Background Tasks and Services

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

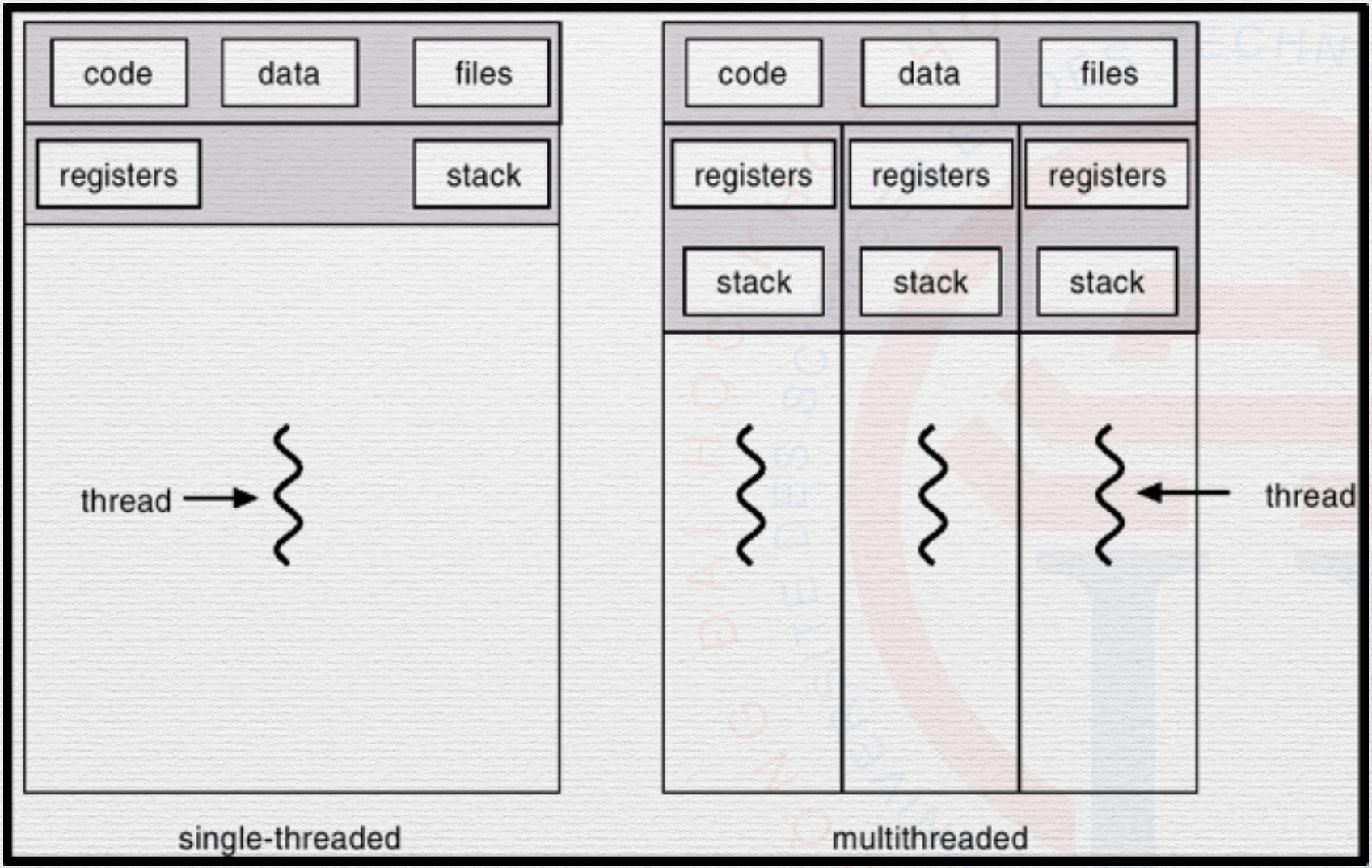
Contents

Threading

Threading

- What's a thread? Why thread?
- Threads in HTML5?
- Context switches?
- Threads vs applications?

Threading


code
data
files
registers
stack
thread
single-threaded
code
data
files
registers
registers
registers
stack
stack
stack
thread
multithreaded

Why Threading?

- Better CPU utilization
- Separation of tasks
- Responsiveness

Why NOT Threading?

- Complication
  - Architecture
  - Load balancing
- Synchronization
- Thread pool...

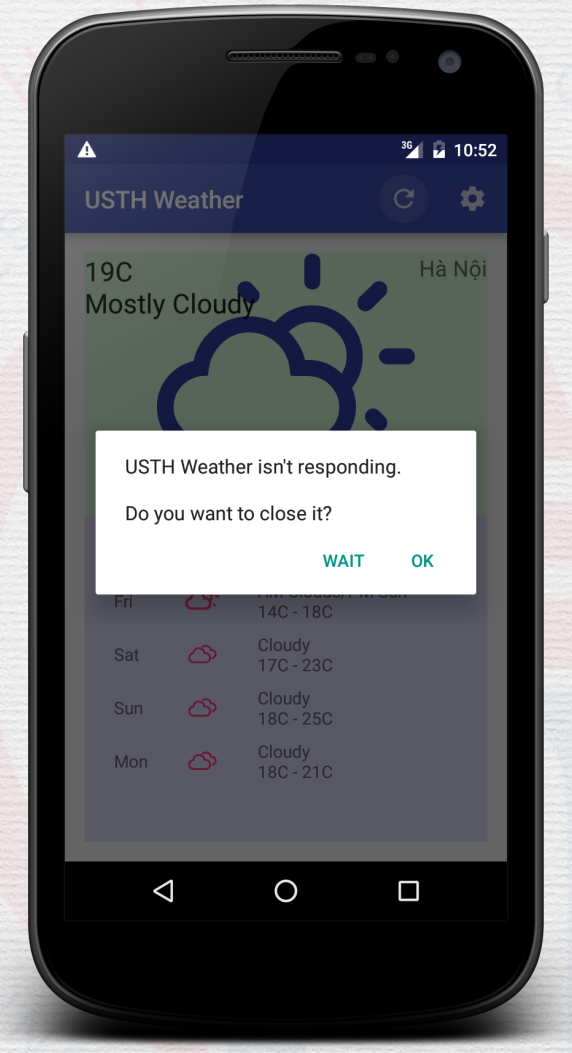
Android Thread Model

Android Thread Model

- Main thread
  - Drawing widgets
  - Dispatching user inputs
  - Widget toolkit is not thread-safe
- Worker threads

Android Thread Model

- Don't do **sllloooooowwwww** things on main thread
  - Calculation
    - Bitcoin mining
    - Image processing
    - ...
  - Network access

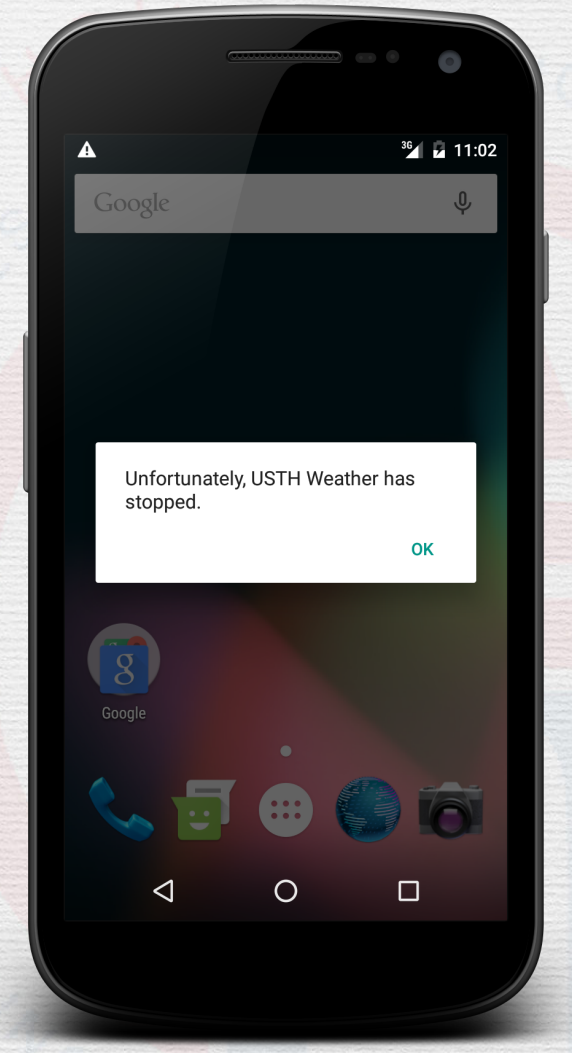


Android Thread Model

- Don't manipulate Views on worker thread
- Crash

```
android.view.ViewRootImpl$CalledFromWrongThreadException:
Only the original thread that created a view hierarchy can
touch its views.

at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:6556)
at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:907)
...
```



Android Thread Model

- Create a new worker thread

```
public class WeatherActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                // do something heavy in the new thread.
                // don't access UI Views here.
            }
        });
        t.start();
    }
}
```

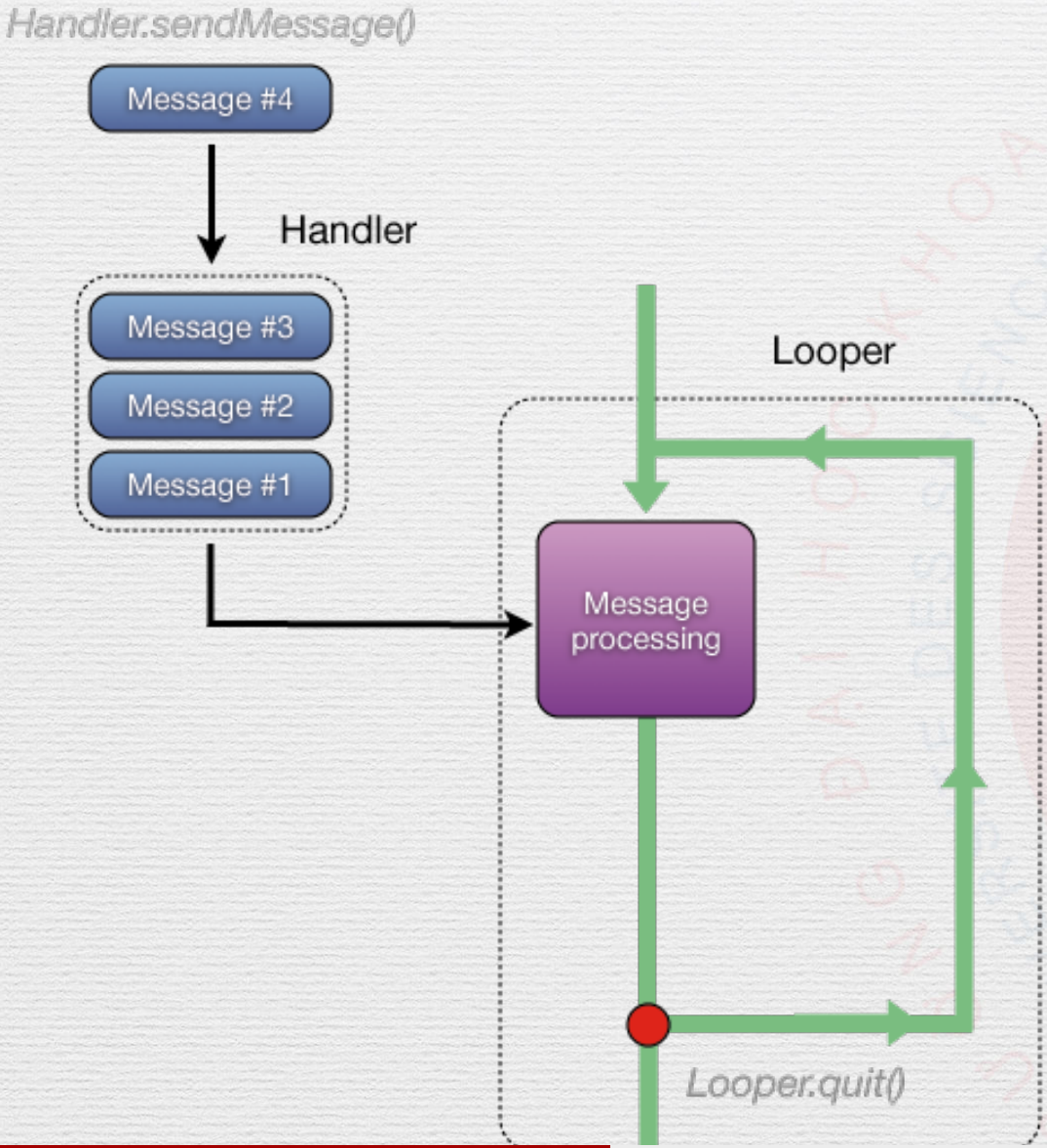
Android Thread Model

- If we cannot access UI Views in worker threads, how can we show on UI that something has finished?
  - Notification between worker threads and main thread

Android Thread Model

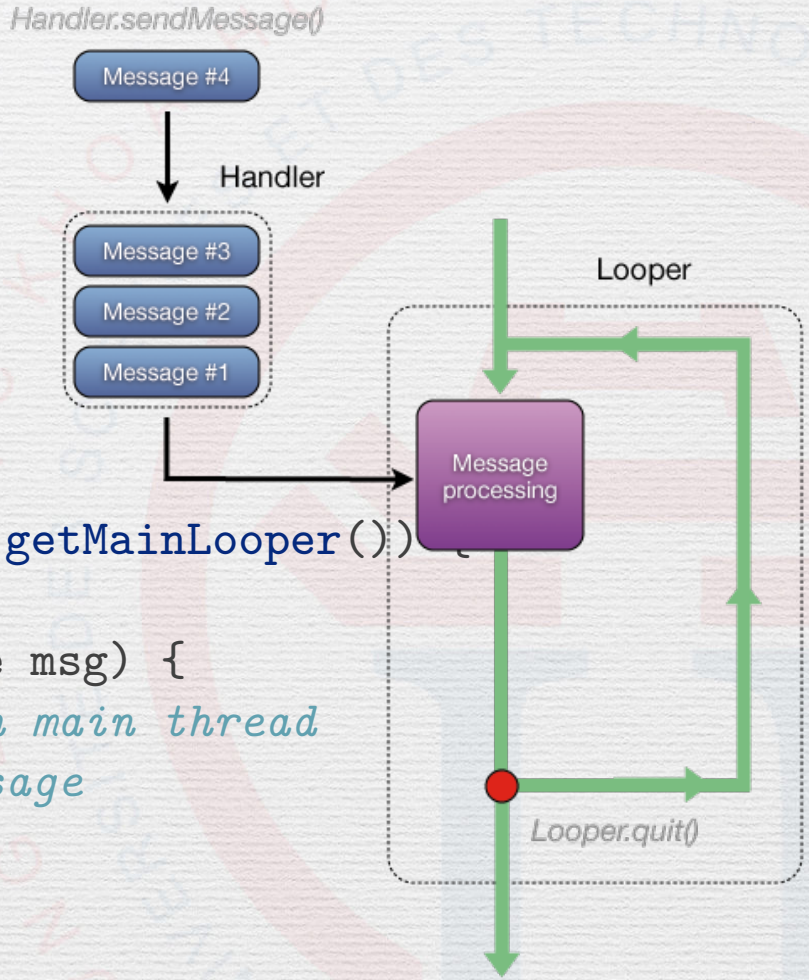
- «Handler»
  - A way to communicate with main thread
  - `Handler.handleMessage()` is executed on main thread
  - Worker thread: call `handler.sendMessage()`

Handler


Handler.sendMessage()
Message #4
Handler
Message #3
Message #2
Message #1
Looper
Message processing
Looper.quit()

Handler

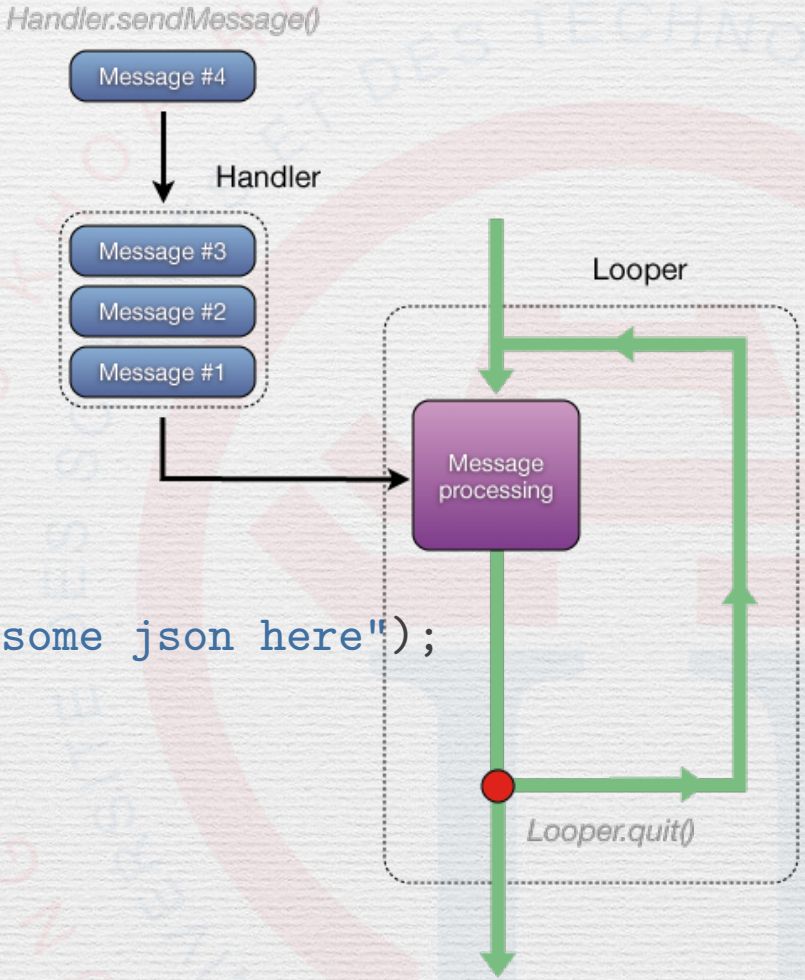
- Main thread:




```
Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(Message msg) {
        // this method is executed on main thread
        // do something with the message
    }
};
```

Handler

- Network (background) thread:




```
Bundle bundle = new Bundle();
bundle.putString("server_response", "some json here");

Message msg = new Message();
msg.setData(bundle);

handler.sendMessage(msg);
```

Handler

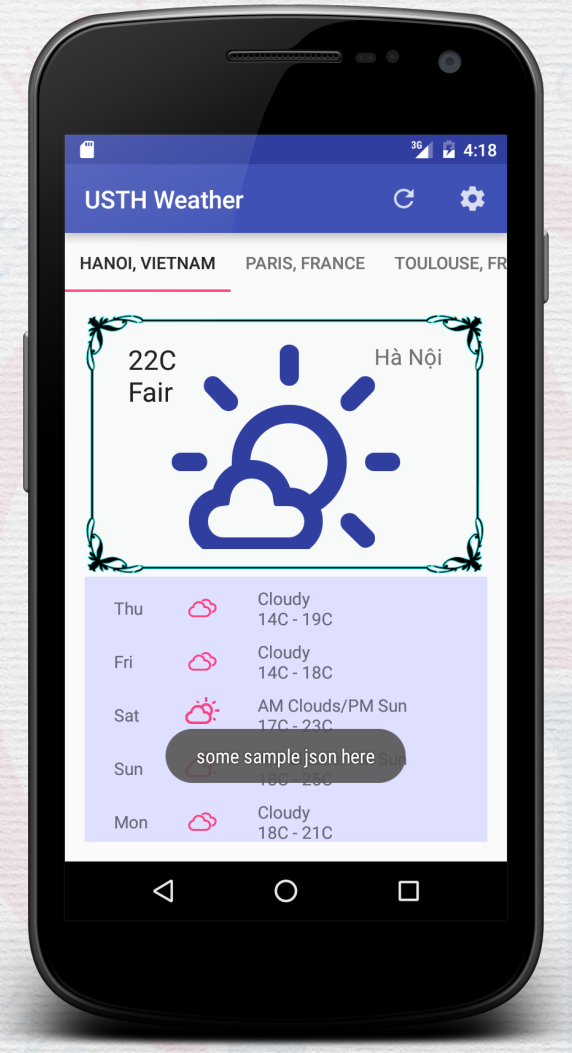
```
final Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(Message msg) {
        // This method is executed in main thread
        String content = msg.getData().getString("server_response");
        Toast.makeText(getActivity(), content, Toast.LENGTH_SHORT).show();
    }
};
Thread t = new Thread(new Runnable() {
        @Override
        public void run() {
            // this method is run in a worker thread
            try {
                // wait for 5 seconds to simulate a long network access
                Thread.sleep(5000);
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }

            // Assume that we got our data from server
            Bundle bundle = new Bundle();
            Bundle.putString("server_response", "some sample json here");

            // notify main thread
            Message msg = new Message();
            msg.setData(bundle);
            handler.sendMessage(msg);
        }
    });
t.start();
```

Practical Work 13

- Add a “Refresh” button on your activity (or fragment)
  - Best place would be an action on App Bar
- Use thread and handler, simulate a network request and show on a toast



Background Tasks

Background Tasks

- «AsyncTask»
  - An encapsulation of Handler and Thread
  - Also allow the worker thread to report its work progress to the UI

AsyncTask: param

- 3 Generic **Types**:
  - **AsyncTask<Param, Progress, Result>**
  - Params: param type to pass to the worker thread
  - Progress: type to report progress back
  - Result: result type to be delivered

```
AsyncTask<String, Integer, Bitmap>
```

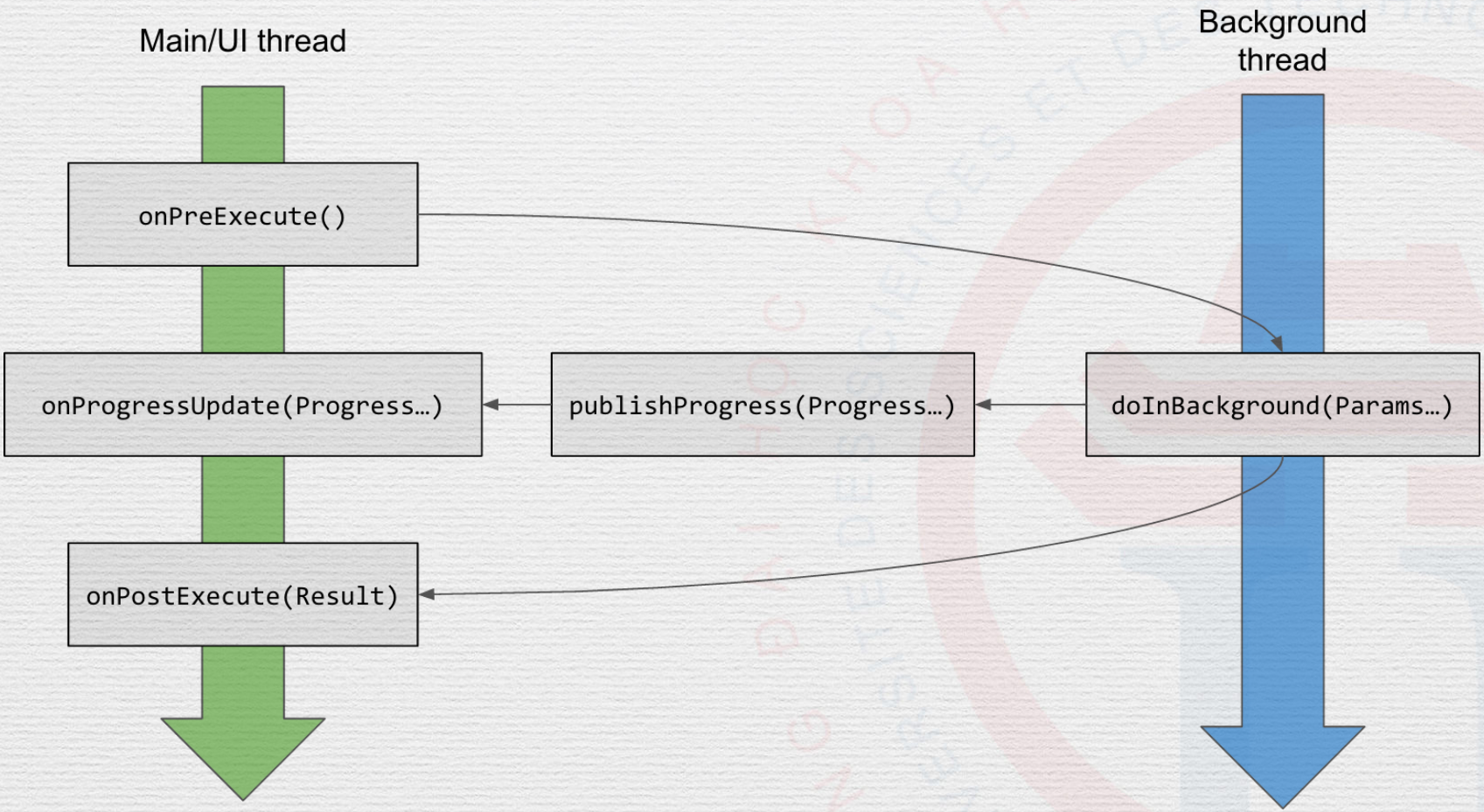
AsyncTask: param

- 3 Generic Types:
  - AsyncTask<Param, Progress, Result>
  - Params: param type to pass to the worker thread
  - Progress: type to report progress back
  - Result: result type to be delivered

AsyncTask: overriding methods

- [optional] `onPreExecute()`: for preparation
- [required] `doInBackground()`: do the real work
- [optional] `onProgressUpdate()`: for updating progress to UI
- [optional] `onPostExecute()`: for delivering result

AsyncTask: methods



AsyncTask: example

- Write a background task to
  - Download a PNG file from the Internet
  - Periodically report the download progress
  - Decode its content to bitmap data
  - Show it on an `ImageView`

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String`

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**
- 2nd param: `Integer`

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**
- 2nd param: `Integer` , for updating the **Percentage** during the progress

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**
- 2nd param: `Integer` , for updating the **Percentage** during the progress
- 3rd param: `Bitmap`

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**
- 2nd param: `Integer` , for updating the **Percentage** during the progress
- 3rd param: `Bitmap` , for the **Decoded Bitmap** for the image

AsyncTask: example params

```
AsyncTask<String, Integer, Bitmap>
```

- 1st param: `String` , for the **URL**
- 2nd param: `Integer` , for updating the **Percentage** during the progress
- 3rd param: `Bitmap` , for the **Decoded Bitmap** for the image

AsyncTask: example methods

Download a PNG file
Report download progress
Decode to bitmap data
Show it on an ImageView

AsyncTask: example methods

Download a PNG file
Report download progress
Decode to bitmap data
Show it on an ImageView

→

AsyncTask: example methods

Download a PNG file
Report download progress
Decode to bitmap data
Show it on an ImageView

→ doInBackground()

AsyncTask: example methods

Download a PNG file		doInBackground()
Report download progress	→	onProgressUpdate()
Decode to bitmap data		
Show it on an ImageView		

AsyncTask: example methods

Download a PNG file		doInBackground()
Report download progress	→	onProgressUpdate()
Decode to bitmap data		doInBackground()
Show it on an ImageView		

AsyncTask: example methods

Download a PNG file		doInBackground()
Report download progress	→	onProgressUpdate()
Decode to bitmap data		doInBackground()
Show it on an ImageView		onPostExecute()

AsyncTask: example code

```
public class WeatherActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        AsyncTask<String, Integer, Bitmap> task = new AsyncTask<>() {
                @Override
                protected void onPreExecute() {
                    // do some preparation here, if needed
                }
                @Override
                protected Bitmap doInBackground(String... params) {
                    // This is where the worker thread's code is executed
                    // params are passed from the execute() method call
                    return null;
                }
                @Override
                protected void onProgressUpdate(Integer... values) {
                    // This method is called in the main thread, so it's possible
                    // to update UI to reflect the worker thread progress here.
                    // In a network access task, this should update a progress bar
                    // to reflect how many percent of data has been retrieved
                }
                @Override
                protected void onPostExecute(Bitmap bitmap) {
                    // This method is called in the main thread. After #doInBackground returns
                    // the bitmap data, we simply set it to an ImageView using ImageView.setImageBitmap()
                    }
                };
        task.execute("http://ict.usth.edu.vn/wp-content/uploads/usth/usthlogo.png");
    }
}
```

Practical Work 14

- «Upgrade» your Thread/Handler combo with an AsyncTask
- Simulate a network request and show on a toast. Hint:
  - sleep() in doInBackground
  - Toast in onPostExecute()