Modern Related Technology on M obile Devices

Lec 8: Thread I

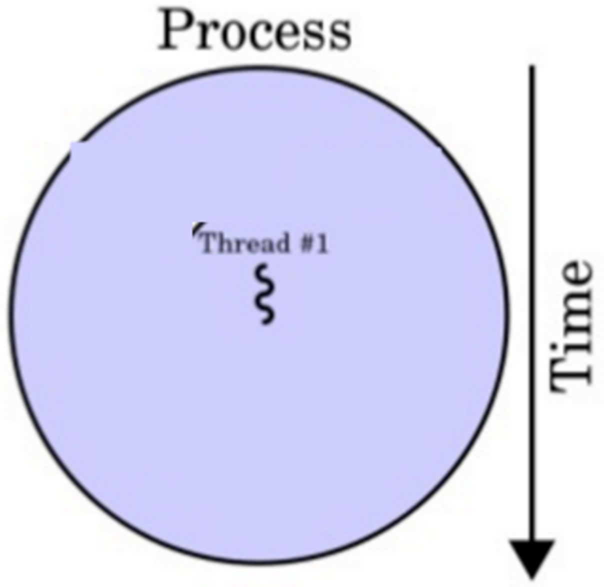
Ekarat Rattagan, PhD

Outline

- Process & Thread

Process & Thread

- When an application component, <activity>, <service>, <receiver>, and <provider>, starts a
nd the application does not have any other comp
onents running, the Android system starts a new
Linux process for the applicatio :hr
ead of execution.
- By default, all components of the same application run in the same process and thread (called the "main" thread).



Threads

- The main 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. As such, the main thread is also sometimes called the “UI thread.”

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 a ctions or a lifecycle callback method) always run in the UI thread of the process.

Threads

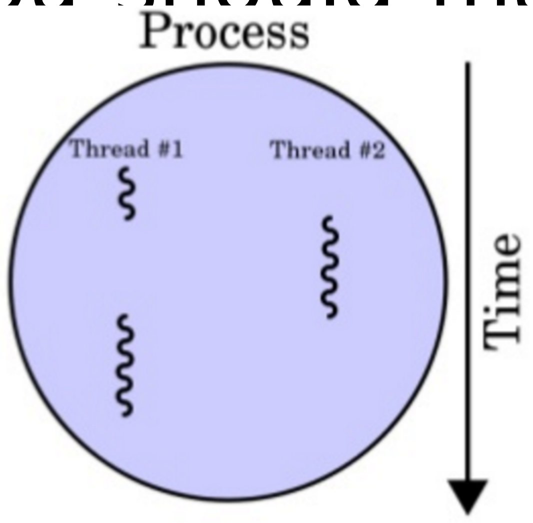
- When your app performs intensive work in response to user interaction, this single thread model can yield poor performance.
- Specifically, if everything is happening in the UI thread, performing long operations such as network access or database queries will block the whole UI. When the thread is blocked, no events can be dispatched, including drawing events.
- Even worse, if the UI thread is blocked for about 5 seconds the user is presented with the infamou

Threads

- Additionally, the Android 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. Thus, the re are simply two rules to Android's single thread model:
  1. Do not block the UI thread
  2. Do not access the Android UI toolkit from outside the UI thread

Worker threads

- Because of the single threaded model described above, it's vital to the responsiveness of your application's UI that you do not block the UI thread. If you have operations to perform that are not instantaneous, you should make sure to do them in separate thr ound" or "worker" threads).



Worker threads

- To fix this problem, Android offers several ways to access the UI thread from other threads. Here is a list of methods that c

  - Activity.runOnUiThread (Runna
  - View.post (Runnable)
  - View.postDelayed (Runnable, l

```
Example:

public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            // a potentially  time consuming
task
            final Bitmap bitmap =
                    processBitMap("image.png");
             mImageView.post(new Runnable() {
                public void run() {

mImageView.setImageBitmap(bitmap);
                }
            });
        }
    }).start();
```

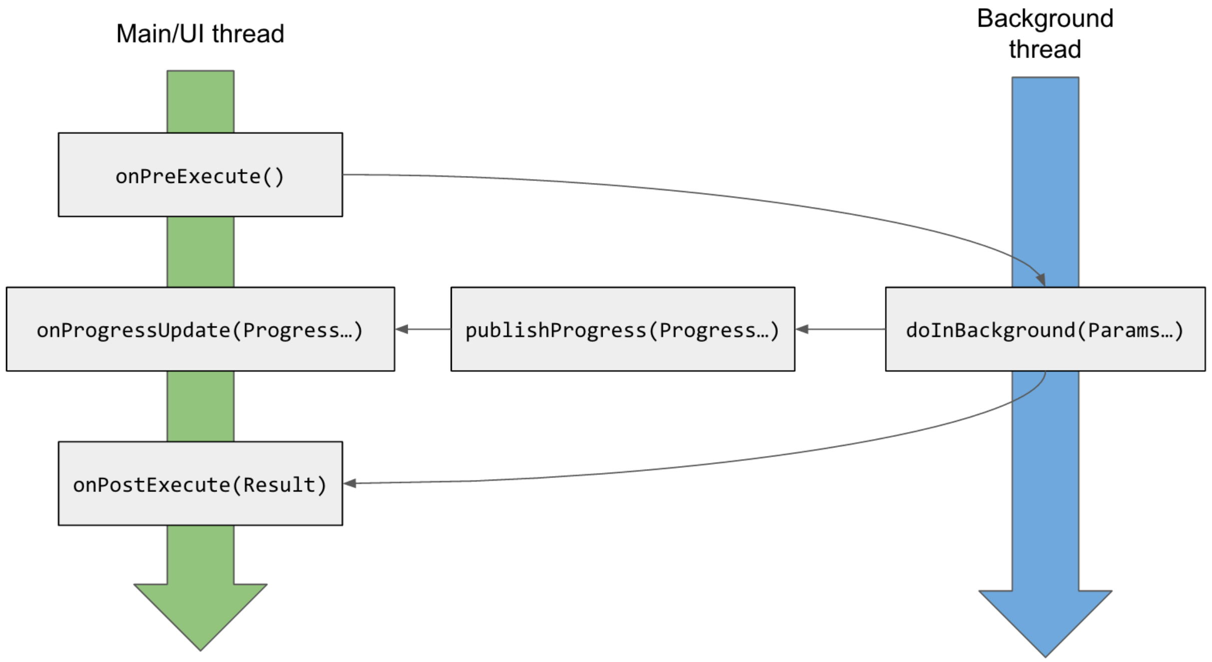
AsyncTask

- AsyncTask enables proper and easy use of the UI thread.
- This class allows you to perform background operations and publish results on the UI thread w ithout having to manipulate threads and/or handl ers.

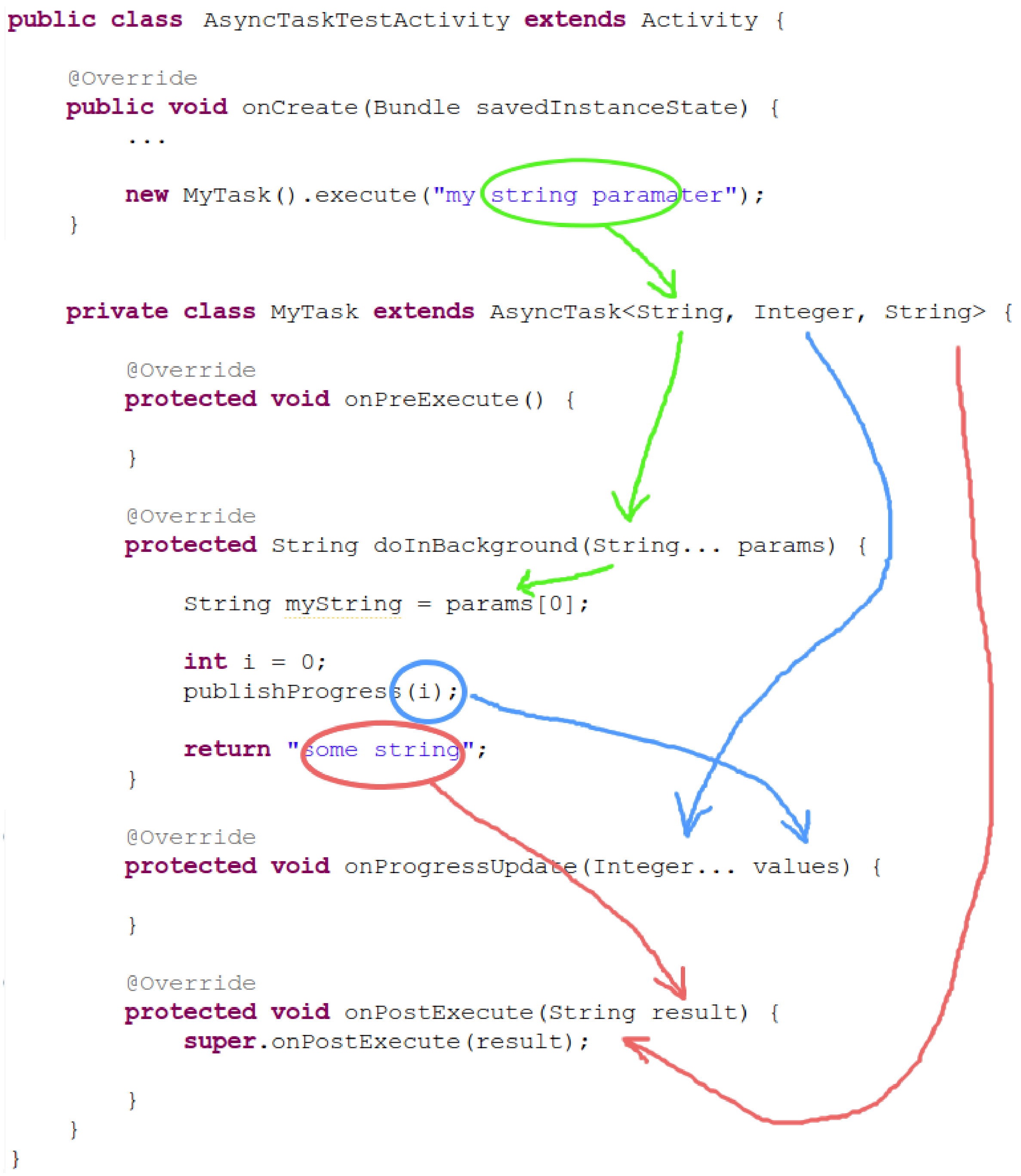
AsyncTask

- An asynchronous task is defined by 3 generic types, called
  - Params
  - Progress
  - Result

And 4 steps, called
  - onPreExecute()
  - doInBackground(Params… params)
  - onProgressUpdate (Params… params)
  - onPostExecute(Result result)

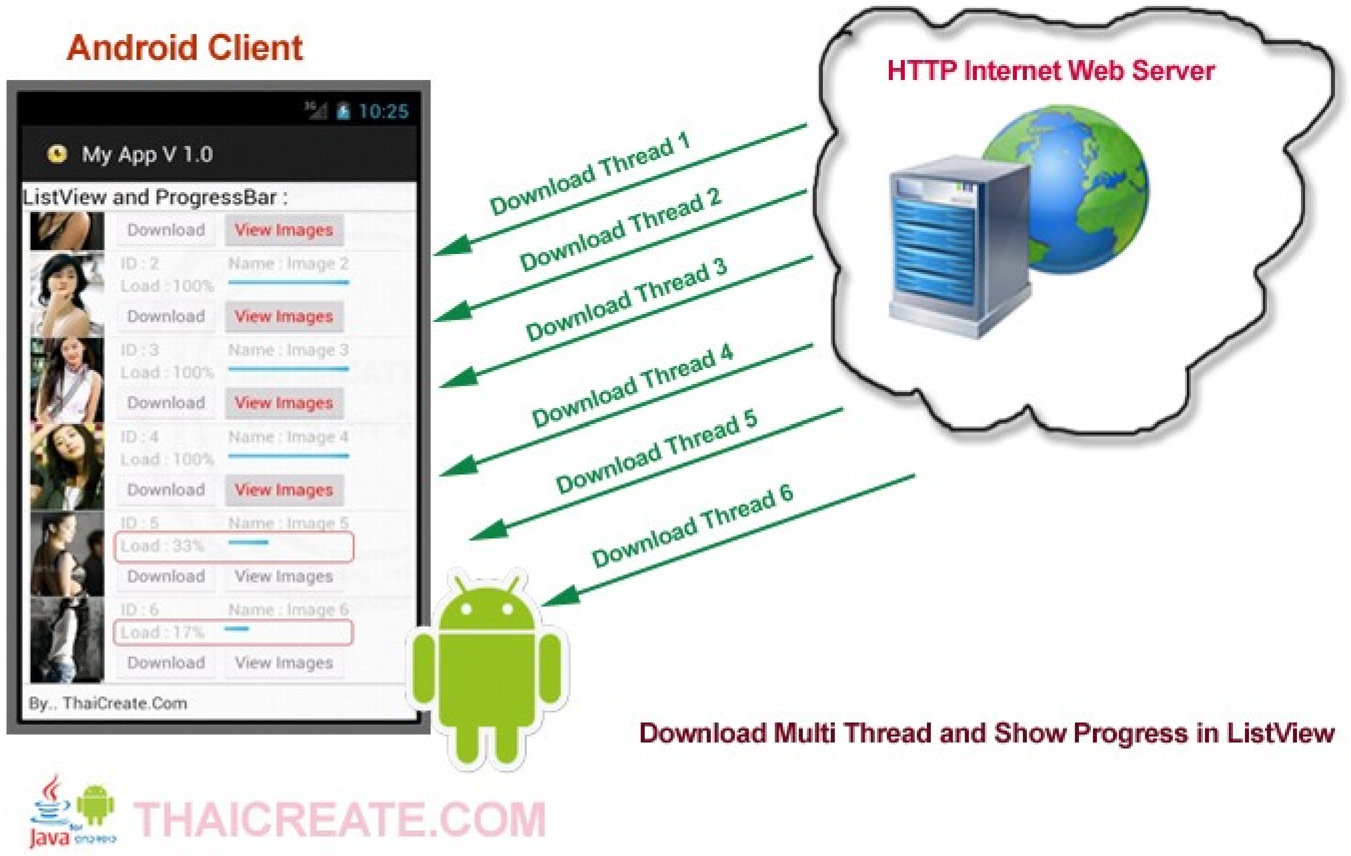


AsyncTasl



```
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);

        }
    }
}
```

Thread application



http://resource.thaicreate.com/upload/tutorial/android-progressbar-listview-download-file-00.jpg?v=1001