Mobile Application Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

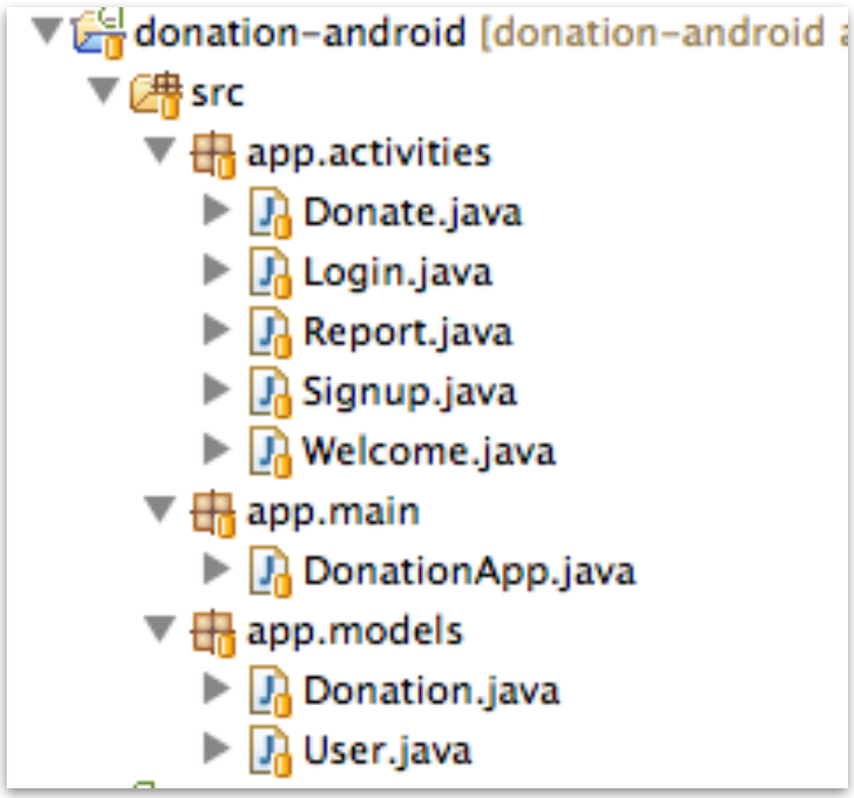
http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

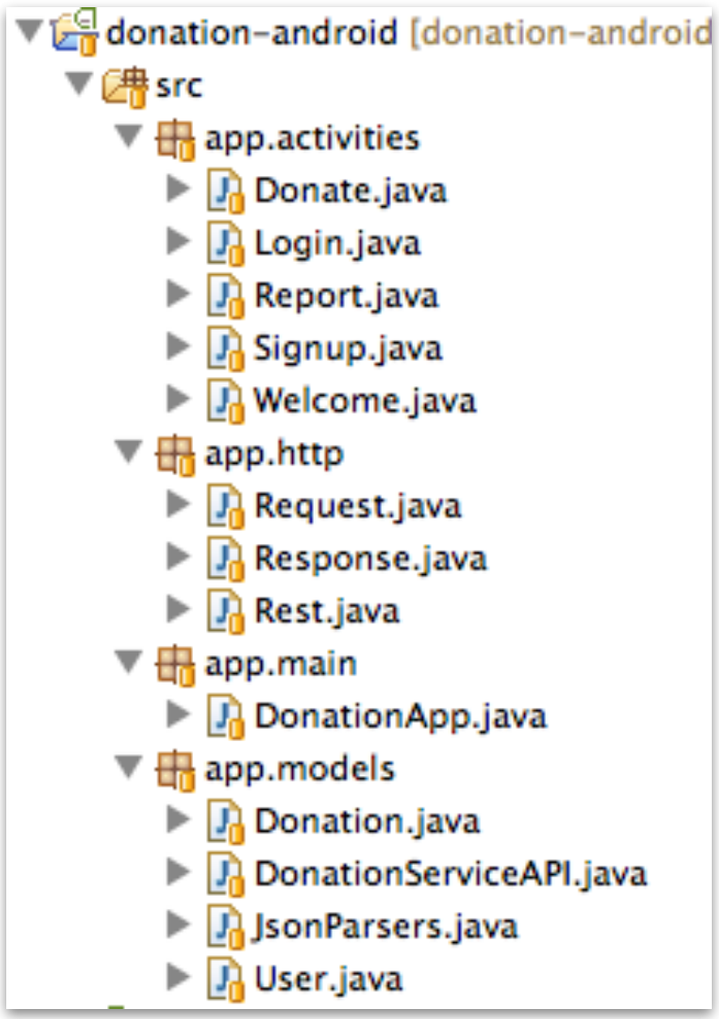
Android <-> Play (2) - Concurrency

donation-android project v2



- **activities**
  - display and hander all UI
- **main**
  - retain application wide data structures (users + donations)
- **models**
  - core information models for the application

donation-android Project v3



- **activities**
  - display and hander all UI
- **main**
  - retain application wide data structures (users + donations)
- **models**
  - Gateway object for accessing donation-service application
  - Local copies of core information models for the application (download from donation-play)
  - Parsers (transformers) for converting objects into format suitable for upload/download to/from donation-service
- **http**
  - General purpose classes to support asynchronous http request/response to/from donation-service

HTTP Requests

- A HTTP request is inherently indeterminate and subject to various error conditions:
  - The URL may be incorrect
  - The Network may be slow or intermittent
  - The DNS may be slow
  - The service itself may be slow
  - The service may crash, or return incorrect or badly formatted results
- For all of these reasons, Android does not permit HTTP requests to be performed directly in Activities
- In order to connect to a web service for any reason, you must initiate another '*Thread of Execution*'

Concurrency

- Concurrency is the ability to run several programs or several parts of a program in parallel.

- If a time consuming task can be performed asynchronously or in parallel, this improves the throughput and the interactivity of the program.

- A modern computer has several CPU's or several cores within one CPU. The ability to leverage these multi-cores can be the key for a successful high-volume application.

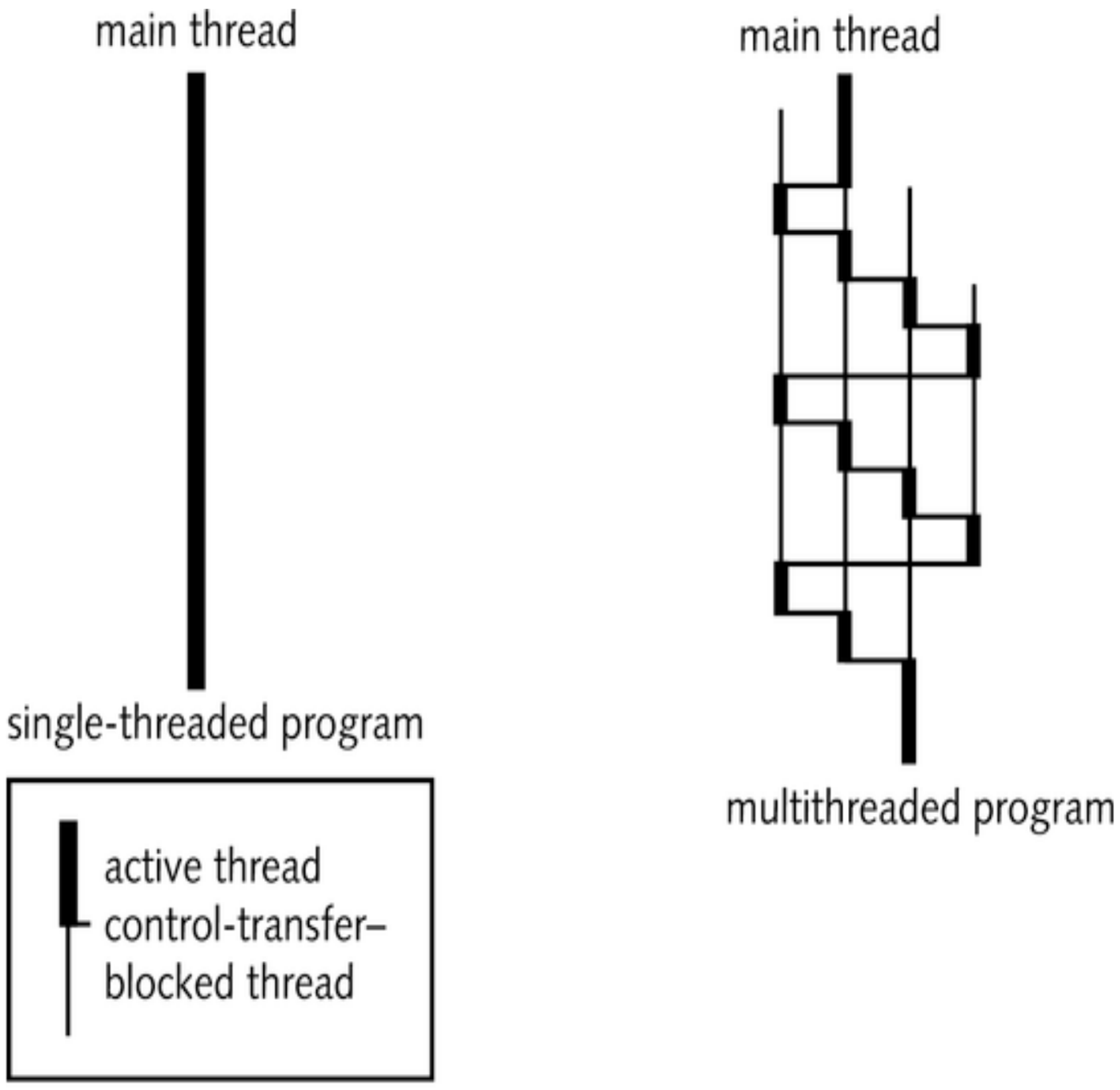
Process & Threads

- A process runs independently and isolated of other processes.
  - It cannot directly access shared data in other processes.
  - The resources of the process, e.g. memory and CPU time, are allocated to it via the operating system.

- A thread is a lightweight process.
  - It has its own call stack, but can access shared data of other threads in the same process. Every thread has its own memory cache.
  - If a thread reads shared data it stores this data in its own memory cache.
  - A thread can re-read the shared data.

Java Programs & Threads

- A Java application runs by default in one process.
- Within a Java application you work with several threads to achieve parallel processing or asynchronous behavior.



Android Processes and Threads

- When an application starts the Android system starts a new Linux process for the application with a single thread of execution.

- By default, all components of the same application run in the same process and thread (called the "main" thread).

- However, you can arrange for different components in your application to run in separate processes, and you can create additional threads for any process.

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

Android Threads

- When an application is launched, the system creates a thread of execution for the application, called “the main thread."

- This thread is very important because 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”.

Main (UI) Threads

- The system does not create a separate thread for each instance of a component.

- All components that run in the same process are instantiated in the UI thread, and system calls to each component are dispatched from that thread.

- Consequently, methods that respond to system callbacks (such as onKeyDown() to report user actions or a lifecycle callback method) always run in the UI thread of the process.

Blocking the Main Thread

- 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.

- 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.

Android AsyncTask Class

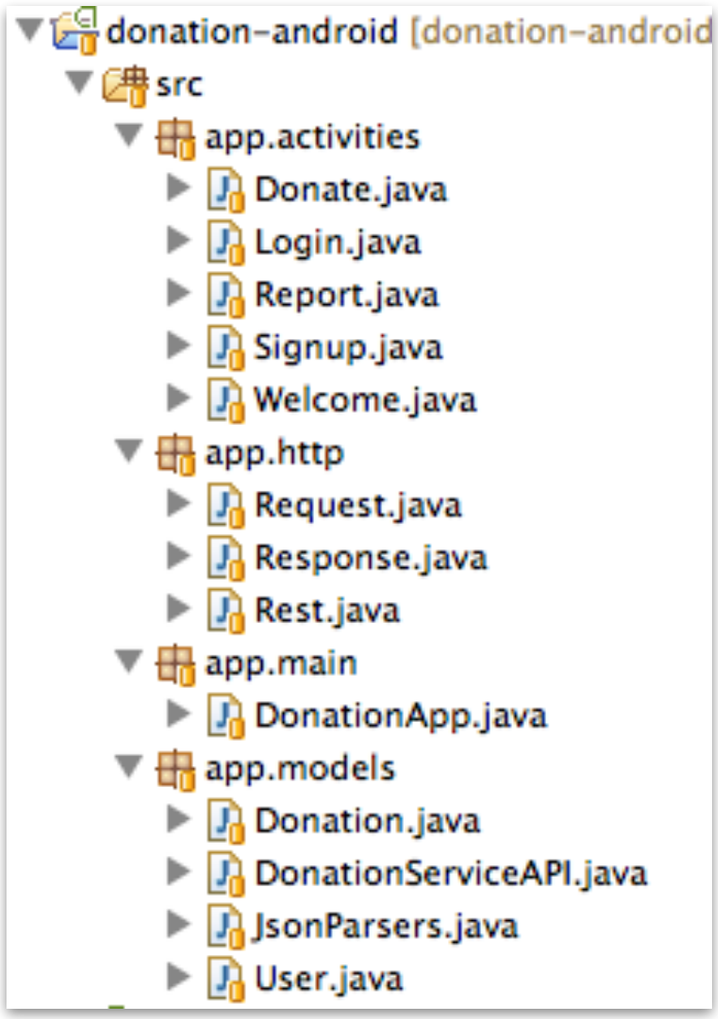
- AsyncTask allows you to perform asynchronous work on your user interface.

- It performs the blocking operations in a **worker thread** and then publishes the results on the UI thread.

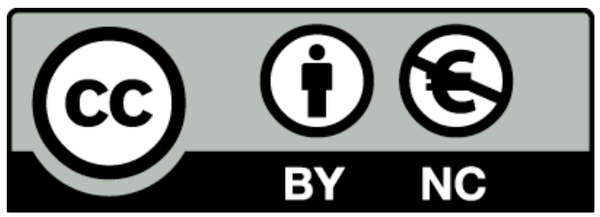
- You subclass AsyncTask and implement the doInBackground() callback method, which runs in a pool of background threads.

- To update your UI, you implement onPostExecute(), which delivers the result from doInBackground() and runs in the UI thread, so you can safely update your UI..

donation-android Project v4



- **http**
  - General purpose classes to support **asynchronous** http request/ response to/from donation-service
  - These requests are performed in a separate thread of execution



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see http://creativecommons.org/licenses/by-nc/3.0/

Some of this material is adapted from
http://www.vogella.com/articles/JavaConcurrency/article.html
An excellent source for well structured tutorials and explanations of all thing related ot Java, Eclipse and Android development