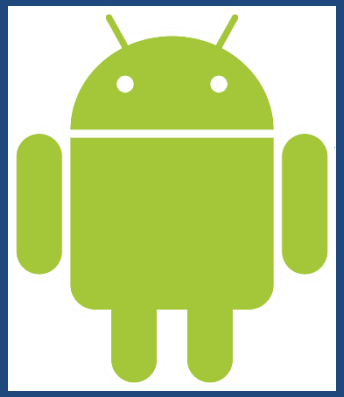
Mobile App Development

AsynchTask



Dr. Christelle Scharff
cscharff@pace.edu
Pace University, USA

Objectives

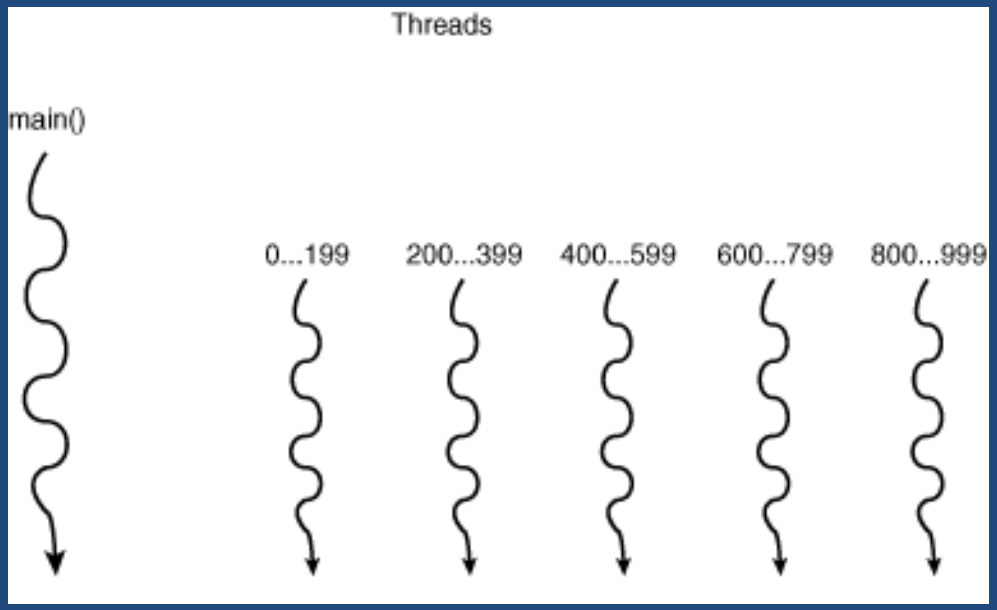
- Refresh your knowledge of Java threads
- Understand what the UI thread is and how to use it
- Understand and use AsyncTask, one of the fundamentals building block of Android, to execute long operations in Android (including accessing data from the Internet)
- Getting familiar with file systems classes and methods (e.g., HttpURLConnection)

Threads and Android UI Thread

What is a Thread?

- In concurrent programming, there are two basic units of execution: *processes* and *threads*
- A process is a collection of:
  - Threads
  - Memory space
  - Permissions
- http://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html
- Each Android app runs in its own Linux process and has its own virtual machine

Threads Operating on Different Sets of Data



Same operation on sets of data

Defining a Thread in Java

- **Choice 1 – with an anonymous class**

```
new Thread(new Runnable(){public void run(){…};
…}).start();
```

- **Choice 2 – without an anonymous class**

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }
}
```

Android UI Thread

- In Android, there is ***only ONE*** MAIN thread or UI thread which updates the UI
- The UI thread runs only one Activity at a time
- **Rule 1** – You can create new threads but you cannot interact with UI from them
- **Rule 2** – You cannot block the UI thread (e.g., long operations block the UI thread)
- If you misuse the UI thread you will get *an App Not Responding Crash* (ANR)

Question

- Provide examples of operations that would require time to be executed (in Java)

UI Thread

- Any long-running operation should take place in its own thread:
  - Acquiring data
  - Opening a network connection
  - Reading from a file
  - …
- Anything that does not involve setup or modification of the UI should not be done in the UI thread

AsyncTask

AsyncTask

- AsyncTask hides the use of threads
- AsyncTask permits proper and easy use of the UI thread
  - It allows to perform background operations
  - It publishes the results on the UI thread (without having to manipulate any thread)
  - It synchronizes with the UI thread to show the progress and completion of the tasks
- Each AsyncTask is executed only once
- http://developer.android.com/reference/android/os/AsyncTask.html

AsyncTask class

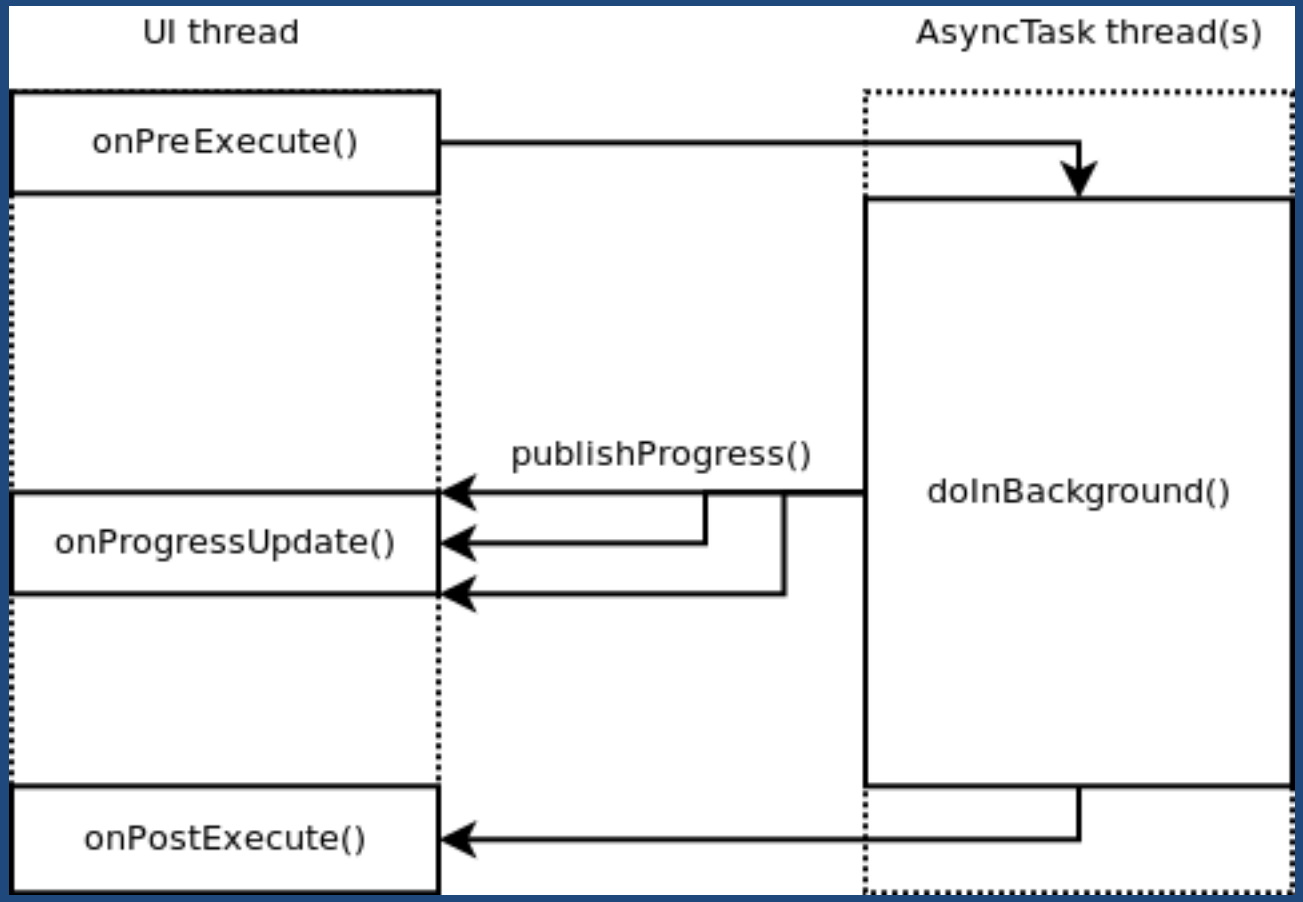
AsyncTask Methods

Protected Methods	
abstract Result	doInBackground(Params... params) Override this method to perform a computation on a background thread.
void	onCancelled(Result result) Runs on the UI thread after `cancel(boolean)` is invoked and `doInBackground(Object[])` has finished.
void	onCancelled() Applications should preferably override `onCancelled(Object)`.
void	onPostExecute(Result result) Runs on the UI thread after `doInBackground(Params...)`.
void	onPreExecute() Runs on the UI thread before `doInBackground(Params...)`.
void	onProgressUpdate(Progress... values) Runs on the UI thread after `publishProgress(Progress...)` is invoked.
final void	publishProgress(Progress... values) This method can be invoked from `doInBackground(Params...)` to publish updates on the UI thread while the background computation is still running.

AsyncTask Methods

- `onPreExecute()` – Run in the main thread. To setup a loading dialog or similar. Called before doInBackground
- `doInBackground()` – Does not run in the main thread. Contain the long operations to be performed. Use publishProgress to update onProgressUpdate
- `onProgressUpdate()` – Run in the main thread. Publish progress in the UI (e.g., status of completion)
- `onPostExecute()` – Run in the main thread. Update the UI
- `onCancelled()` – Cancel the thread
- You should **not** call onPreExecute, onPostExecute, doInBackground, and onProgressUpdate manually
- AsyncTask takes over the main thread with the onPreExecute and onPostExecute methods

AsyncTask Execution



AsyncTask Parameters

- To use AsyncTask, we define a class that extends AsyncTask
- AsyncTask requires 3 generic type parameters
  - AsyncTask<X,Y,Z>
  - X – Input parameters. Passed from the activity to the thread to perform the long operations
  - Y – Progress values. Passed between doInBackground and onProgressUpdate
  - Z – Result value. Returned by onPostExecute and passed back to the activity

LongOperation is an AsyncTask

Do not write classes that extends AsyncTask, generate them using your IDE!

Void is used for an unused type

```
package cs639.pace.com.cs639sampleproject;
import android.os.AsyncTask;

public class LongOperation extends AsyncTask<String, Void, Integer> {

    @Override
    protected Integer doInBackground(String... params) {
        // LONG OPERATION
        return null;
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }

    @Override
    protected void onPostExecute(Integer integer) {
        super.onPostExecute(integer);
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
}
```

Executing an AsyncTask

- An AsyncTask is executed using the execute method

```
LongOperation lop = new LongOperation();
lop.execute("Hello");
```

- To get the result from the AsyncTask

```
Integer myResult = lop.get();
```

Example
Getting Data Online

Getting Data from the Internet

- Accessing data from the Internet using `HttpURLConnection` as follows:
  - Wrap the code in an `AsyncTask`
  - A URL is created with the data to be accessed
  - Open the connection and use the URL using `openHttpConnection()`
  - Configure the request to access the HTTP server (GET, POST etc)
  - Define an `InputStream` through the `HttpURLConnection`
  - Read the data through the `InputStream` and close it
  - Add the required permissions in `AndroidManifest.xml`

Accessing an Image Online

```
try{
    URL url = new URL(params[0]);
    HttpURLConnection con = (HttpURLConnection)url.openConnection();
    if(con.getResponseCode() != 200){
        throw new Exception("Failed to connect");
    }
    InputStream is = con.getInputStream();
    is.close();
    Bitmap bmpa = BitmapFactory.decodeStream(is);
    return bmap;
} catch(Exception e){
    Log.e("Image", "Failed to load image", e);
    Log.e("error", e.getMessage());
}
```

http://developer.android.com/training/basics/network-ops/connecting.html
http://developer.android.com/reference/java/net/HttpURLConnection.html

Review the code

Code to be used in doInBackground

Lab
(also part of your assignment)

- Insert a remote image in the UI of an Activity