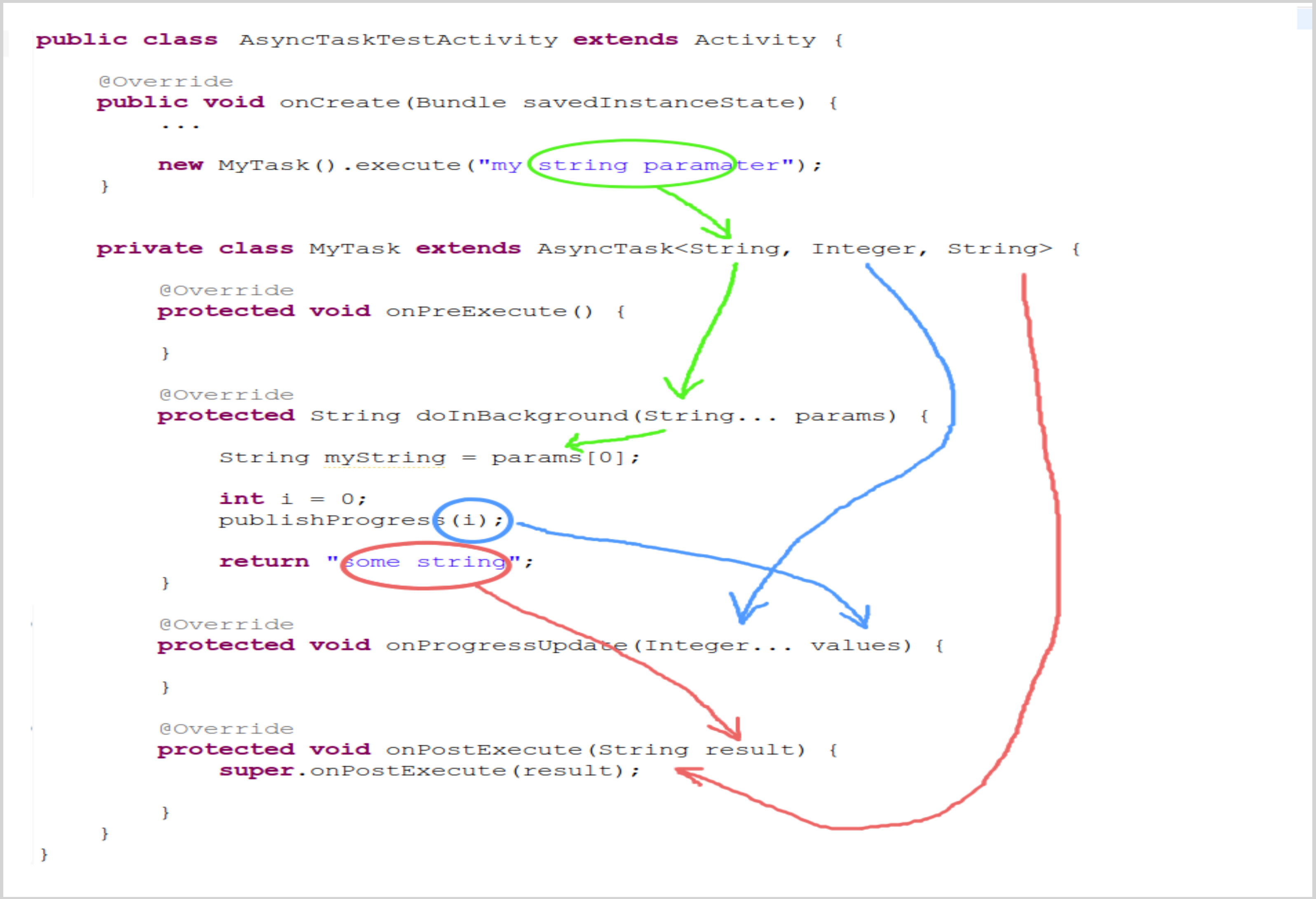
```
public class SomeTask
     extends AsyncTask<String, Integer, Double>
```

Data type passed when Task execute() method called and passed to doInBackground() method

Data type passed to onProgressUpdate() method When publishProgress() method called from doInBackground() method

Data type passed to Task's onPostExecute() method returned from doInBackground() method

AsyncTask Instantiation



```
public class AsyncTaskTestActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...

        new MyTask().execute("my string paramater");
    }


    private class MyTask extends AsyncTask<String, Integer, String> {

        @Override
        protected void onPreExecute() {

        }

        @Override
        protected String doInBackground(String... params) {

            String myString = params[0];

            int i = 0;
            publishProgress(i);

            return "some string";
        }

        @Override
        protected void onProgressUpdate(Integer... values) {

        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

        }
    }
}
```

Limitations of AsyncTask

- When device configuration changes, Activity is destroyed.
- AsyncTask cannot connect to Activity anymore.
- New AsyncTask created for every config change.
- Old AsyncTasks stay around.
- App may run out of memory or crash.

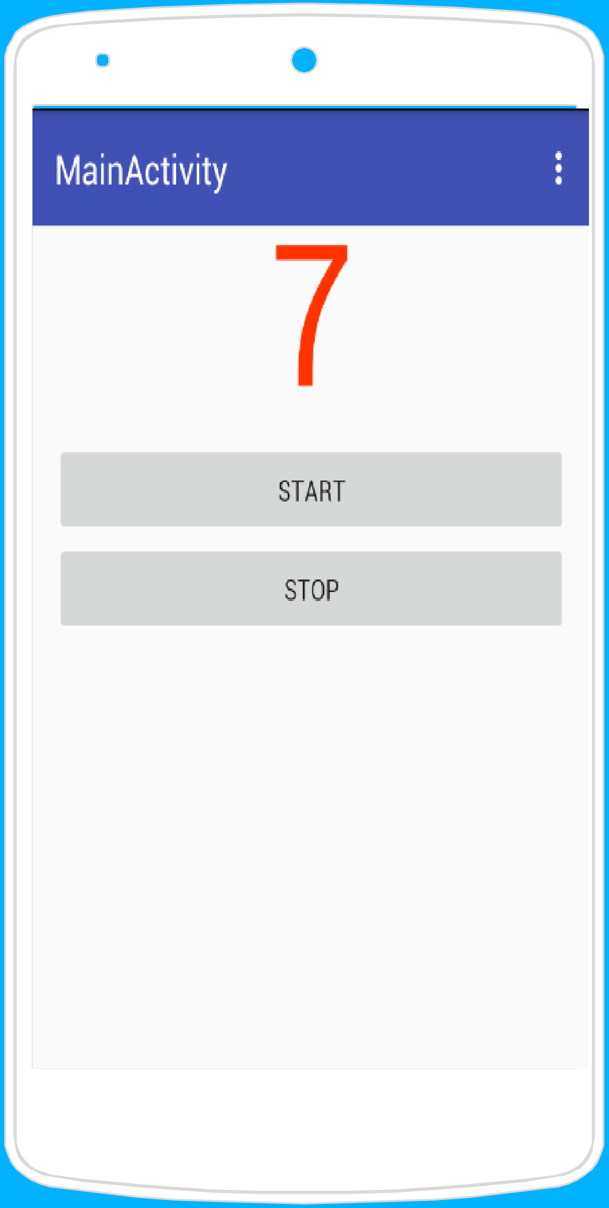
When to use AsyncTask

- Short or interruptible tasks
- Tasks that do not need to report back to UI or user
- Lower priority tasks that can be left unfinished

For Advance usage : Use ***AsyncTaskLoader***

Task: Random Number Generator

- ❑ Buat Aplikasi yang memungkinkan ketika tombol start di klik, TextView akan menggenerate angka 0 – 9 secara random.
- ❑ Delay antar pergantian angka 0.5 detik.
- ❑ Ketika tombol stop di tekan, aplikasi akan berhenti menggenerate angka.
- ❑ Kerjakan menggunakan Thread atau AsyncTask.



Thanks

We are moving..

http://j.gs/18164083/papb-tif