CS 580 Client-Server Programming
Fall Semester, 2012
Doc 7 Android Concurrency
Sept 18, 2012

Assignment 1 Comments

IO & Names

```
public void GetValueFromHash( String key){
        System.out.println("-----------------------------------------\n");
        System.out.println("Value of " + key + " in Dictionary is = "  +
                                VarHashDictionary.get(key) + "\n");
        System.out.println("-----------------------------------------\n");
}
```

static

```
public static void put(String key, Object value ) {

}
```

Network Permission

Using the Internet

In Manifest file

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Concurrency in Android

Processes and Threads

Processes have own address space

- Take longer to start
- Consume more memory

Threads share address space

Android, Processes and Threads

Android application starts with

- One process running one thread
- The thread is called the main or UI thread
- Activity code runs in main (UI) thread

Can create more threads to run in same process

Can configure activities to run in separate processes

- Not as common as creating threads

Android Thread Rules

Don't block the UI thread

Activity code runs on the UI thread
Create threads to perform long operations

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

Use the following to access UI thread
Activity.runOnUiThread(Runnable)
View.post(Runnable)
View.postDelayed(Runnable, long)

Actually these are the rules for all UI frameworks that I have used

Android Background Tools

Java threads

Handler

- Messages
- Runnables

AsyncTask

Services

AsyncTask

Why AsyncTask

Make it easier to deal with threads

Handle the common case for you

AsyncTask

Replaces threads & Messages

Android 1.5

Subclass AsyncTask

onPreExecute()

Run in UI thread

Done first

doInBackground(Params...)

Run in separate thread

publishProgress(Progress...)

Call in doInBackground() to register progress

onProgressUpdate(Progress...)

Run in UI thread

Called by publishProgress

onPostExecute(Result)

Run in UI thread

Run after doInBackground ends

Rules

The AsyncTask subclass instance must be created on the UI thread

execute(Params...)

Starts the task

Must be invoked on the UI thread

Do not call manually

onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...)

The task can be executed only once

AsyncTask Types

private class SampleTask extends AsyncTask<Params, Progress, Result>

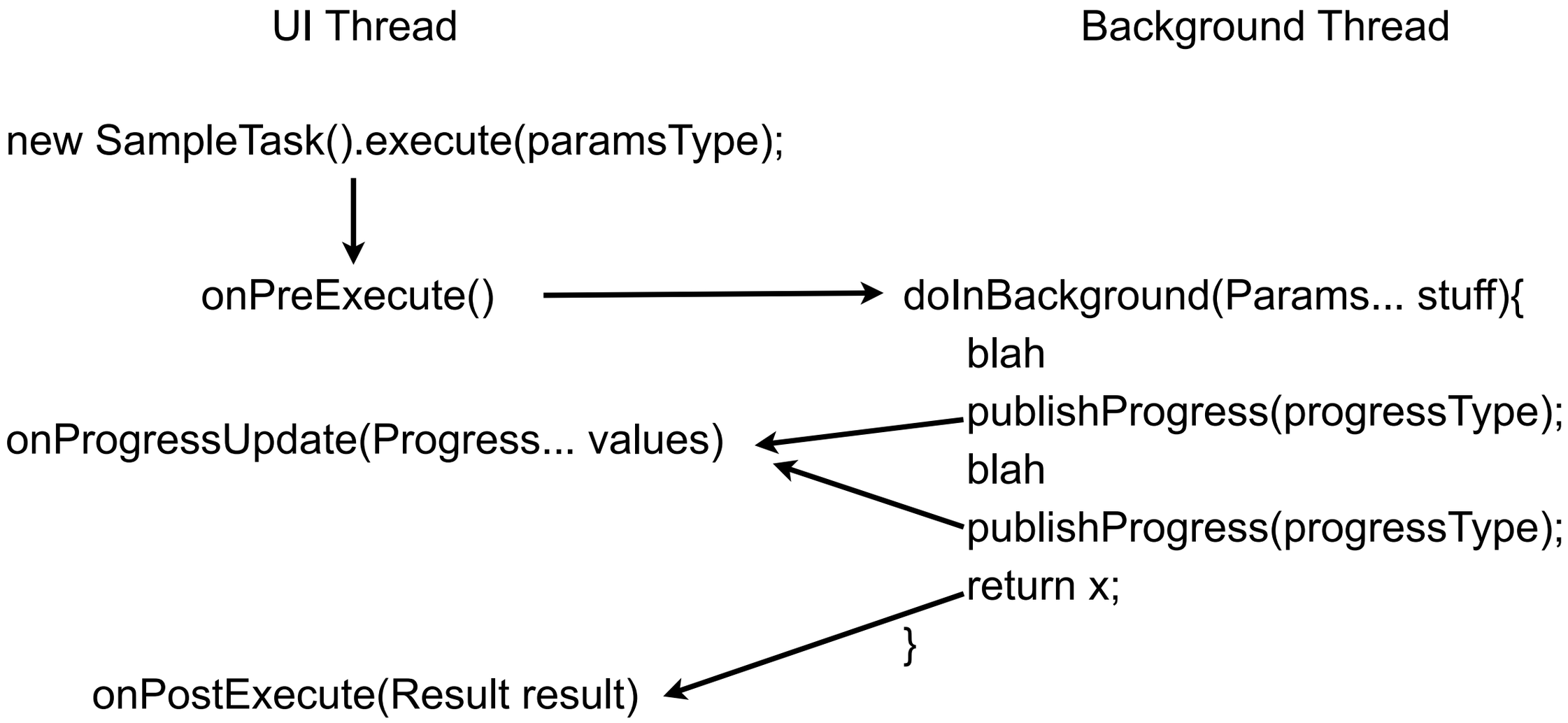
Params
- Type of argument for doInBackground()
- execute()

Progress
- Type of argument for publishProgress()
- onProgressUpdate()

Result
- Return type for doInBackground()
- Type of argument for onPostExecute()

How it Works

private class SampleTask extends AsyncTask<Params, Progress, Result>



Example

Loops in the background and displays Toast

ThreadExample

```
public class ThreadExample extends Activity {

    private class SampleTask extends AsyncTask<String, String, Void> {
        protected Void doInBackground(String... words) {
            for (String word : words) {
                publishProgress(word);
                SystemClock.sleep(1000);
            }
            return (null);
        }

        protected void onPostExecute(Void unused) {
            Toast.makeText(ThreadExample.this, "Done", Toast.LENGTH_SHORT)
                    .show();
        }
```

ThreadExample

```
        protected void onPreExecute() {
            Toast.makeText(ThreadExample.this, "Start", Toast.LENGTH_SHORT).show();
        }

        protected void onProgressUpdate(String... word) {
            Toast.makeText(ThreadExample.this, word[0], Toast.LENGTH_SHORT).show();
        }
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onStart() {
        super.onStart();
        String[] text = { "Bat", "cat", "dat", "fat", "hat", "mat" };
        new SampleTask().execute(text);
    }
}
```