Concurrency

Mobile Application Development

Using AsyncTask

- **The problem with Java threads is that they are unable to alter the UI components.**

- **The Android** AsyncTask **is an abstract class for managing background operations that are expected to manipulate or perform asynchronous work on your user interface.**

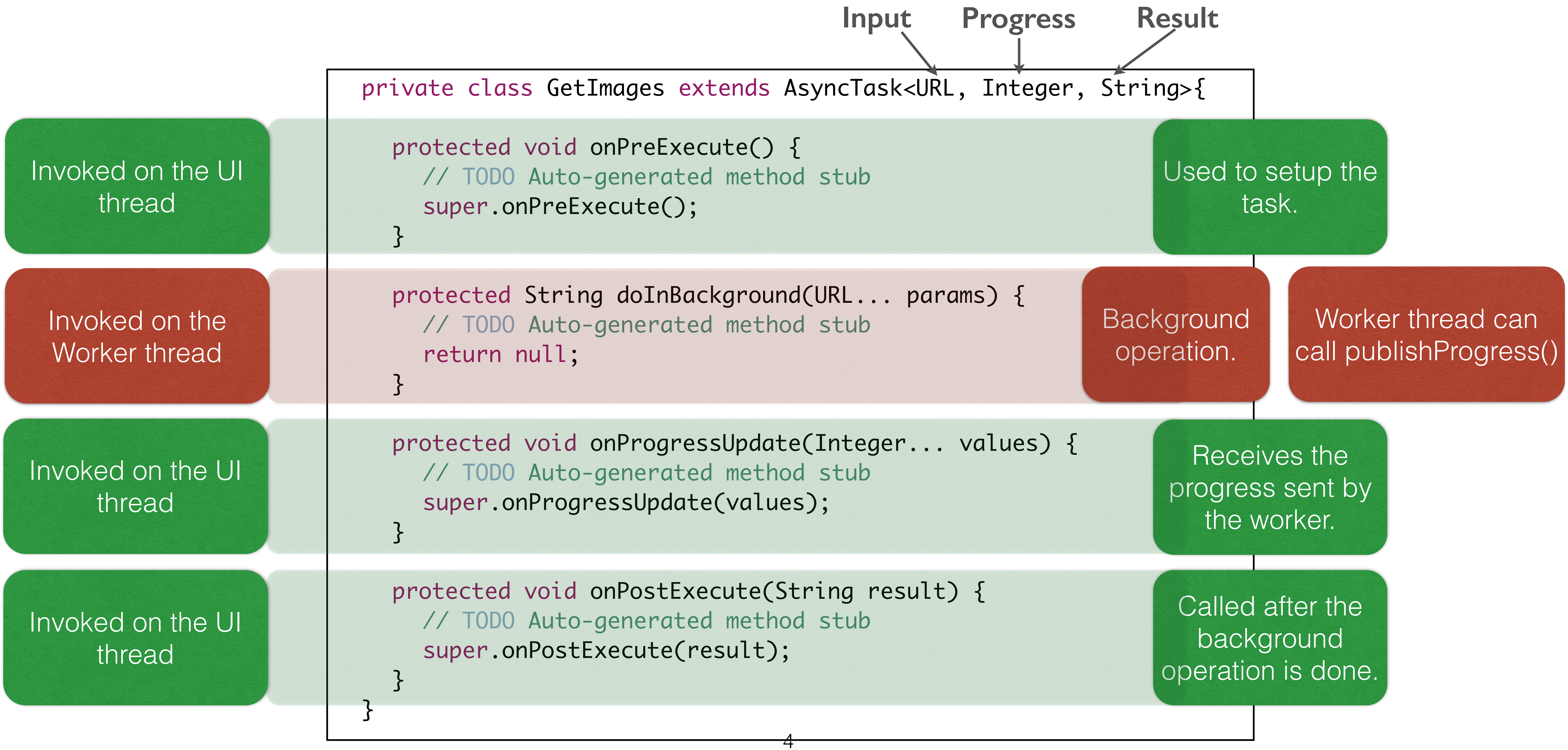
- **To use the** AsyncTask**, you have to create a subclass of** AsyncTask

Using AsyncTask

- **An asynchronous task is defined by 3 generic types**
  - ***Params***, the type of the parameters sent to the task upon execution.
  - ***Progress***, the type of the progress units published during the background computation.
  - ***Result***, the type of the result of the background computation.

- **To use the `AsyncTask`, you have to create a subclass of `AsyncTask` that implements the methods:**
  - `onPreExecute` - Runs in main thread, for setup before doing the background work.
  - `doInBackground` - Runs in worker thread, this is where the background work is done.
  - `onProgressUpdate` - Runs in main thread, called whenever the task wants to send an update to the main tread.
  - `onPostExecute` - Runs in main thread, called after the task is completed.

Using AsyncTask



Input Progress Result

```
private class GetImages extends AsyncTask<URL, Integer, String>{

    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
    }

    protected String doInBackground(URL... params) {
        // TODO Auto-generated method stub
        return null;
    }

    protected void onProgressUpdate(Integer... values) {
        // TODO Auto-generated method stub
        super.onProgressUpdate(values);
    }

    protected void onPostExecute(String result) {
        // TODO Auto-generated method stub
        super.onPostExecute(result);
    }
}
```

Invoked on the UI thread

Used to setup the task.

Invoked on the Worker thread

Background operation.

Worker thread can call publishProgress()

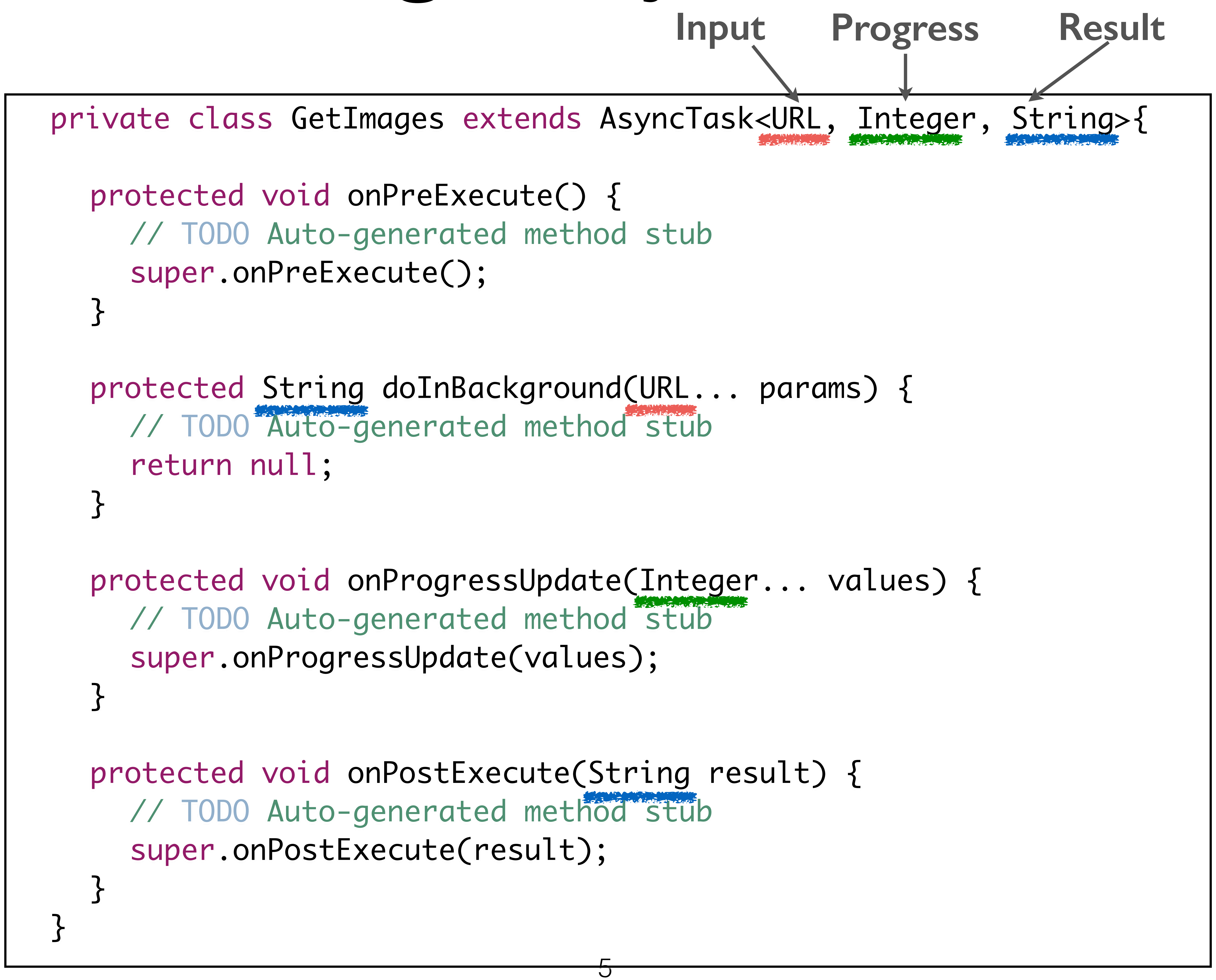
Invoked on the UI thread

Receives the progress sent by the worker.

Invoked on the UI thread

Called after the background operation is done.

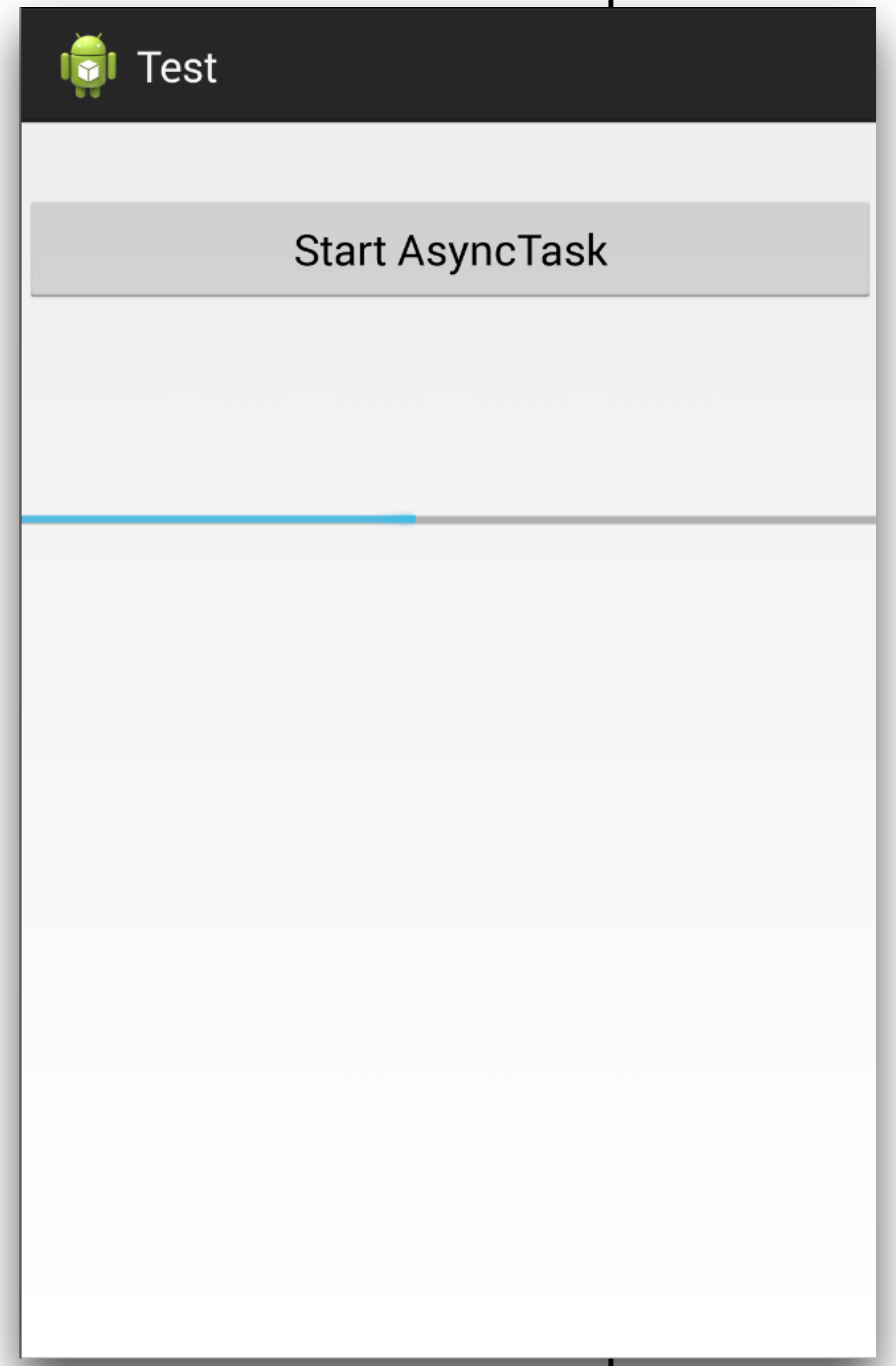
Using AsyncTask

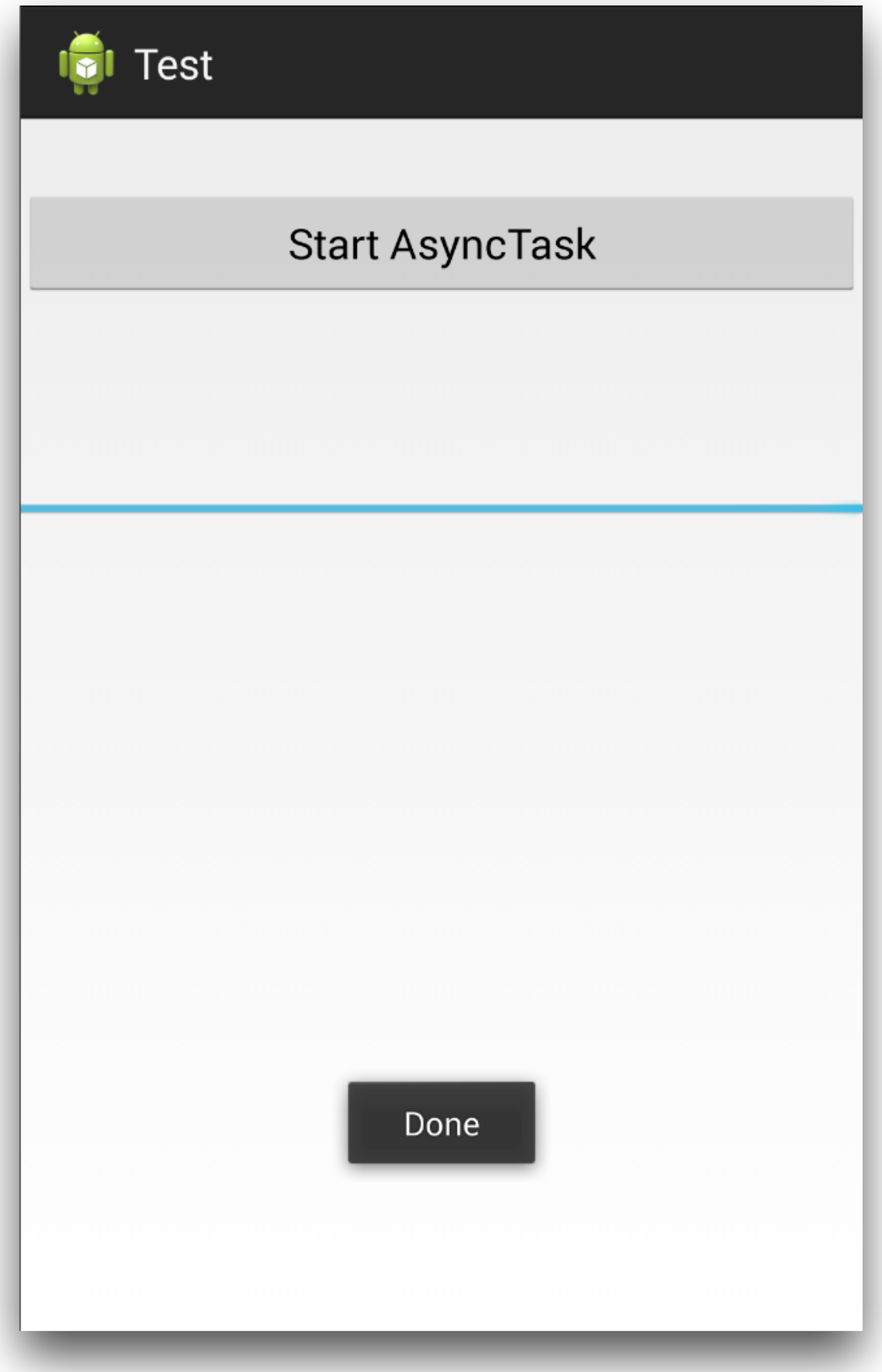


Input Progress Result

```
private class GetImages extends AsyncTask<URL, Integer, String>{

   protected void onPreExecute() {
      // TODO Auto-generated method stub
      super.onPreExecute();
   }

   protected String doInBackground(URL... params) {
      // TODO Auto-generated method stub
      return null;
   }

   protected void onProgressUpdate(Integer... values) {
      // TODO Auto-generated method stub
      super.onProgressUpdate(values);
   }

   protected void onPostExecute(String result) {
      // TODO Auto-generated method stub
      super.onPostExecute(result);
   }
}
```

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.button1).setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
              new MyTask().execute(1000);
          }
       });
    }

    private class MyTask extends AsyncTask<Integer, Integer, Integer>{
       ProgressBar pb;
       @Override
       protected void onPreExecute() {
          pb = (ProgressBar) findViewById(R.id.progressBar1);
          pb.setProgress(0);
          pb.setMax(100);
       }

       @Override
       protected Integer doInBackground(Integer... params) {
          int count = params[0];
          for(int i=0; i<100; i++){
             for(int j=0; j<count;j++){
             }
             publishProgress(i+1);
          }
          return count * 100;
       }

       @Override
       protected void onProgressUpdate(Integer... values) {
          pb.setProgress(values[0]);
       }

       @Override
       protected void onPostExecute(Integer result) {
          Toast.makeText(MainActivity.this, "Done", Toast.LENGTH_LONG).show();
       }
    }
}
```


Test
Start AsyncTask


Test
Start AsyncTask
Done