The AsyncTask Framework: Example Application

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Recognize the capabilities provided by the Android AsyncTask framework
- Know which methods are provided by AsyncTask class
- Understand what black-box & white-box framework are... & how AsyncTask implements both types of frameworks
- Learn how the AsyncTaskInterrupted program works

See github.com/douglascraigschmidt/POSA/tree/master/ex/M5/GCD/AsyncTaskInterrupted
Runtime Behavior of the AsyncTaskInterrupted App
Runtime Behavior of the AsyncTaskInterrupted App

- Use AsyncTasks & a ThreadPoolExecutor to compute the greatest common divisor (GCD) of two numbers, which is the largest positive integer that divides two integers without a remainder.
Runtime Behavior of the AsyncTaskInterrupted App

- Use AsyncTasks & a ThreadPoolExecutor to compute the greatest common divisor (GCD) of two numbers, which is the largest positive integer that divides two integers without a remainder.

The user can cancel AsyncTask computations at any time.
Runtime Behavior of the AsyncTaskInterrupted App

- Use AsyncTasks & a ThreadPoolExecutor to compute the greatest common divisor (GCD) of two numbers, which is the largest positive integer that divides two integers without a remainder.

The device’s runtime configuration can also change at any time without affecting running computations.
Implementation of the AsyncTaskInterrupted App
Implementation of the AsyncTaskInterrupted App

- This app showcases the black-box & white-box frameworks in Android’s AsyncTask class

See [github.com/douglascraigschmidt/POSA/tree/master/ex/M5/GCD/AsyncTaskInterrupted](github.com/douglascraigschmidt/POSA/tree/master/ex/M5/GCD/AsyncTaskInterrupted)
Implementation of the AsyncTaskInterrupted App

• This app showcases the black-box & white-box frameworks in Android’s AsyncTask class

Super class automatically logs lifecycle hook method calls to aid debugging
Implementation of the AsyncTaskInterrupted App

- This app showcases the black-box & white-box frameworks in Android’s AsyncTask class

Start & cancels AsyncTasks that repeatedly compute GCD of two random #’s
Implementation of the AsyncTaskInterrupted App

- This app showcases the black-box & white-box frameworks in Android’s AsyncTask class

Stores state (including the AsyncTasks & ThreadPoolExecutor) that’s passed between instances of the MainActivity after runtime configuration changes
Implementation of the AsyncTaskInterrupted App

- This app showcases the black-box & white-box frameworks in Android’s AsyncTask class

Extends AsyncTask & in a ThreadPoolExecutor thread repeatedly computing the GCD of two numbers in a manner that can be cancelled at any point
Implementation of the AsyncTaskInterrupted App

- We'll now analyze the source code for this app.

```java
public class GCDAsyncTask extends AsyncTask<Integer, String, Boolean> {
    // Passed to doInBackground()
    Integer, // Passed to onProgressUpdate()
    String, // Returned from doInBackground()
    // and passed to onPostExecute()
    Boolean>

    /**
    * Debugging tag used by the Android logger.
    */
    private final String TAG = getClass().getSimpleName();

    /**
    * A reference to the MainActivity.
    */
    private WeakReference<MainActivity> mActivity;

    /**
    * Random number generator.
    */
    private final Random mRandom;

    /**
    * Keeps track of the AsyncTask number.
    */
    private int mAsyncTaskNumber;

    public class MainActivity extends LifecycleLoggingActivity {

        /**
        * EditText field for entering the desired number of iterations.
        */
        private EditText mCountEditText;

        /**
        * Number of times to iterate if the user doesn't specify
        * otherwise.
        */
        private final static int sDEFAULT_COUNT = 100000000;

        /**
        * Number of threads to put in the ThreadPoolExecutor.
        */
        private final static int sMAX_TASK_COUNT = 2;

        /**
        * Keeps track of whether the edit text is visible for the
        * user to enter a count.
        */
        private boolean mIsEditTextVisible = false;

        /**
        * Reference to the "set" floating action button.
        */
        private FloatingActionButton mSetFab;

        See github.com/douglastschmidt/POSA/tree/master/ex/M5/GCD/AsyncTaskInterrupted
```
End of the AsyncTask Framework: Example Application