Université Mohammed V
FACULTE DES SCIENCES
RABAT / FSR
Département informatique

Master IAO
Master II–Semestre 3
Cours

Mobile & Cloud Computing

Pr. REDA Oussama Mohammed

2019-2020

Concuruncy in Android

Android Application Model, Processes, UI Thread

and

Handlers

Application Components

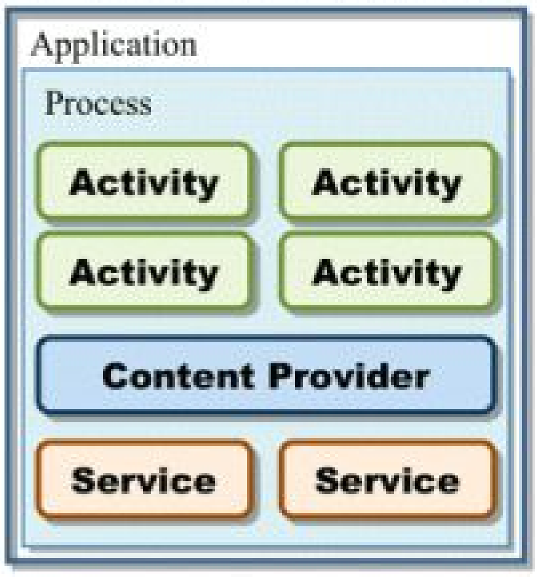
- Android applications do not have a single entry point (e.g. no main () function).
- They have essential components that the system can instantiate a nd run as needed.
- Four basic components

Components	Description
Activity	UI component typically corresponding to one screen
Service	Background process without UI
Broadcast Receiver	Component that responds to broadcast Intents
Content Provider	Component that enables applications to share data

Activities	Presents a visual user interface for one focused endeavor the user can undertake. List of menu items a user can choose from or display photographs along with their captions
Services	Doesn't have a visual user interface, instead runs in the background Play background audio as the user attends to other matters
Broadcast Receivers	Receives and reacts to broadcast announcements An application can announce to "whoever is listening" that a picture was taken.
Content Providers	Makes a specific set of the application's data available to other applications. An application uses a contact list component
Intents	A simple message passing framework. Using intents you can broadcast messages system-wide or to a target Activity or Service.

Android Component Model

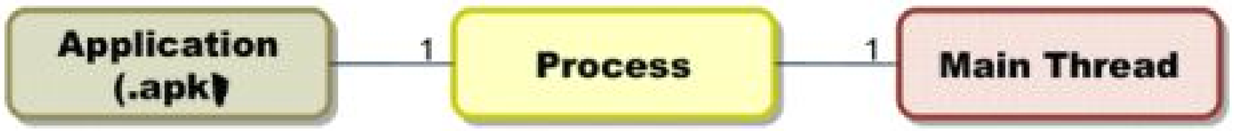
- An Android application is packaged in a .apk file.
  - ✓ A .apk file is a collection of components.



  - ✓ Components share a Linux process: by default, one process per .apk file.
  - ✓ .apk files are isolated and communicate with each other via Intents or AIDL.
  - ✓ Every component has a managed lifecycle.

Processes and Threads

- Processes
  - ✓ When the first of an application's components needs to be run, Android starts a Linux process for it with a single thread of execution (Main Thread). Additional threads can be spawned for any process.

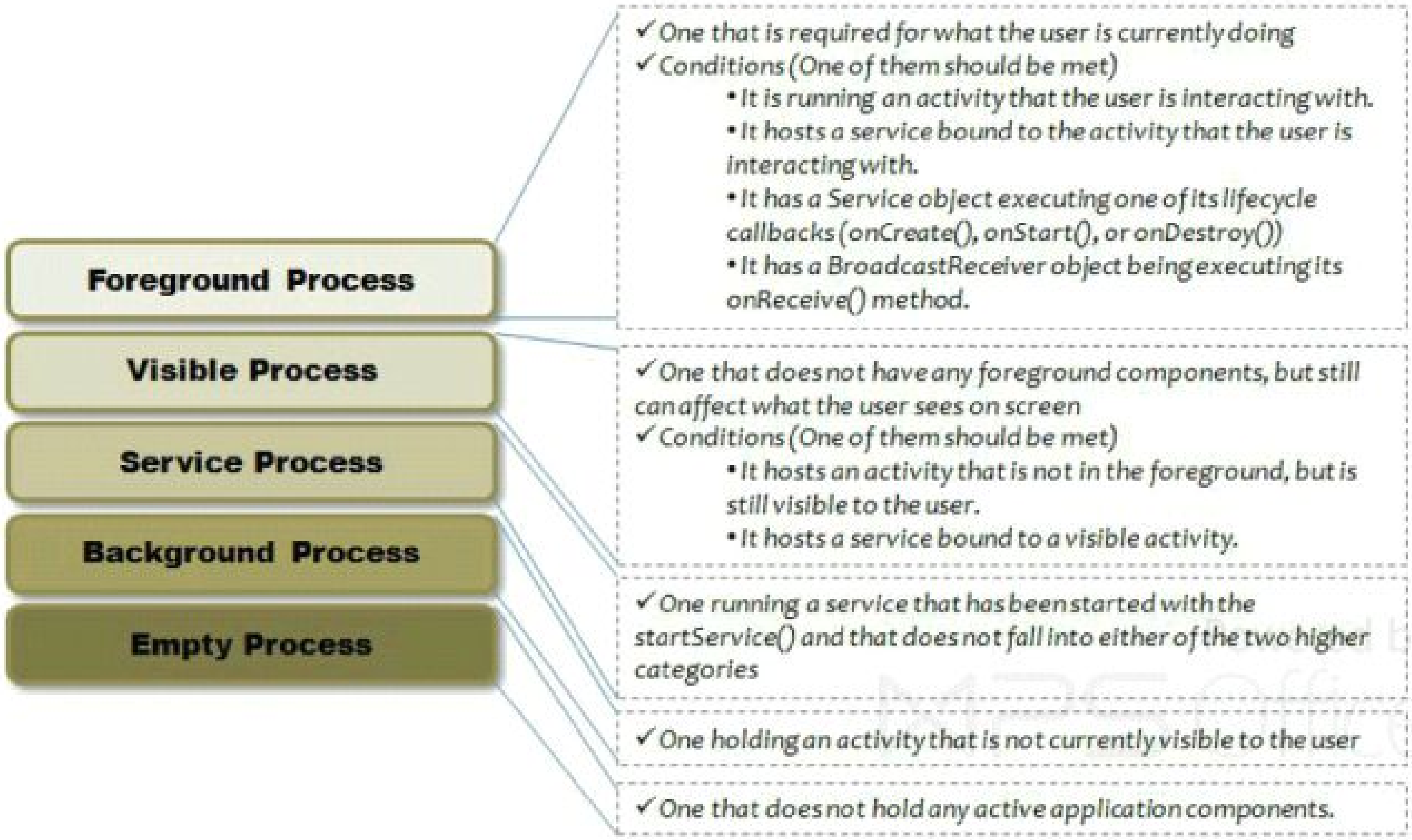


    - Each component can run in its own process.
    - You can arrange for components to run in other processes.
    - Some components share a process while others do not.
    - Components of different applications also can run in the same process.

  - ✓ Android may decide to kill a process to reclaim resources.

Component Lifecycles (Cont)

- Processes and Lifecycles (Cont)
  - ✓ Five levels in the Importance Hierarchy



✓ One that is required for what the user is currently doing

✓ Conditions (One of them should be met)

- It is running an activity that the user is interacting with.
- It hosts a service bound to the activity that the user is interacting with.
- It has a Service object executing one of its lifecycle callbacks (onCreate(), onStart(), or onDestroy())
- It has a BroadcastReceiver object being executing its onReceive() method.

✓ One that does not have any foreground components, but still can affect what the user sees on screen

✓ Conditions (One of them should be met)

- It hosts an activity that is not in the foreground, but is still visible to the user.
- It hosts a service bound to a visible activity.

✓ One running a service that has been started with the startService() and that does not fall into either of the two higher categories

✓ One holding an activity that is not currently visible to the user

✓ One that does not hold any active application components.

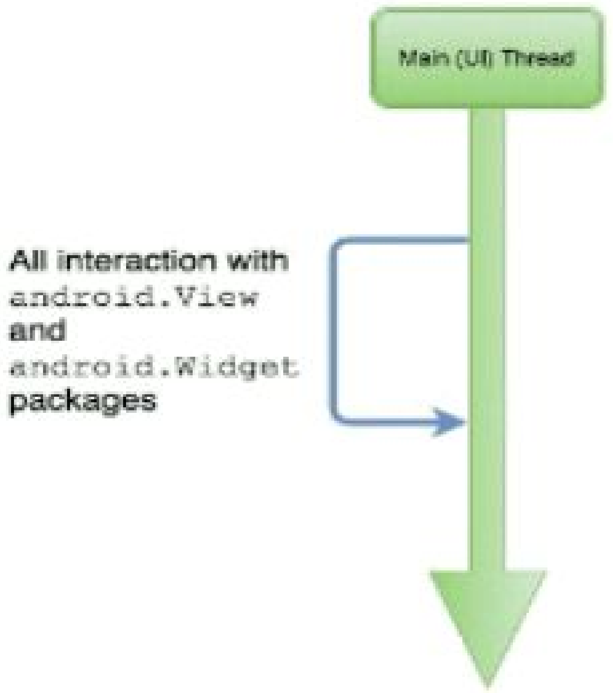
Processes

- Android may decide to shut down a process at some point, when memory is low and required by other processes that are more immediately serving the user.
- Application components running in the process are consequently destroyed.
- A process is restarted for those components when there's again work for them to do.
- When deciding which processes to terminate, Android weighs their relative importance to the user.
- For example, it more readily shuts down a process with activities that are no longer visible on screen than a process with visible activities.
- The decision whether to terminate a process, therefore, depends on the state of the components running in that process.

Processes and Threads

- Threads
  - ✓ Main Thread (UI Thread)
    - ➢ It is in charge of dispatching events to the appropriate user interface widgets, including drawing events.
    - ➢ It is also the thread in which your application interacts with components from the Android UI toolkit (components from the android.widget and android.view packages).
      - ■ As such, the main thread is also sometimes called the UI thread.

Android Threading


Main (UI) Thread
All interaction with `android.View` and `android.Widget` packages

Android creates a thread called "main" (often referred to as the UI thread) for each application when it starts.

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

Processes and Threads (Cont)

- Threads
  - ✓ Main Thread
    - ➢ All components are instantiated in the main thread (UI Thread) of the specified process.
    - ➢ System calls to the components are dispatched from the main thread (UI widgets and views).
      - ■ Methods that respond to those calls always run in the main thread of the process (such as onKeyDown() to report user actions or a lifecycle callback method).

Processes and Threads (Cont)

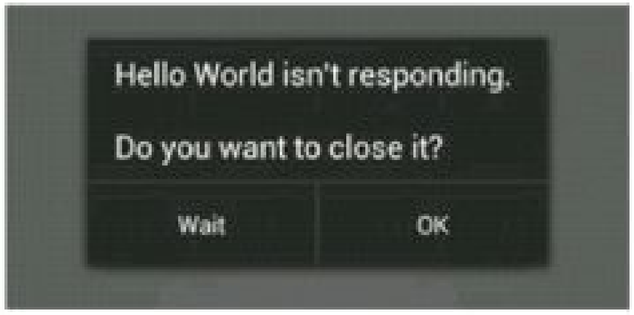
- Threads
  - ✓ Main/UI Thread (Exemple)
    - ➢ The user touches a button on the screen
      - ■ The application's UI thread dispatches the touch event to the widget.
      - ■ The widget sets its pressed state and posts an invalidate request to the event queue.
      - ■ The UI thread dequeues the request and notifies the widget that it should redraw itself.

Processes and Threads (Cont)

- Threads
  - ✓ Main Thread (UI Thread)
    - ➢ When an app performs intensive work in response to user interaction, this single thread model can yield poor performance unless the application is implemented properly.
    - ➢ Specifically, if everything is happening in the UI thread, performing long operations such as network access or database queries will block the whole UI.
      - ■ When the thread is blocked, no events can be dispatched, including drawing events.
      - ■ From the user's perspective, the application appears to hang.

Processes and Threads (Cont)

- Threads
  - ✓ If the UI thread is blocked for more than a few seconds (about 5 seconds currently) the user is presented with the infamous "application not responding" (ANR) dialog.



Hello World isn't responding.

Do you want to close it?

- ➢ The user might then decide to quit your application and uninstall it if they are unhappy. So,

1)

No component should perform long or blocking operations (e.g. I/O operations, network access, computation loops)

Processes and Threads (Cont)

- Threads (Cont)
  - ✓ Solution
- Use a background thread to do the task (e.g. I/O operations, net work access, computation loops)
  - ✓ Consequence

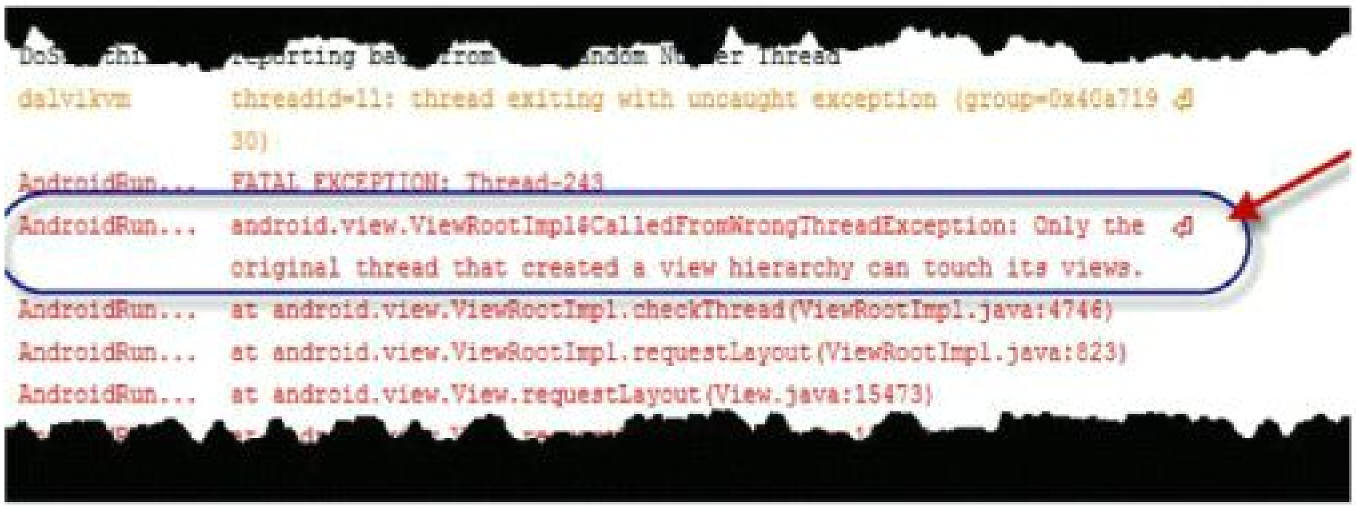
    Background thread and UI thread are running concurrently and may have race conditions if they modify UI simultaneously (e.g., UI switches to a different orientation)

Problem:

The Andoid UI toolkit is **not thread-safe**

Processes and Threads (Cont)

- Threads
  - ✓ The Andoid UI toolkit is **not thread-safe**. So, you must not manipulate your UI from a worker thread—you must do all manipulation to your user interface from the UI thread.

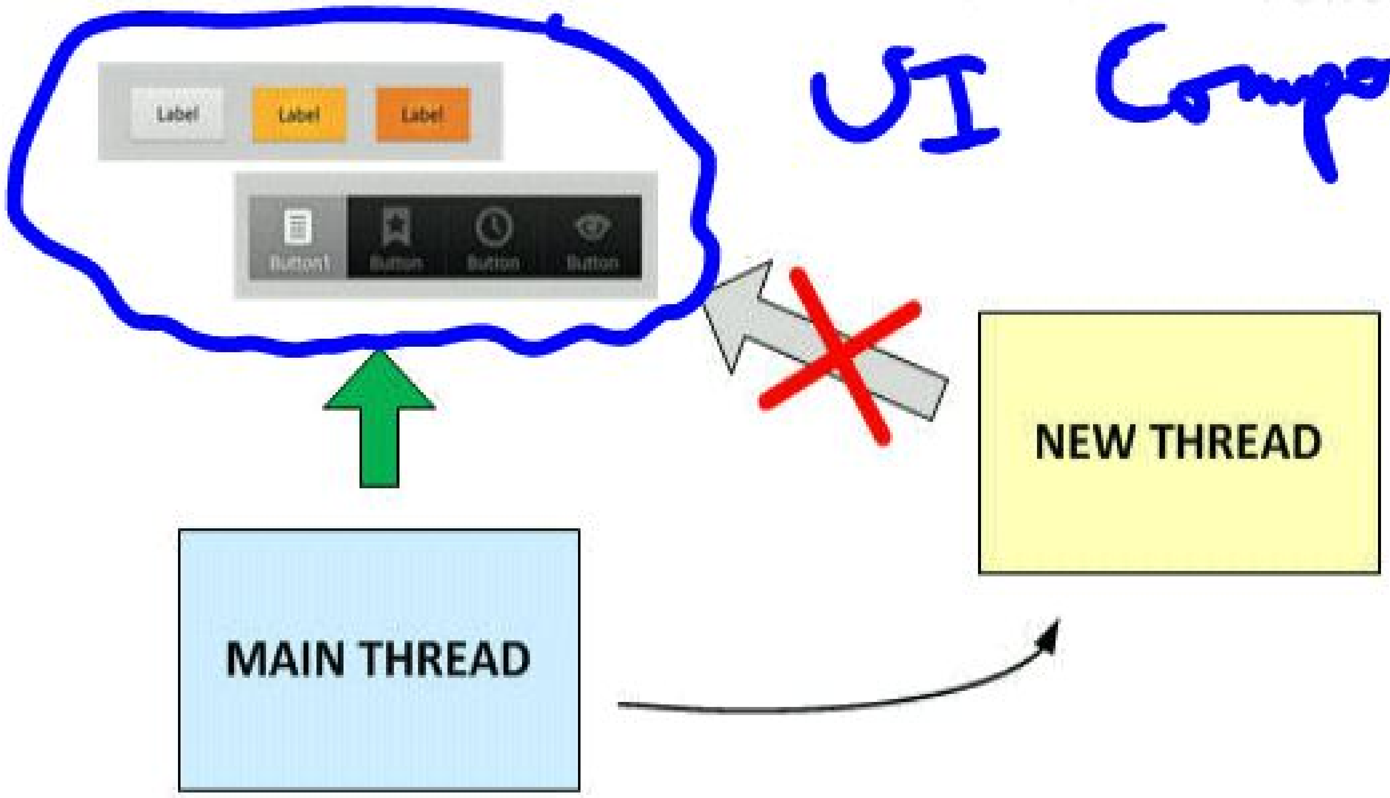


```
DoS  chi      eporting ba  from    andom Nu  er Thread
dalvikvm       threadid=11: thread exiting with uncaught exception (group=0x40a719
               30)
AndroidRun...  FATAL EXCEPTION: Thread-243
AndroidRun...  android.view.ViewRootImpl$CalledFromWrongThreadException: Only the
               original thread that created a view hierarchy can touch its views.
AndroidRun...  at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:4746)
AndroidRun...  at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:823)
AndroidRun...  at android.view.View.requestLayout(View.java:15473)
```

2)

Do not access the Android UI toolkit from outside the UI thread

UI Components



Processes and Threads (Cont)

No component should perform long or blocking operations (such as networking operations or computation loops) when called by the system, since this will block any other components also in the process.

- Since the user interface must always be quick to respond to user actions, the thread that hosts an activity should not also host time-consuming operations like network downloads.
- Anything that may not be completed quickly should be assigned to a different thread (Spawn separate threads for long operations (background work)).

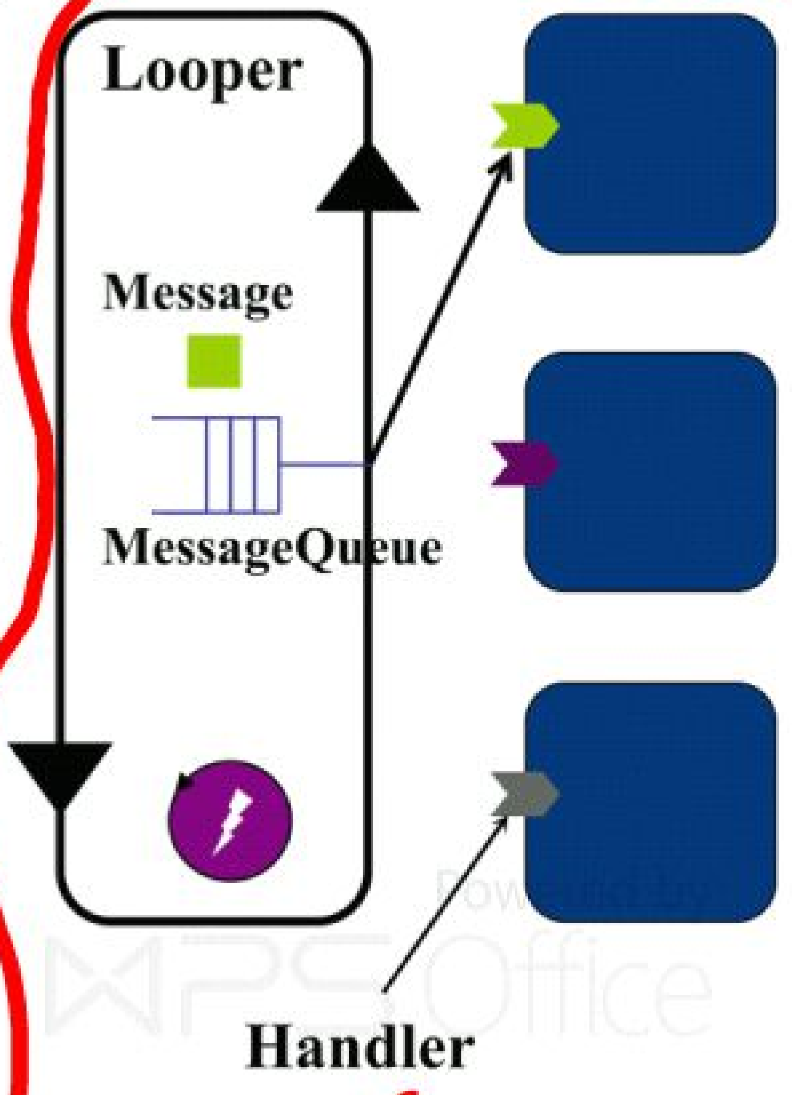
That's Multithreading in Android

Processes and Threads (Cont)

- Threads (Cont)
  - ✓ Anything that may not be completed quickly should be assigned to a different thread.
    - ➢ Threads are created in code using standard Java Thread objects.
  - ✓ Some convenience classes Android provides for managing threads:
    - ➢ Looper for running a message loop within a thread
    - ➢ Handler for processing messages
    - ➢ HandlerThread for providing a handy way for starting a new thread that has a looper

Android event classes: some details

- Android defines a set of classes for event-driven programming in conjunction with threads.
- A thread may have at most one Looper bound to a MessageQueue.
- Each Looper has exactly one thread and exactly one MessageQueue.
- The Looper has an interface to register Handlers.
- There may be any number of Handlers registered per Looper.
- These classes are used for the UI thread, but have other uses as well.



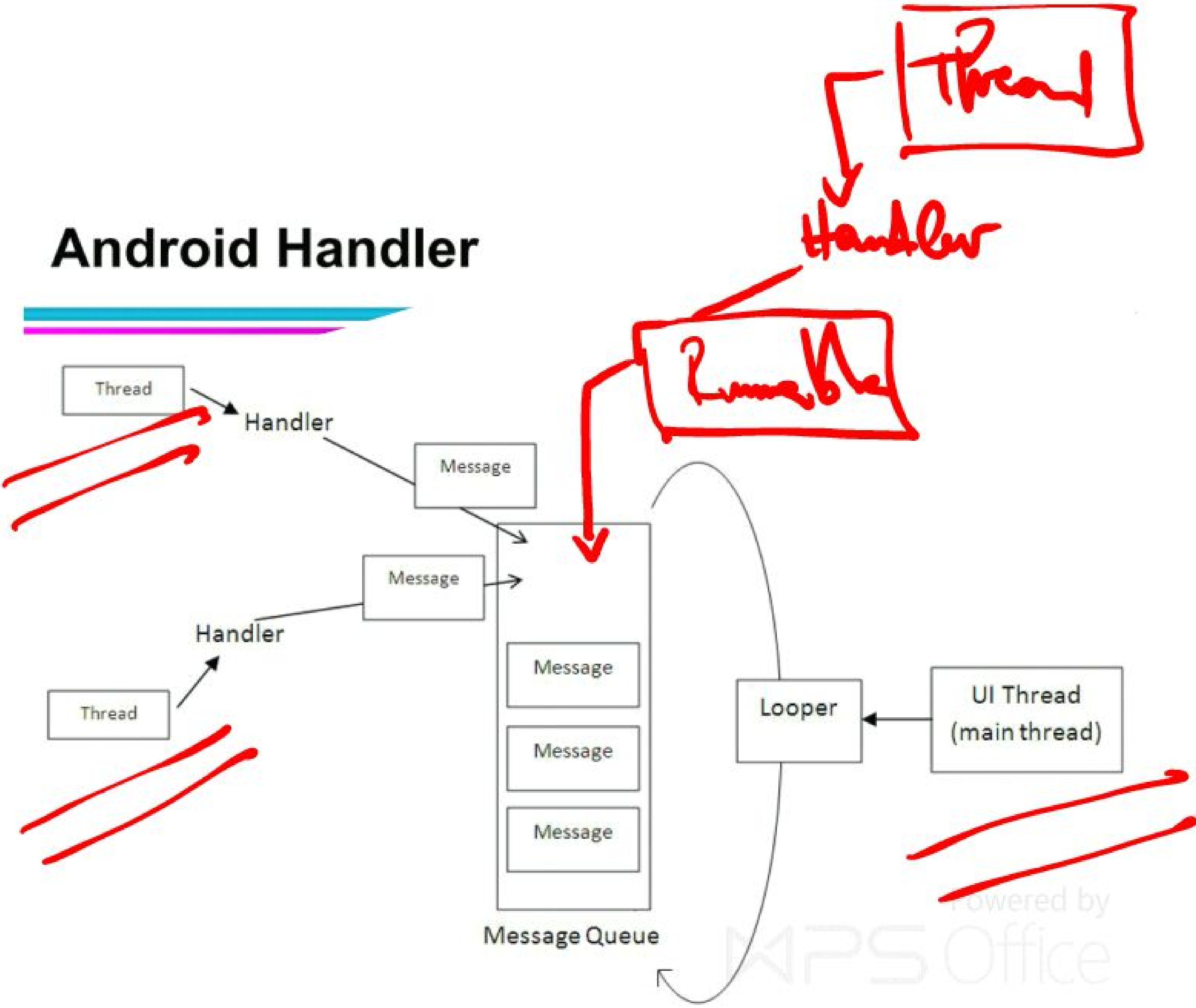
A Thread with its associated Looper and Handlers

Android Handler

- Android's mechanism to send and process Message and Runnable objects associated with a thread's MessageQueue.

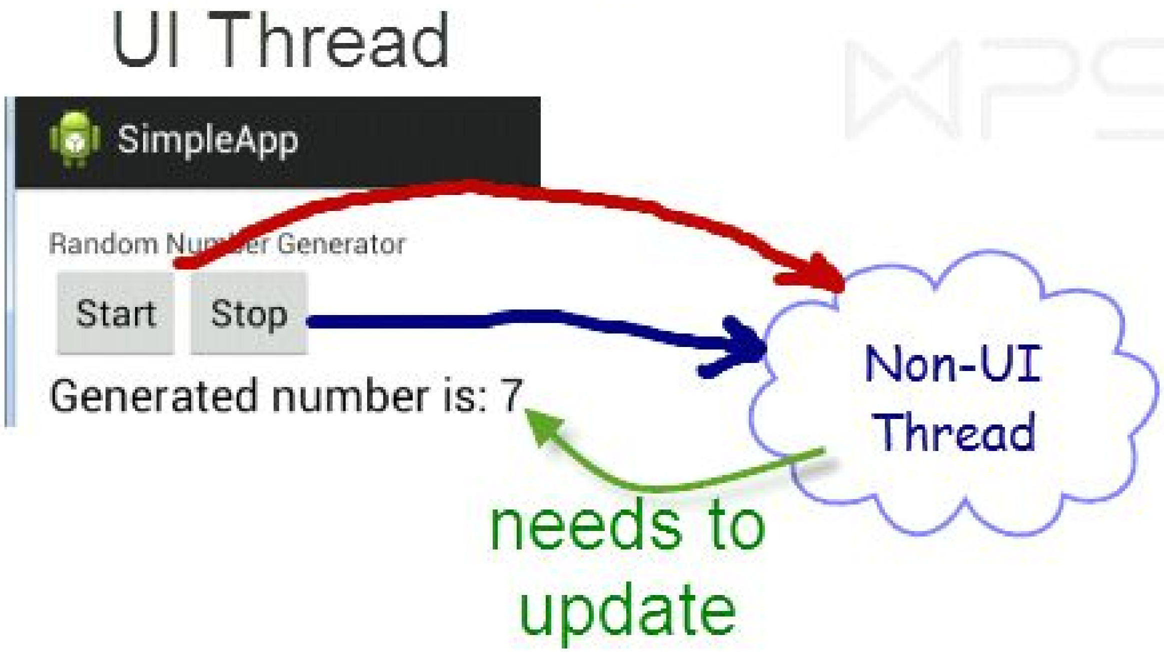
- Each Handler instance is associated with a single thread and that thread's message queue
    - A handler is bound to the thread / message queue of the thread that creates it
    - from that point on, it will deliver messages and runnables to that message queue
    - That thread processes msgs

Android Handler



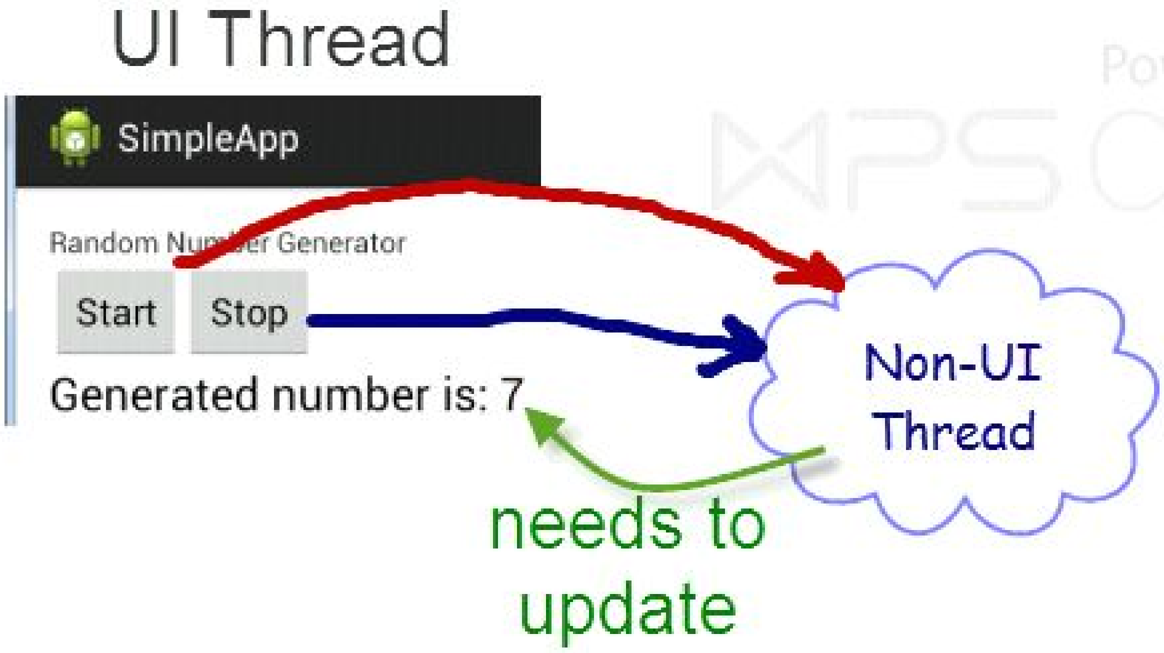
Android Handler

```
public class MyActivity extends Activity {
        [ . . . ]
        // Need handler for callbacks to the UI thread
        final Handler mHandler = new Handler();
        // Create runnable task to give to UI thread
        final Runnable mUpdateResultsTask = new Runnable() {
           public void run() {
              updateResultsInUi();
           }
        };
        @Override
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           [ . . . ]
        }
```



Android Handler

```
protected void startLongRunningOperation() {
      // Fire off a thread to do some work that we shouldn't do directly in the UI thread
      Thread t = new Thread() {
         public void run() {
            mResults = doSomethingExpensive();
            mHandler.post(mUpdateResultsTask);   // post() method
         }
      };
      t.start();
   }
   private void updateResultsInUi() {
      // Back in the UI thread -- update our UI elements based on the data in mResults
      [ . . . ]
   }
}
```



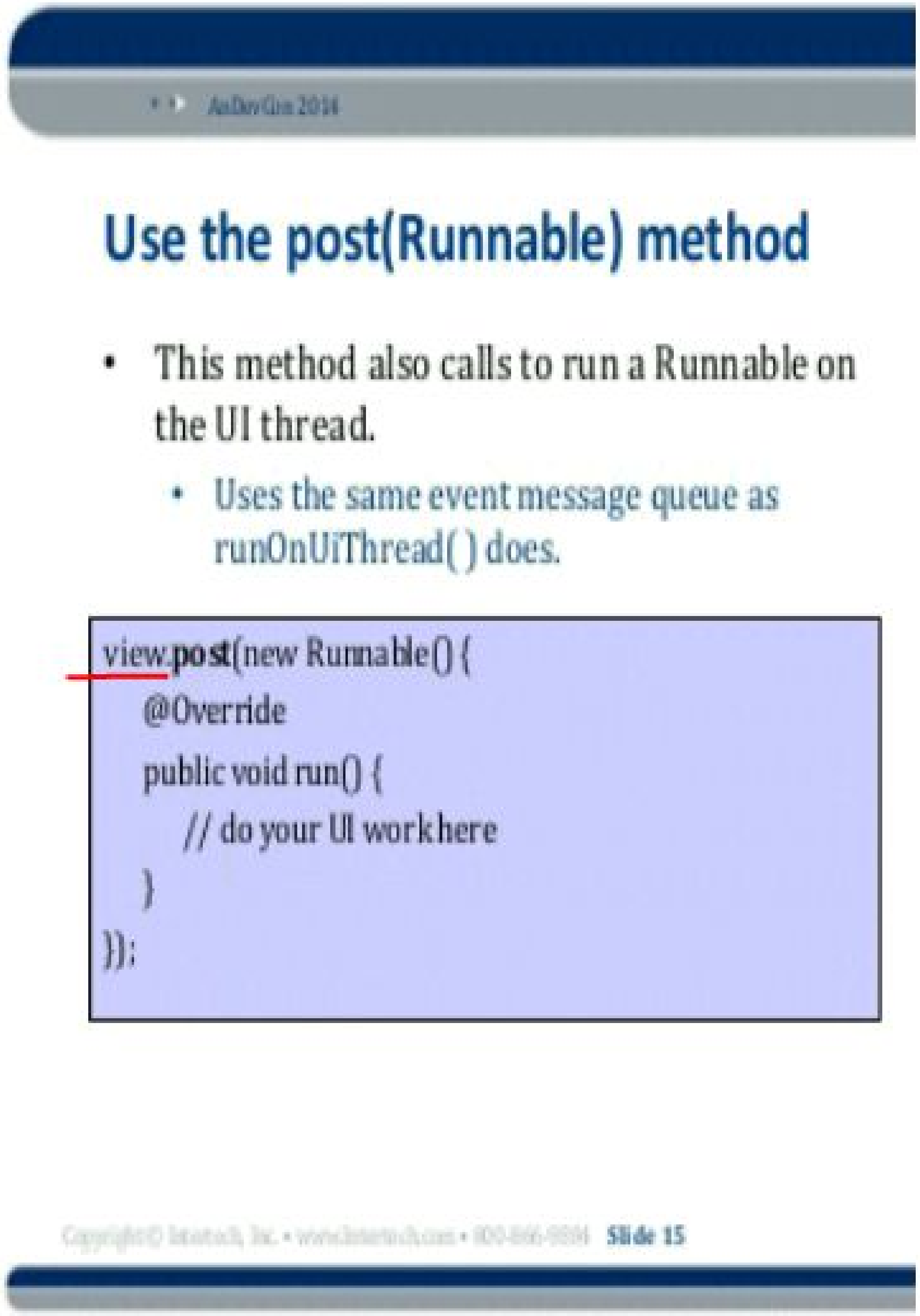
Common Pattern

```
private Handler mHandler = new Handler(); // UI handler
private Runnable longTask = new Runnable() {

   // processing thread
   public void run() {
       while (notFinished) {
          // doSomething

          mHandler.post(taskToUpdateProgress);
       }

       // mHandler.post(taskToUpdateFinalResult);

  };

Thread thread = new Thread(longTask);
thread.start();
```

update progressively

update at the end of processing

view.post()
&
activity.runOnUiThread()



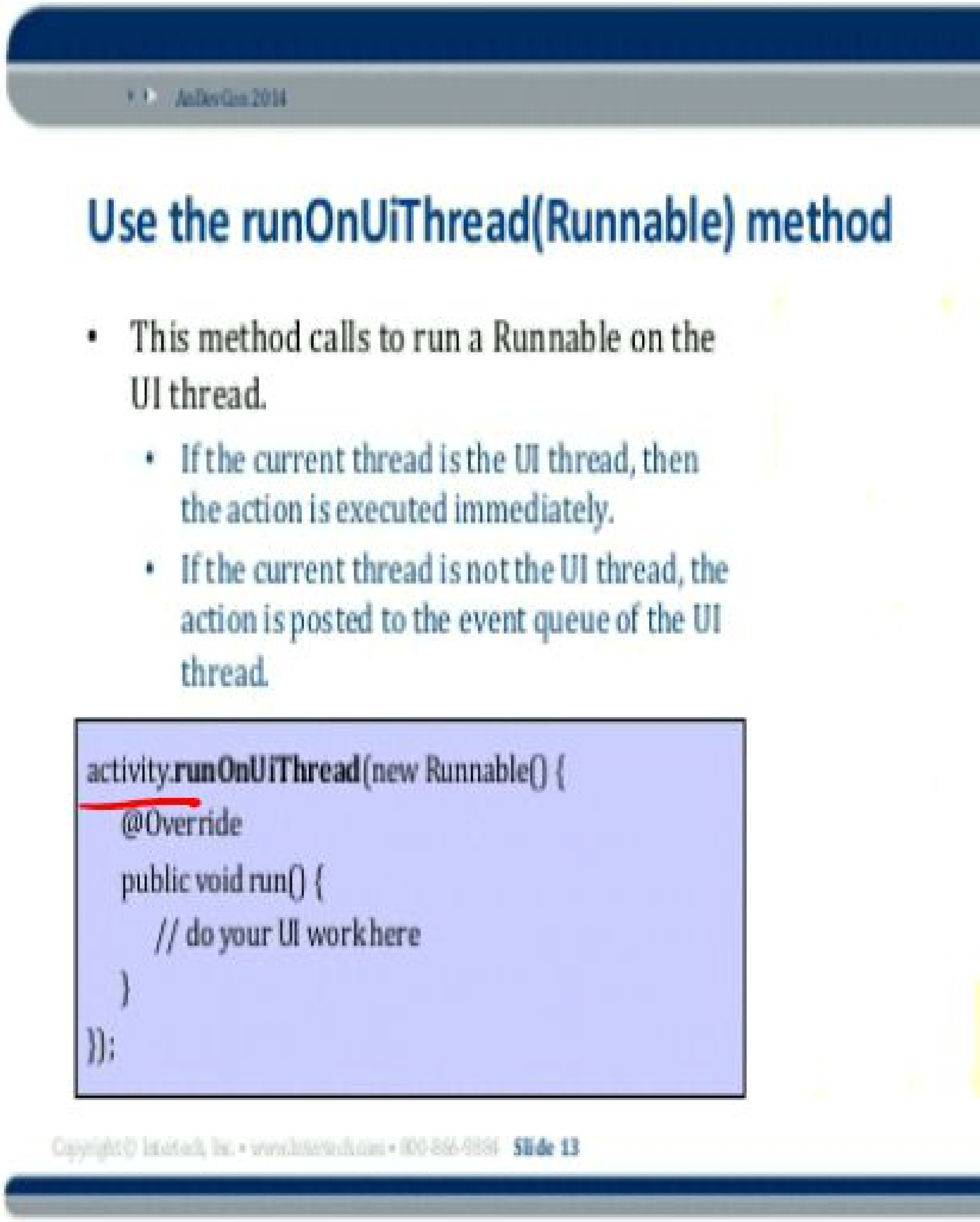
AnDevCon 2014

Use the post(Runnable) method

- This method also calls to run a Runnable on the UI thread.
  - Uses the same event message queue as runOnUiThread( ) does.

```
view.post(new Runnable() {
   @Override
   public void run() {
      // do your UI work here
   }
});
```

Copyright © Intertech, Inc. • www.Intertech.com • 800-866-9884 **Slide 15**



AnDevCon 2014

Use the runOnUiThread(Runnable) method

- This method calls to run a Runnable on the UI thread.
  - If the current thread is the UI thread, then the action is executed immediately.
  - If the current thread is not the UI thread, the action is posted to the event queue of the UI thread.

```
activity.runOnUiThread(new Runnable() {
   @Override
   public void run() {
      // do your UI work here
   }
});
```

Copyright © Intertech, Inc. • www.Intertech.com • 800-866-9884 **Slide 13**

Concuruncy in Android

Android Application Model, Processes, UI Thread

and

Handlers

Android Application Model, Processes, UI Thread and Handlers

Supports de présentation

http://moss.csc.ncsu.edu/~mueller/g1/
http://db.cs.duke.edu/courses/cps110/fall12/slides/
http://zoo.cs.yale.edu/classes/cs434/cs434-2012-fall/lectures/

Université Mohammed V
FACULTE DES SCIENCES
RABAT / FSR
Département informatique

Master IAO
Master II–Semestre 3
Cours

Mobile & Cloud Computing

Pr. REDA Oussama Mohammed

2015/2016

The end...

بسمه تعالی...

Université Mohammed V
FACULTE DES SCIENCES
RABAT / FSR
Département informatique

Master IAO
Master II–Semestre 3
Cours

Mobile & Cloud Computing

Pr. REDA Oussama Mohammed

2019-2020