Université Mohamed V
***Faculté des sciences***
Département d'informatique

***2017/2018***

Module:
**Mobile and cloud computing**

# Asynctasks

***Pr. Oussama REDA***

## AsyncTask

## <Params, Progress, Result>

AsyncTask
<Params, Progress, Result>

**doInBackground()**

$\boxed{\text{onPreExecute()}}$

AsyncTask
<Params, Progress, Result>

doInBackground()

AsyncTask

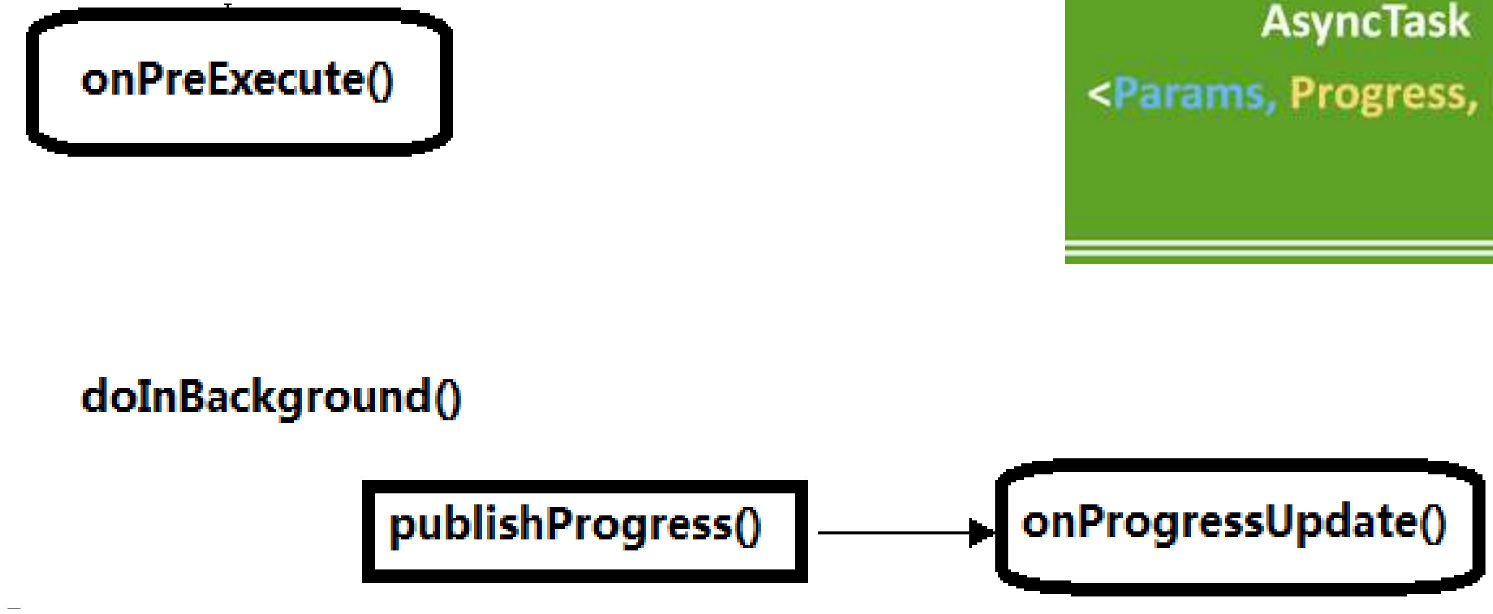
<Params, Progress, Result>

onPreExecute()

doInBackground()

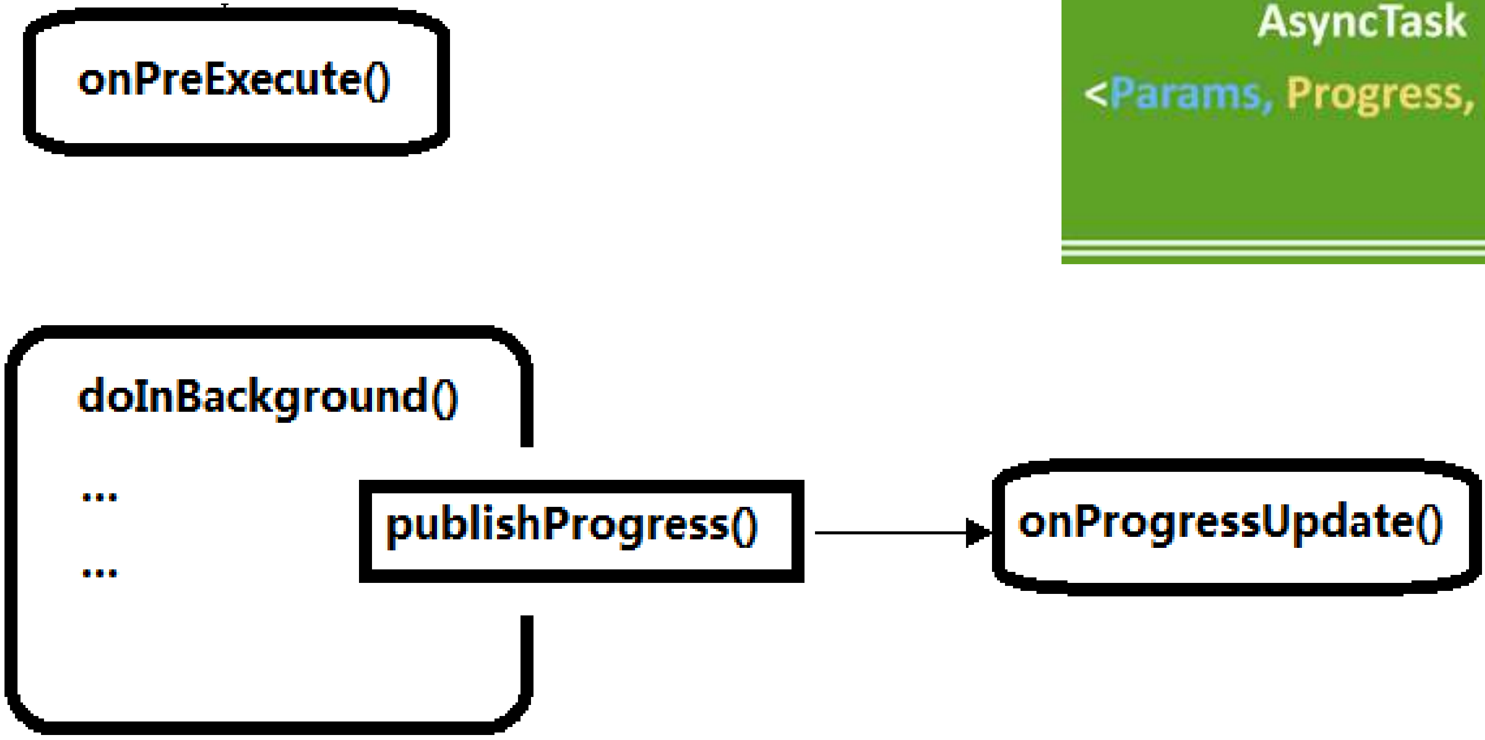
onProgressUpdate()

## AsyncTask
<Params, Progress, Result>


onPreExecute()
doInBackground()
publishProgress()
onProgressUpdate()

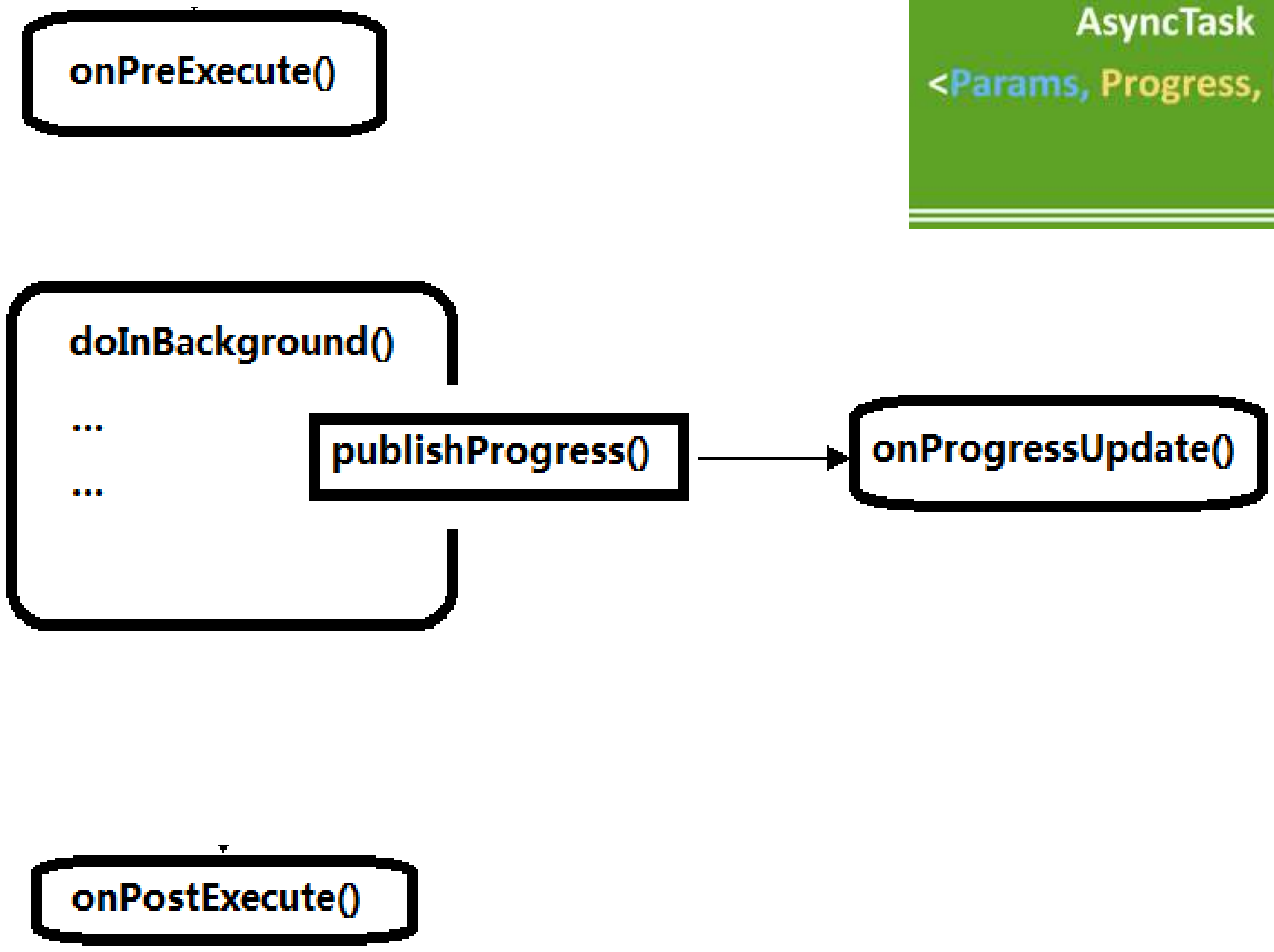
AsyncTask

<Params, Progress, Result>


onPreExecute()
doInBackground()
...
...
publishProgress()
onProgressUpdate()

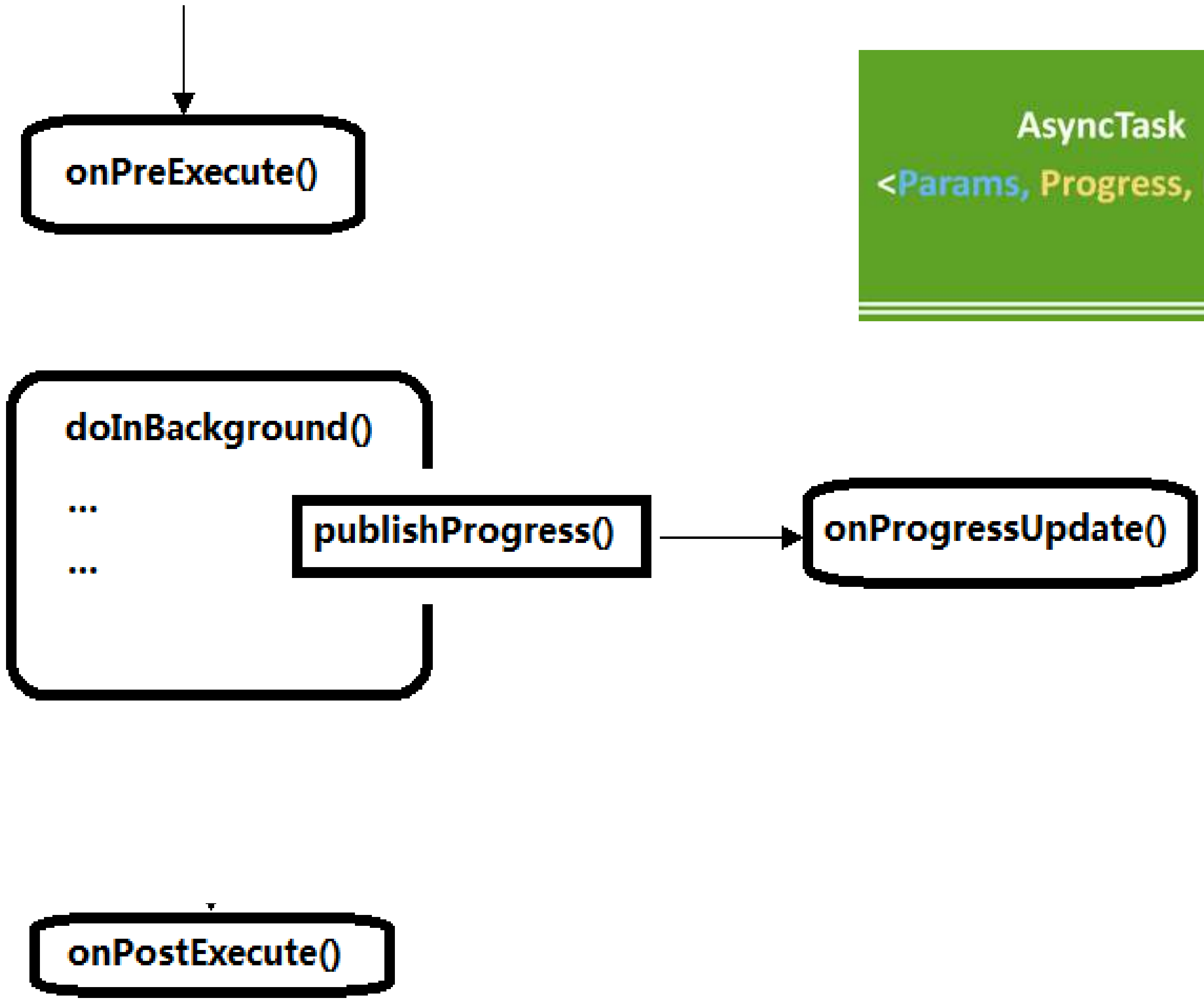
## AsyncTask

<Params, Progress, Result>


onPreExecute()
doInBackground()
...
...
publishProgress()
onProgressUpdate()
onPostExecute()

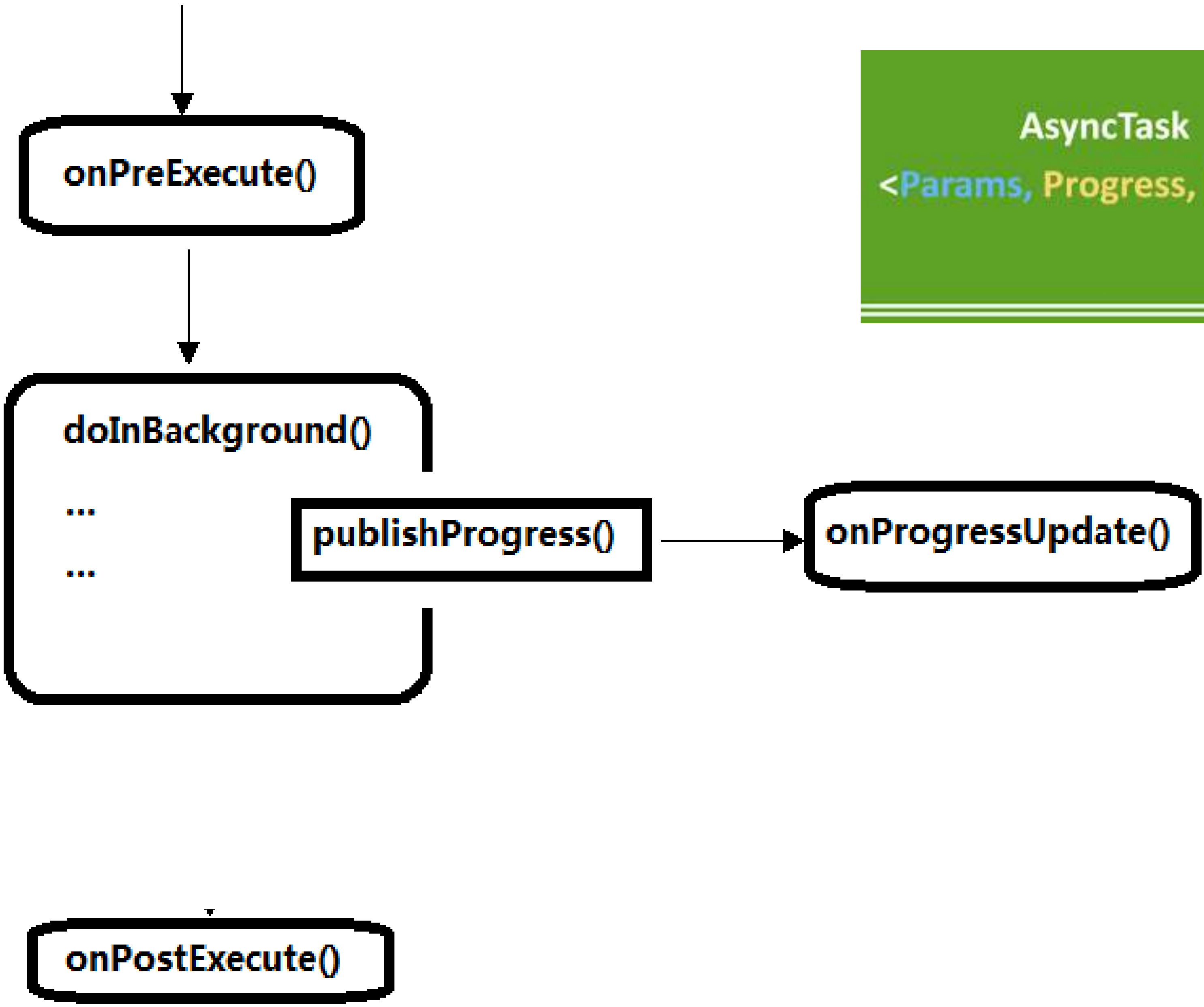
AsyncTask

<Params, Progress, Result>


onPreExecute()
doInBackground()
...
...
publishProgress()
onProgressUpdate()
onPostExecute()

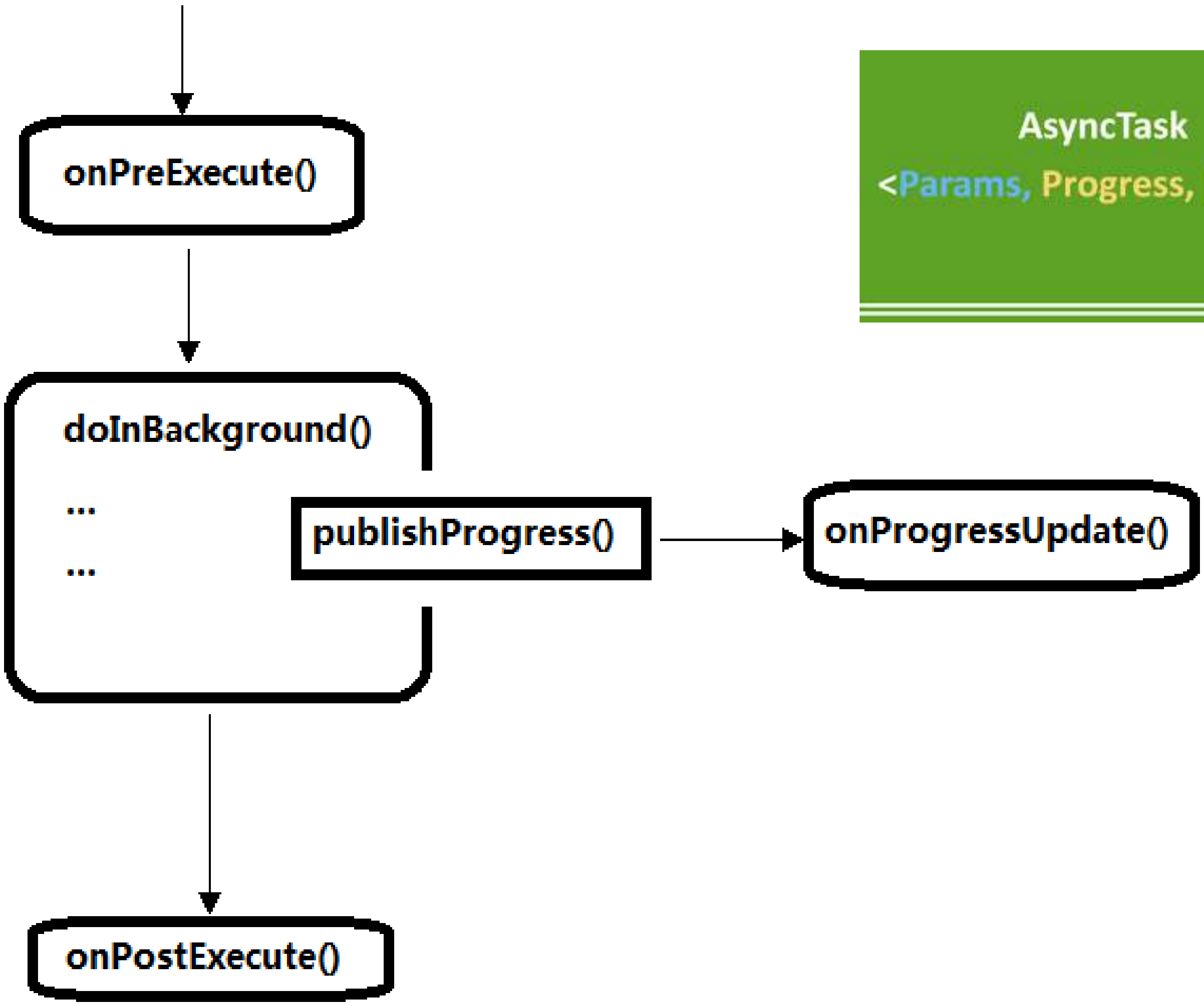
AsyncTask

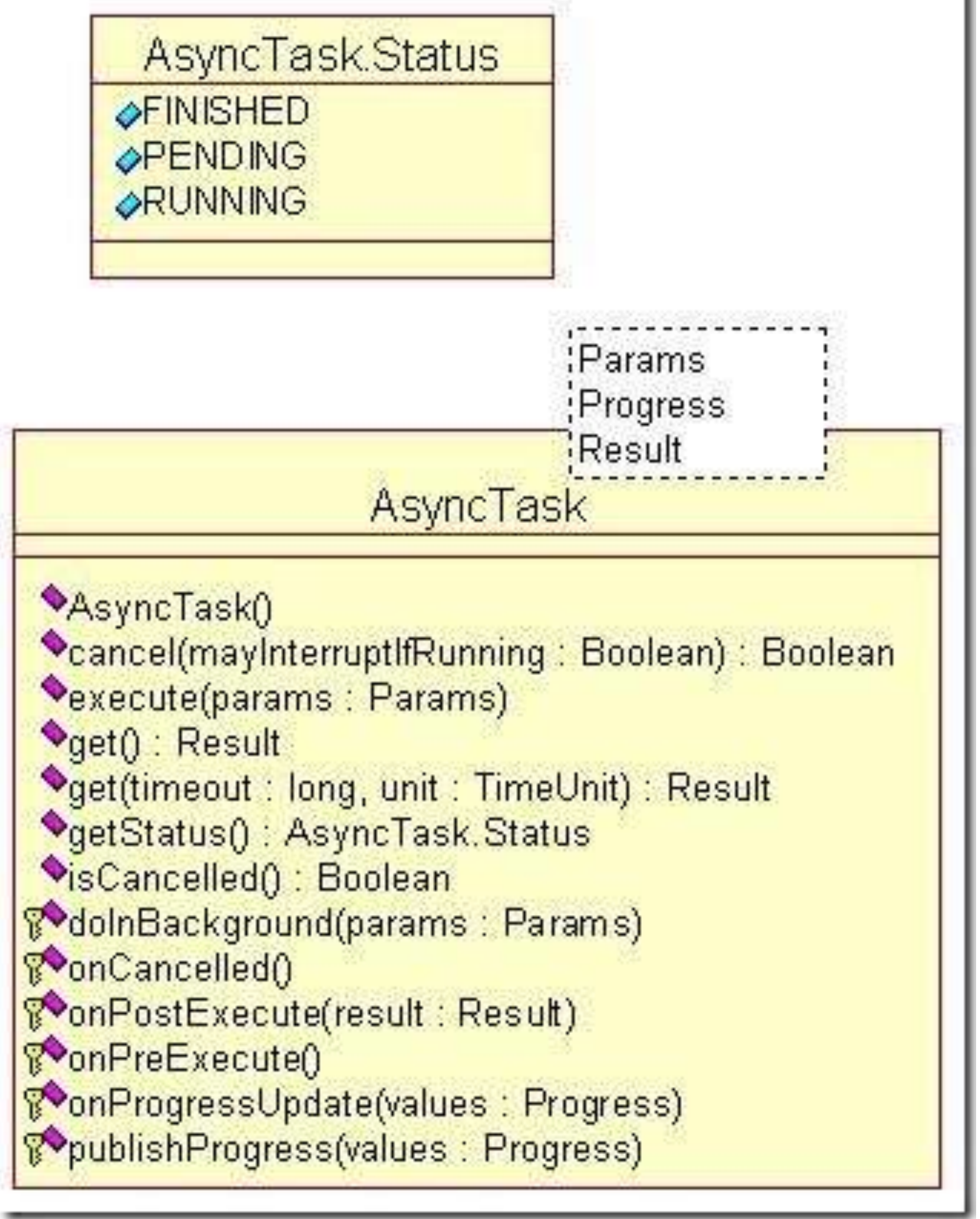
<Params, Progress, Result>


onPreExecute()
doInBackground()
...
...
publishProgress()
onProgressUpdate()
onPostExecute()

AsyncTask

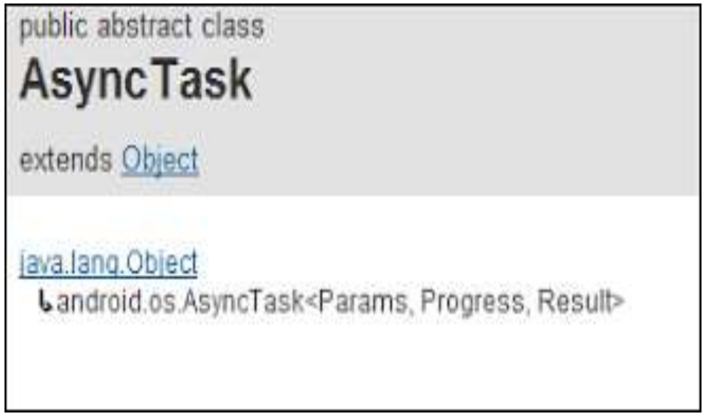
<Params, Progress, Result>


onPreExecute()
doInBackground()
...
...
publishProgress()
onProgressUpdate()
onPostExecute()


AsyncTask.Status
FINISHED
PENDING
RUNNING
Params
Progress
Result
AsyncTask
AsyncTask()
cancel(mayInterruptIfRunning : Boolean) : Boolean
execute(params : Params)
get() : Result
get(timeout : long, unit : TimeUnit) : Result
getStatus() : AsyncTask.Status
isCancelled() : Boolean
doInBackground(params : Params)
onCancelled()
onPostExecute(result : Result)
onPreExecute()
onProgressUpdate(values : Progress)
publishProgress(values : Progress)

**AsyncTask**

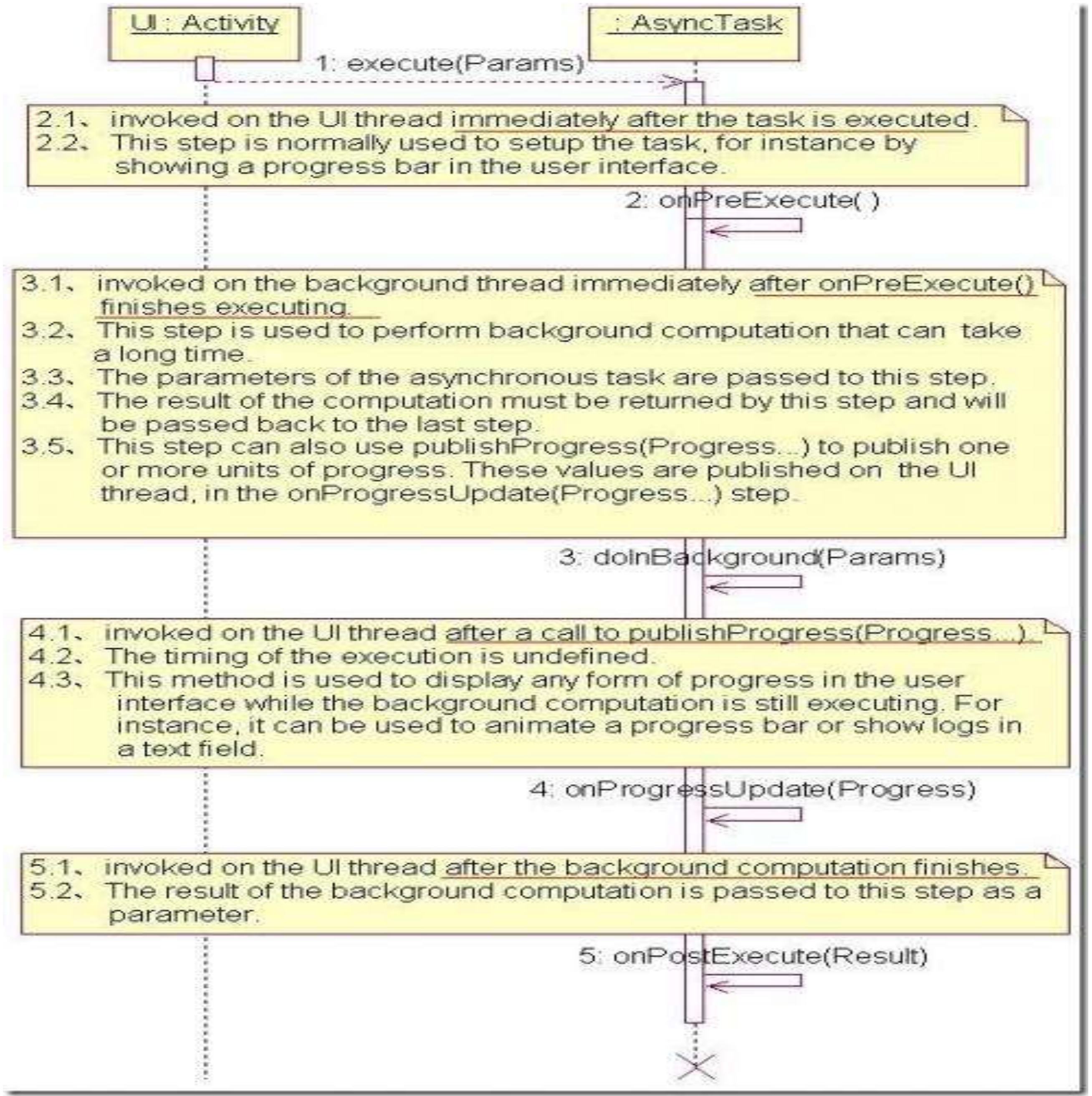
**<Params, Progress, Result>**


public abstract class

**AsyncTask**

extends Object

java.lang.Object
↳android.os.AsyncTask<Params, Progress, Result>

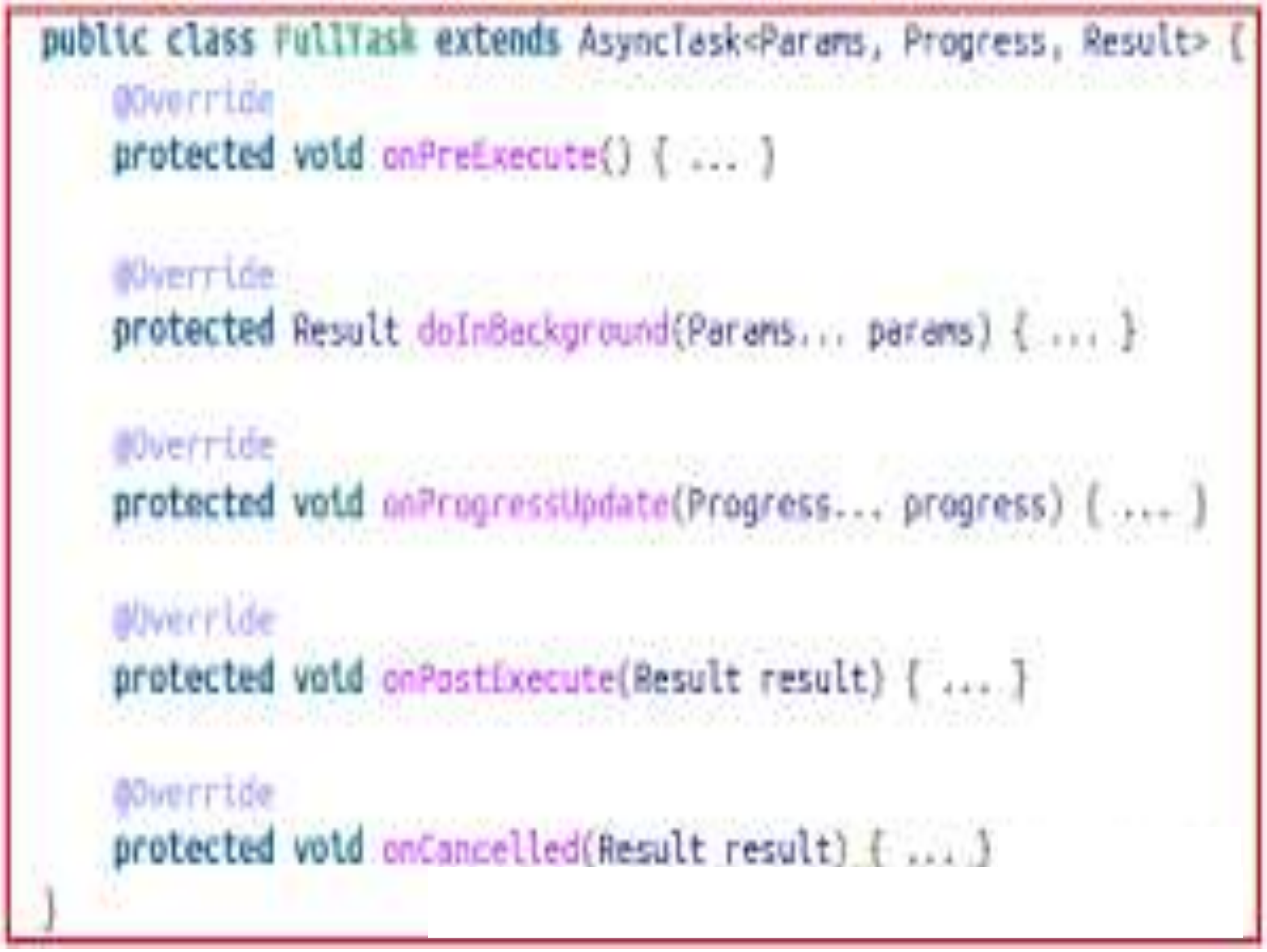


2.1、 invoked on the UI thread immediately after the task is executed.
2.2、 This step is normally used to setup the task, for instance by showing a progress bar in the user interface.

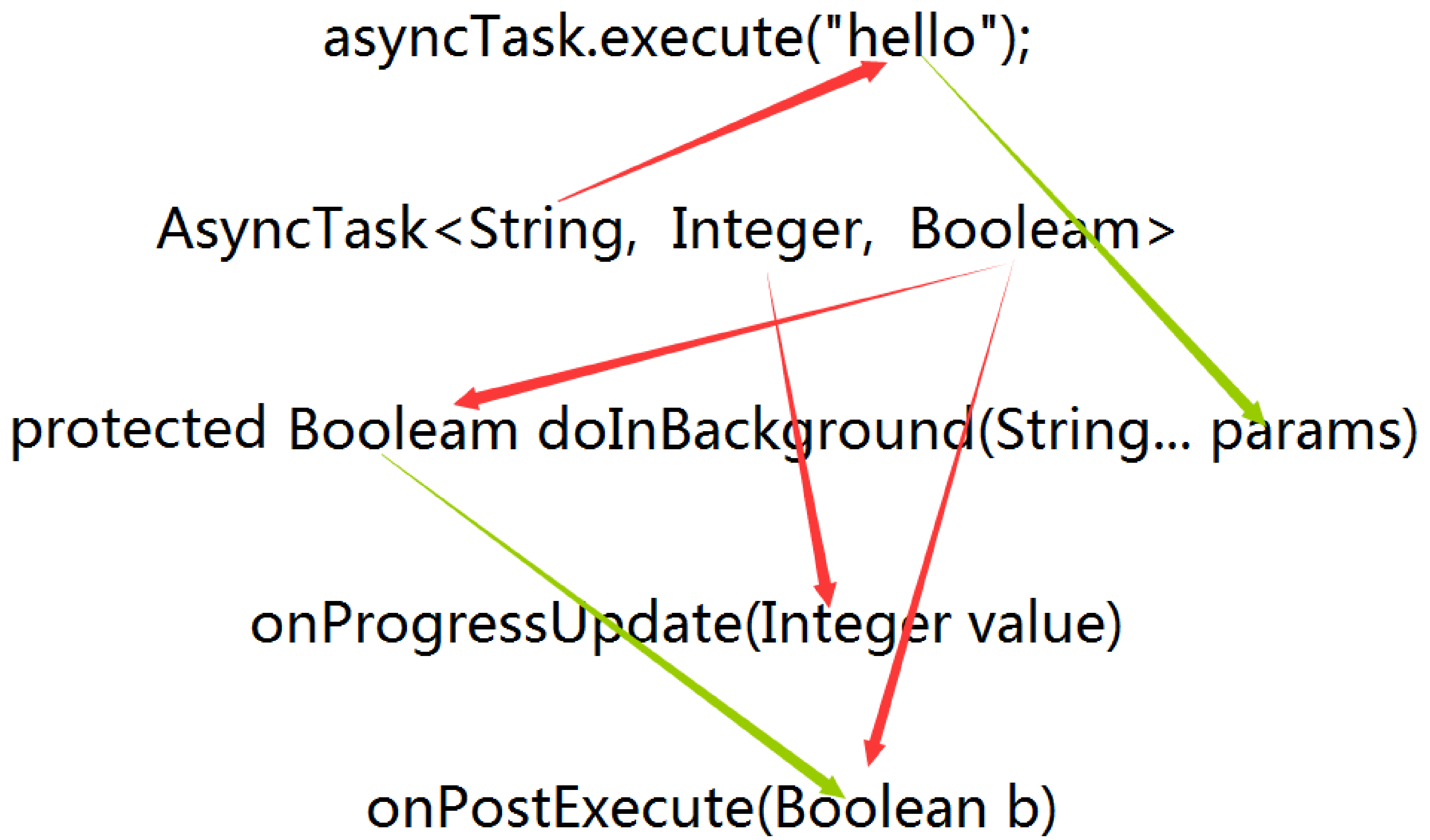
3.1、 invoked on the background thread immediately after onPreExecute() finishes executing.
3.2、 This step is used to perform background computation that can take a long time.
3.3、 The parameters of the asynchronous task are passed to this step.
3.4、 The result of the computation must be returned by this step and will be passed back to the last step.
3.5、 This step can also use publishProgress(Progress...) to publish one or more units of progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.

4.1、 invoked on the UI thread after a call to publishProgress(Progress...).
4.2、 The timing of the execution is undefined.
4.3、 This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.

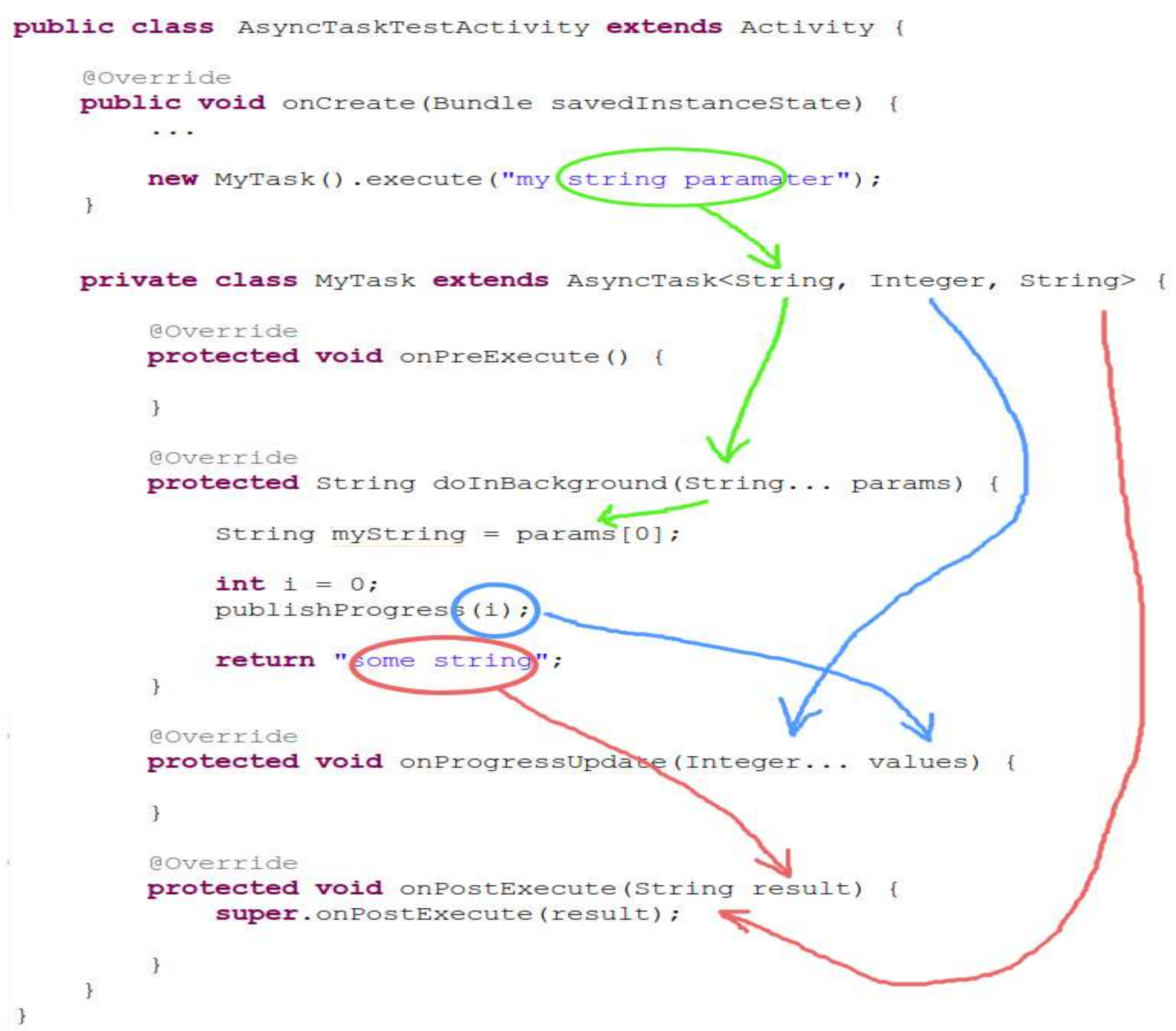
5.1、 invoked on the UI thread after the background computation finishes.
5.2、 The result of the background computation is passed to this step as a parameter.



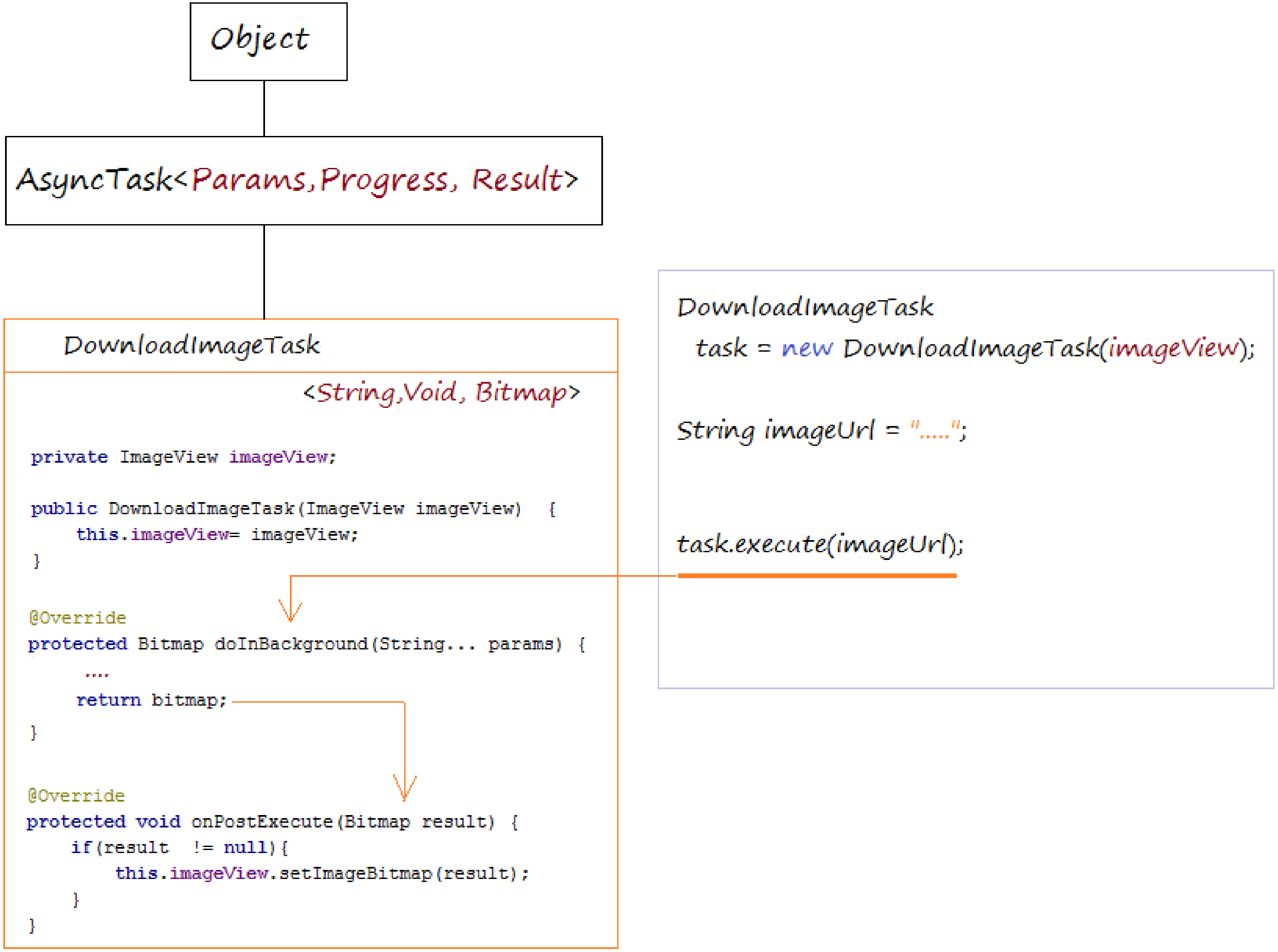
```
public class FullTask extends AsyncTask<Params, Progress, Result> {
    @Override
    protected void onPreExecute() { ... }

    @Override
    protected Result doInBackground(Params... params) { ... }

    @Override
    protected void onProgressUpdate(Progress... progress) { ... }

    @Override
    protected void onPostExecute(Result result) { ... }

    @Override
    protected void onCancelled(Result result) { ... }
}
```



```
asyncTask.execute("hello");

AsyncTask<String,  Integer,  Booleam>

protected Booleam doInBackground(String... params)

onProgressUpdate(Integer value)

onPostExecute(Boolean b)
```

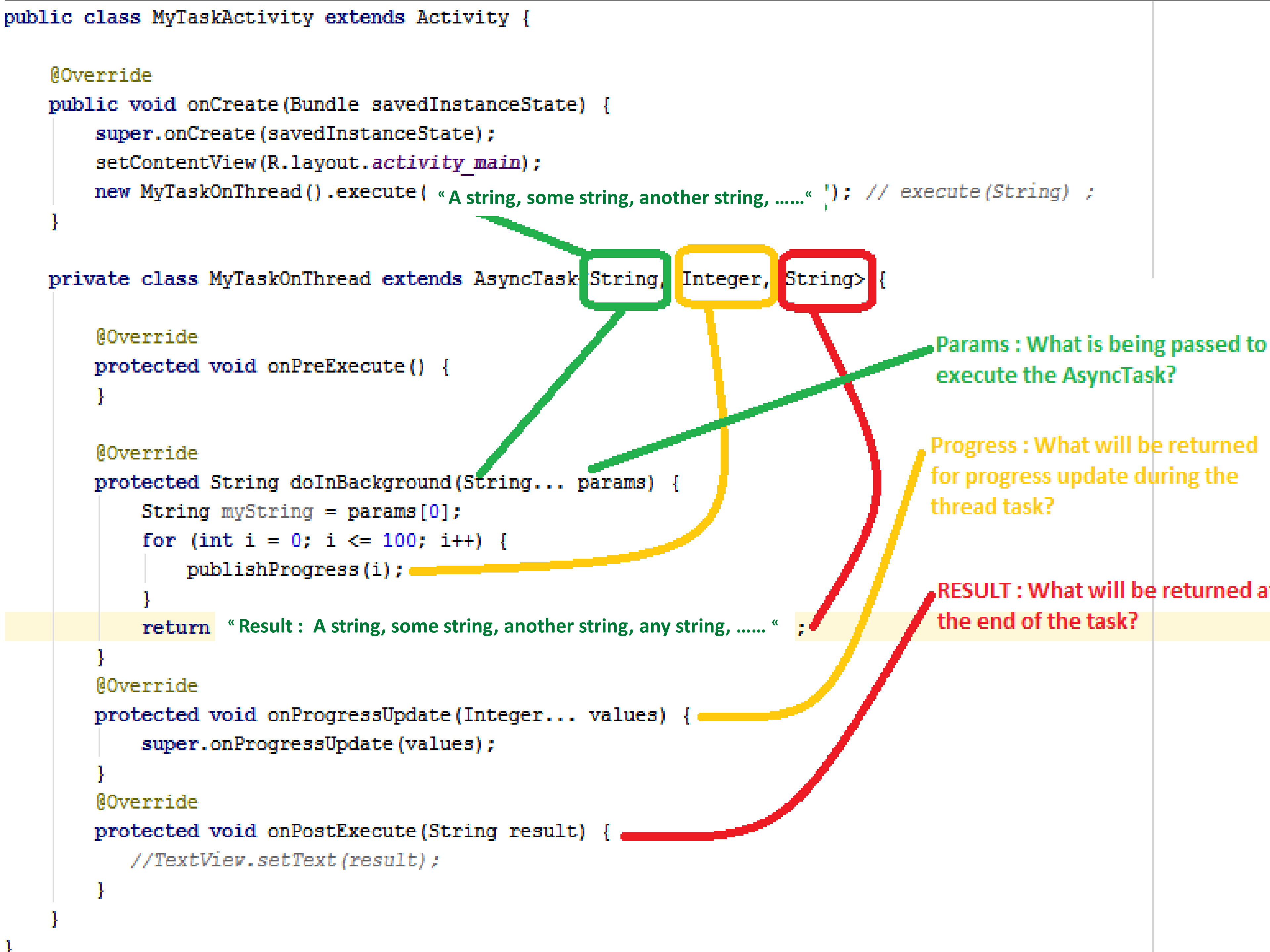


```
public class AsyncTaskTestActivity extends Activity {

    @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);

        }
    }
}
```



```
private ImageView imageView;

public DownloadImageTask(ImageView imageView)  {
    this.imageView= imageView;
}

@Override
protected Bitmap doInBackground(String... params) {
     ....
    return bitmap;
}


@Override
protected void onPostExecute(Bitmap result) {
    if(result  != null){
        this.imageView.setImageBitmap(result);
    }
}
```

```
DownloadImageTask
  task = new DownloadImageTask(imageView);

String imageUrl = ".....";


task.execute(imageUrl);
```

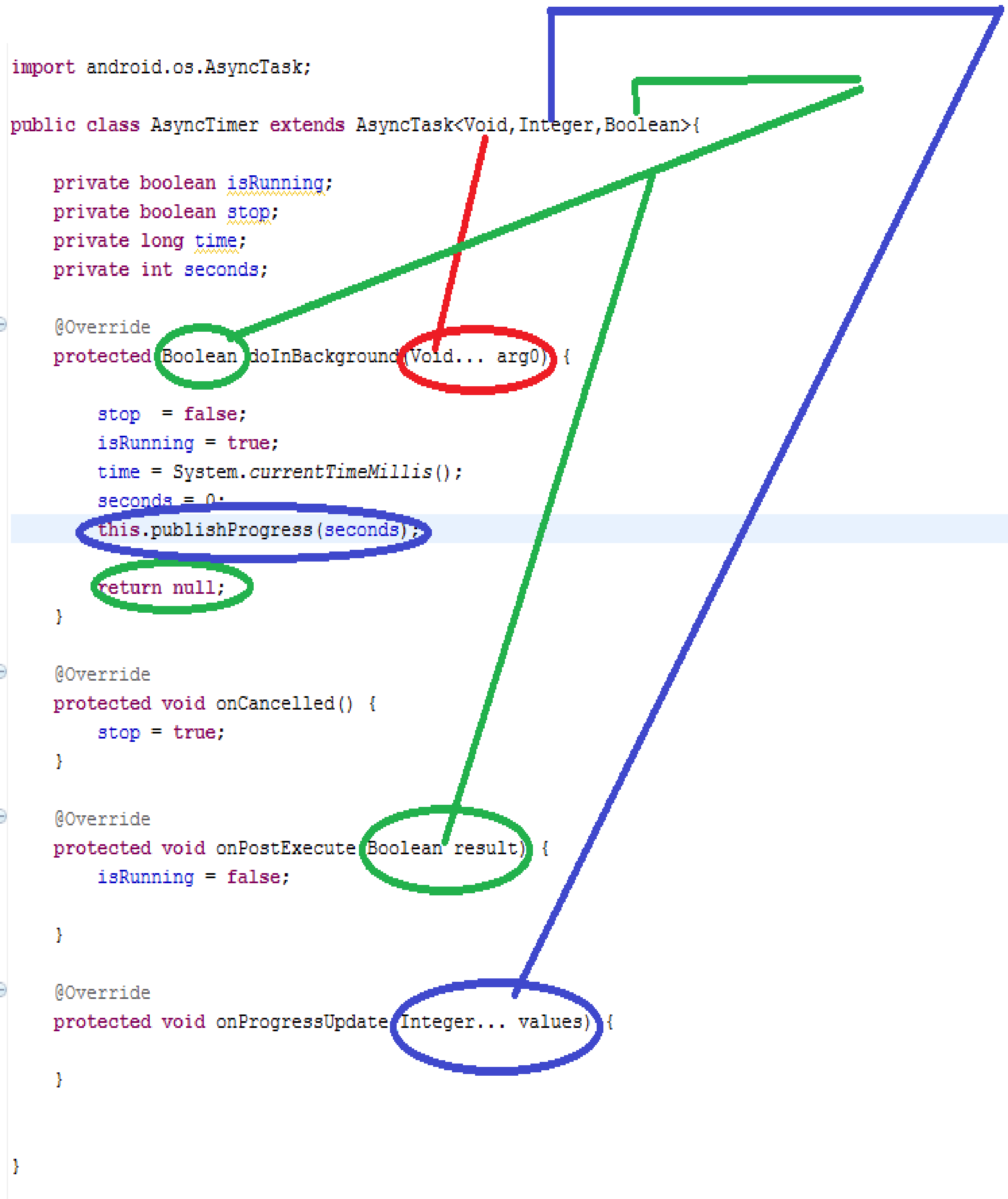


```
public class MyTaskActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        new MyTaskOnThread().execute( « A string, some string, another string, ......« '); // execute(String) ;
    }

    private class MyTaskOnThread extends AsyncTask<String, Integer, String> {

        @Override
        protected void onPreExecute() {
        }

        @Override
        protected String doInBackground(String... params) {
            String myString = params[0];
            for (int i = 0; i <= 100; i++) {
                publishProgress(i);
            }
            return « Result :  A string, some string, another string, any string, ...... « ;
        }
        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
        }
        @Override
        protected void onPostExecute(String result) {
           //TextView.setText(result);
        }
    }
}
```

Params : What is being passed to execute the AsyncTask?

Progress : What will be returned for progress update during the thread task?

RESULT : What will be returned a the end of the task?



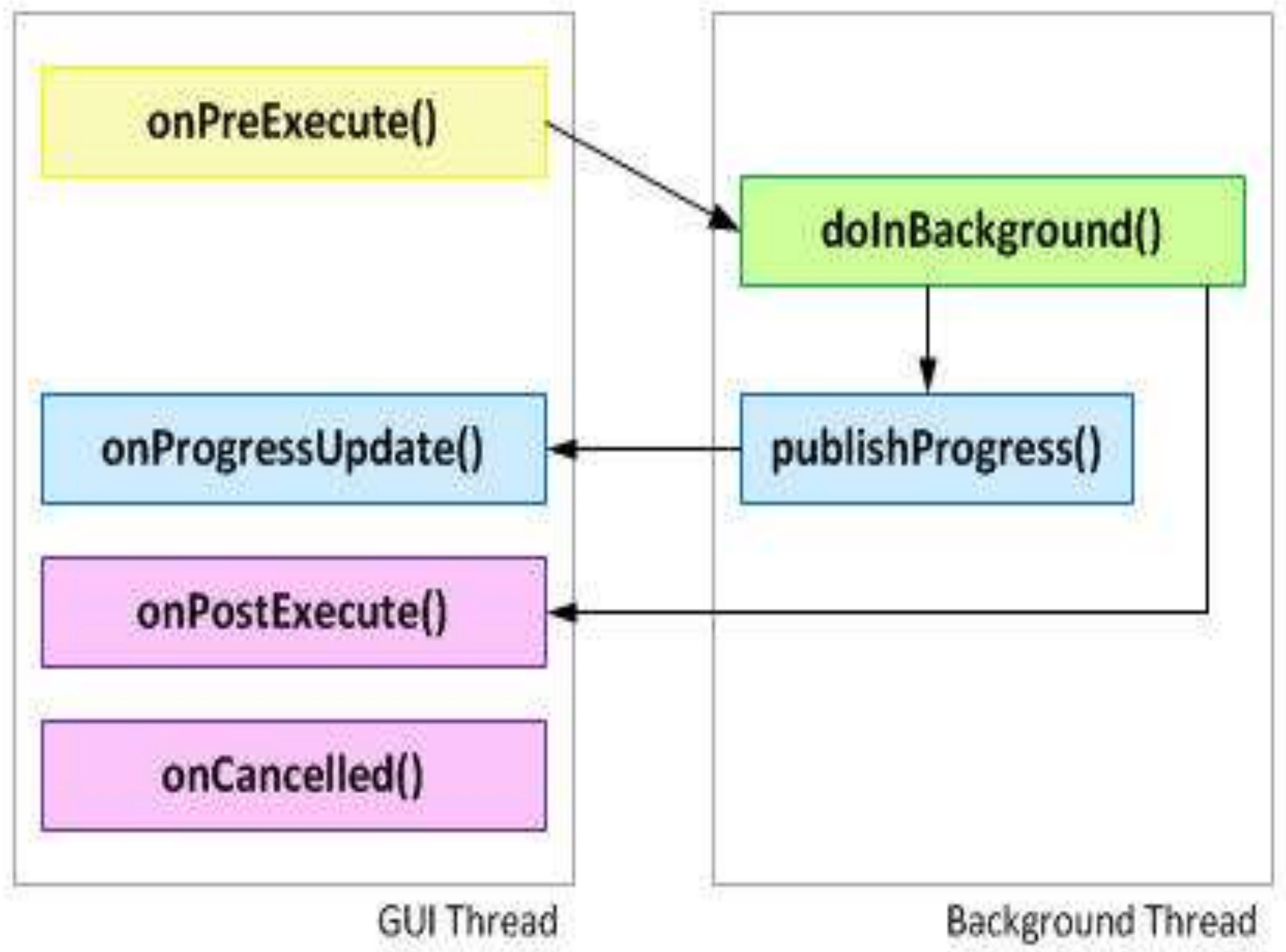
```java
import android.os.AsyncTask;

public class AsyncTimer extends AsyncTask<Void,Integer,Boolean>{

    private boolean isRunning;
    private boolean stop;
    private long time;
    private int seconds;

    @Override
    protected Boolean doInBackground(Void... arg0) {

        stop  = false;
        isRunning = true;
        time = System.currentTimeMillis();
        seconds = 0;
        this.publishProgress(seconds);

        return null;
    }

    @Override
    protected void onCancelled() {
        stop = true;
    }

    @Override
    protected void onPostExecute(Boolean result) {
        isRunning = false;

    }

    @Override
    protected void onProgressUpdate(Integer... values) {

    }


}
```

The first Type in the AsyncTask is the for the paramter of the doInBackground method!

The second type in the AsyncTask is the paramter for the onProgressUpdate method! It is also the parater type for publishProgress!

The third type in the AsyncTask is the return type for the doInBackground method and the onPostExecute method!

It is good to note that anyone of thse types can be the Void type if they have no paramters being passed to them! Also it is good to note that anyone of thse can be any class in java but they cannot be primitive types such as int, long,boolean,ect... If you wanting to pass primitive types from any of the types you must use the class representation of them like I have done for Integer for the onProgressUpdate!


onPreExecute()
doInBackground()
onProgressUpdate()
publishProgress()
onPostExecute()
onCancelled()
GUI Thread
Background Thread

**Exemple:**

**Use an AsyncTask to download an image. The download task starts on clicking on a button.**

**Another button will be used to display a string in concurrency with the download task.**

**The UI should stay responsive to either  button has been clicked.**

# End of Lecture

Université Mohamed V
***Faculté des sciences***
Département d'informatique

***2017/2018***

Module:
**Mobile and cloud computing**

# Asynctasks

***Pr. Oussama REDA***