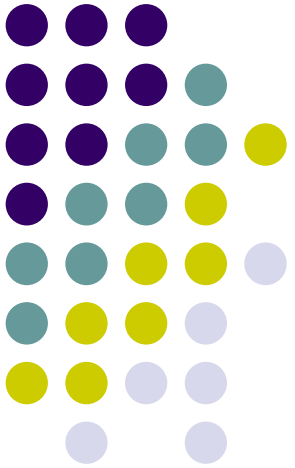
Τεχνολογία και Προγραμματισμός Κινητών Συσκευών

5a



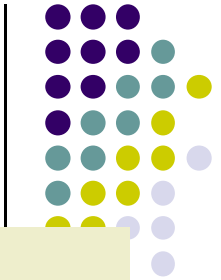
Νήματα - Χειριστές

Ιωάννης Έλληνας

Νήματα και Χειριστές (Threads & Handlers)

- Το κύριο νήμα είναι υπεύθυνο για την ενημέρωση της οθόνης. Οι χρονοβόρες διαδικασίες πρέπει να εκτελούνται σε νέο νήμα.
- Τα νέα νήματα δεν ενημερώνουν την οθόνη.
- Οι μεταβολές στο νήμα UI γίνονται στο ίδιο νήμα.
- Οι χειριστές βρίσκονται στο κύριο νήμα και επικοινωνούν με τα άλλα νήματα μέσω μηνυμάτων.
- Η έλλειψη συγχρονισμού με τα μηνύματα λύνεται με την κλάση AsyncTask.

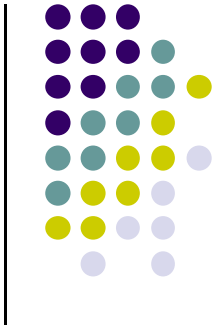
Thread and Runnable



- **Runnables** are **NOT** Threads
- Η Runnable interface χρησιμοποιείται για τον ορισμό μιας συγκεκριμένης εργασίας που θέλουμε να εκτελέσουμε και υλοποιείται στη μοναδική της μέθοδο, run () (χωρίς παραμέτρους).
- Η κλάση Runnable δεν δημιουργεί ξεχωριστό νήμα, γιατί αυτό ακριβώς κάνει η κλάση Thread.

- Η κλάση **Thread** χρησιμοποιείται για να δημιουργήσει ένα νέο νήμα εκτέλεσης από το κύριο πρόγραμμα.
- Σε ένα νέο αντικείμενο Thread, παρέχοντας ένα Runnable ως όρισμα σε έναν κατασκευαστή Thread, δίνεται ουσιαστικά πρόσβαση στην εργασία του Runnable (που ορίζεται στη μέθοδο run ()).
- Σε οποιαδήποτε στιγμή κατά τη διάρκεια του προγράμματος μπορείτε να ξεκινήσετε το νέο νήμα, χρησιμοποιώντας το Thread.start () και ο κώδικας Runnable θα ξεκινήσει να τρέχει.

Δημιουργία νήματος



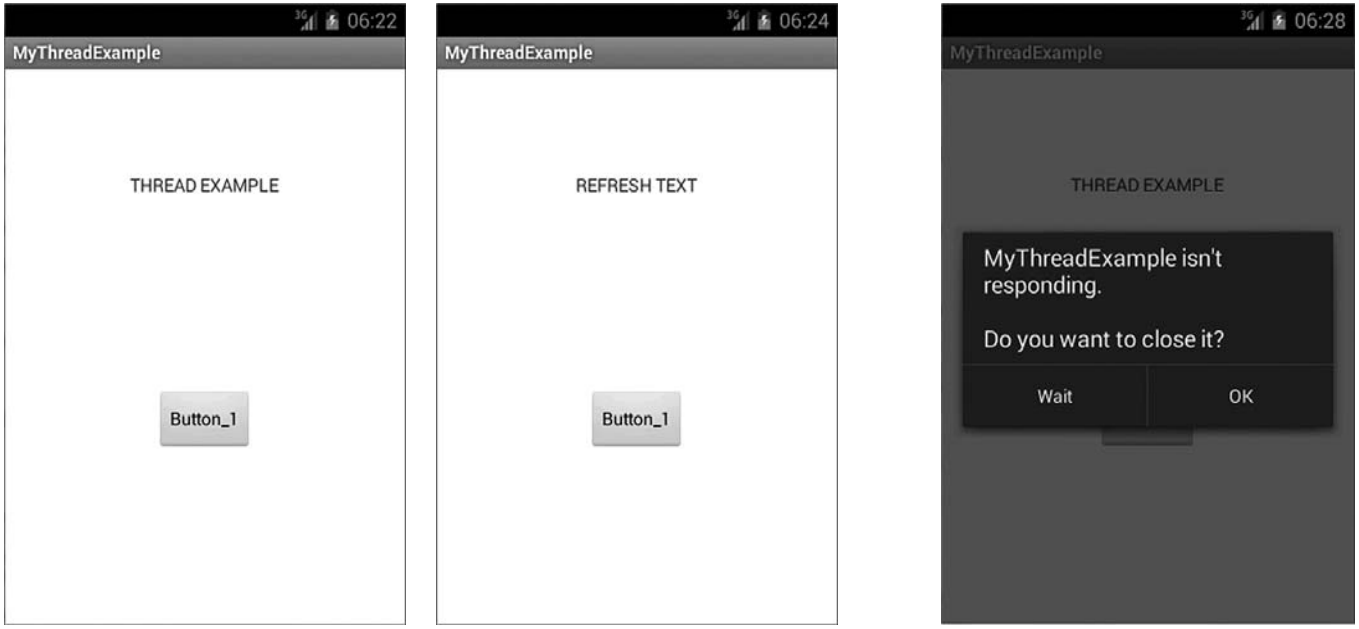
```
// Method called on UI thread
private void myThread() {
  Thread thread = new Thread(threadProcess);
  thread.start();
}

// Runnable public interface – background process
private Runnable threadProcess = new Runnable() {
  public void run() {
    threadJob();
  }
};

// Time consuming process
private void threadJob() {
  //TODO time consuming operations
}
```

ή

```
Runnable threadProcess = new Runnable() {
  public void run() {
    //TODO time consuming operations
  }
};
Thread thread = new Thread(threadProcess);
thread.start();
```

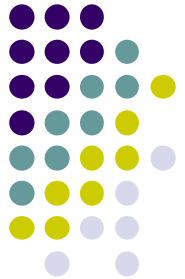
Παράδειγμα *MyThreadExample*

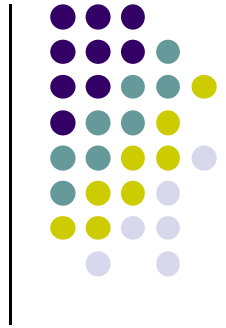
Εκτέλεση προγράμματος χωρίς νέο νήμα



Εναλλαγή δύο λέξεων στο πεδίο κειμένου κάθε 10 sec

```
package gr.mybook.mythreadexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class Activity_1 extends Activity {
  Boolean flipText = false;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_1);
  }

  public void  onClick(View view) {
    flipText = !flipText;
    TextView textView = (TextView) findViewById( R.id.textView1);
    if (flipText) {
      delay();
      textView.setText("REFRESH TEXT");
    }
    else {
      delay();
      textView.setText(getString(R.string.text1));
    }
  }
```

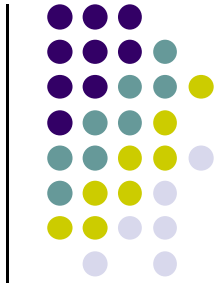




```
  public void delay() {
    long upto = System.currentTimeMillis() + 10000;
    while (System.currentTimeMillis()<upto) {
      try {
        wait(upto-System.currentTimeMillis());
      }
      catch (Exception e) {
        // end thread
      }
    }
  }
}
```

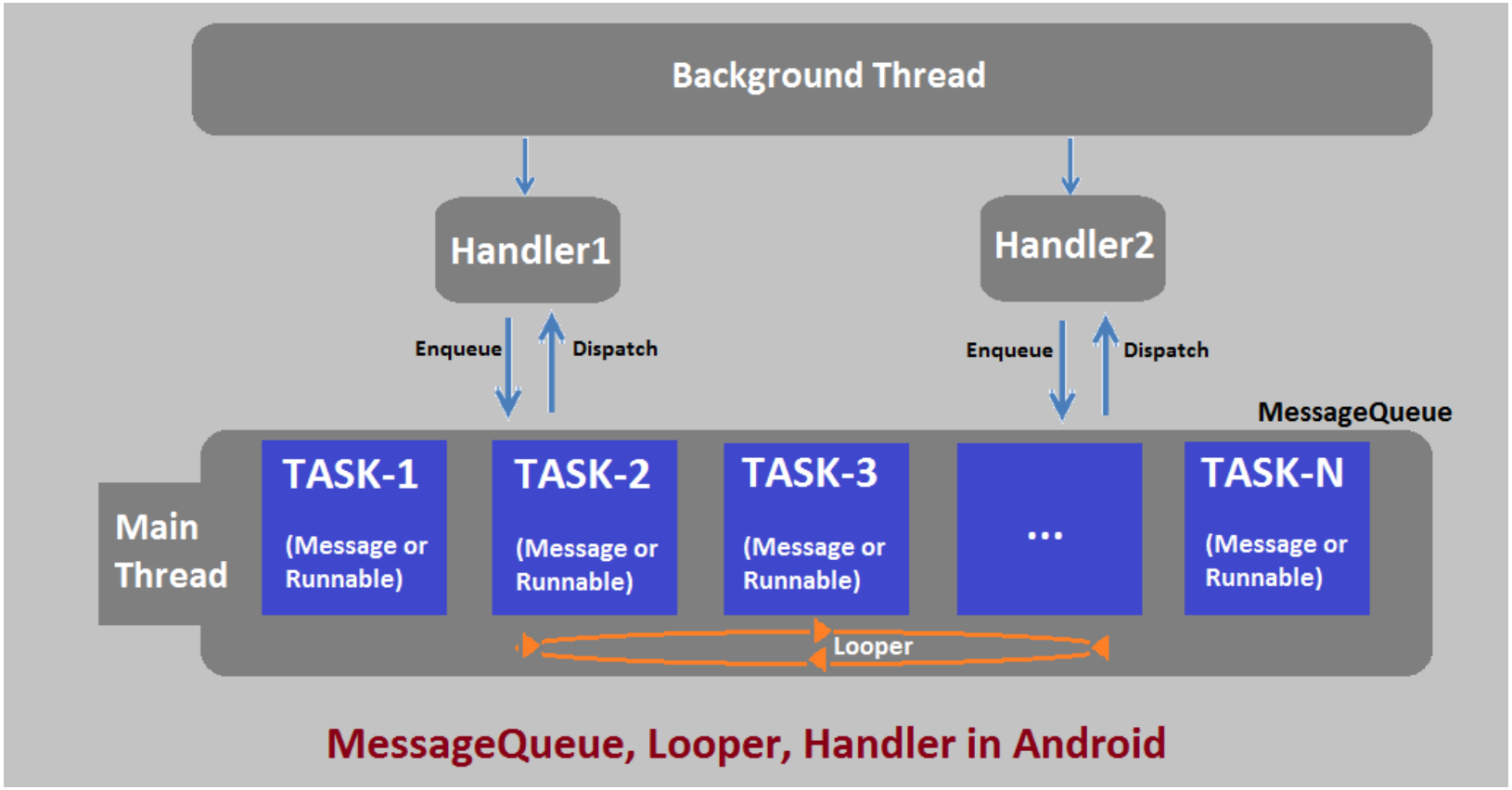
Δημιουργία προβλήματος στην εκτέλεση λόγω της χρονοκαθυστέρησης

ΝΕΟ ΝΗΜΑ

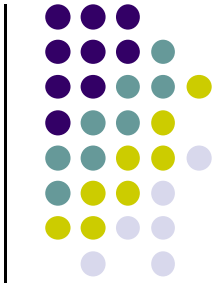


```
    ……..……
  public void delay() {
    Runnable runnable = new Runnable() {
      public void run() {
        long upto= System.currentTimeMillis()+20000;
        while (System.currentTimeMillis()<upto) {
          try {
            wait(upto-System.currentTimeMillis());
          }
          catch (Exception e) {
            // end thread
          }
        }
      }
    };
    Thread thread = new Thread(runnable);
    thread.start();
  }
}
```

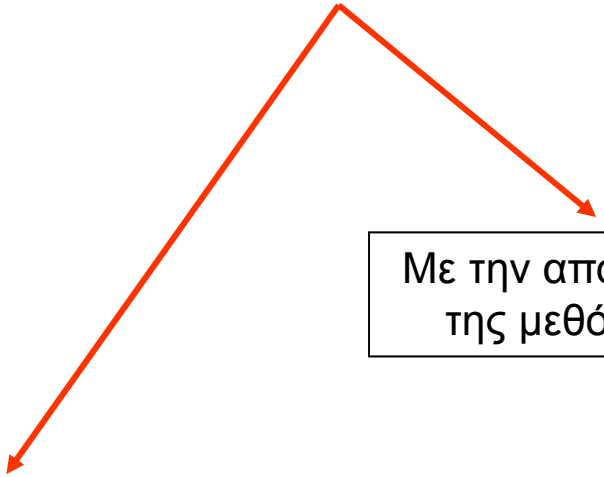
Threads and Handlers



Χειριστές (Handlers)



- Εξασφαλίζουν την επικοινωνία κυρίου νήματος και άλλων νημάτων.

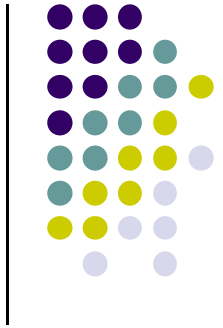


Με την αποστολή ενός αντικειμένου *Runnable*, μέσω της μεθόδου ***post(),*** στην ουρά των μηνυμάτων.

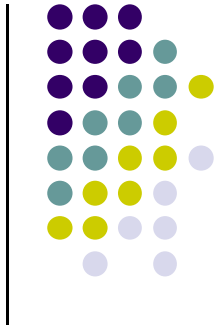
Με την αποστολή μηνύματος μέσω της μεθόδου *sendMessage()*, η οποία τοποθετεί ένα αντικείμενο μηνύματος στην ουρά των μηνυμάτων και το οποίο επεξεργάζεται η μέθοδος *handleMessage()* στο κύριο νήμα.

```
……
public class Activity_1 extends Activity {
  // new handler
  Handler handler = new Handler() {
    // Message reception and handling
    @Override
    public void handleMessage(Message msg) {
      // TO DO -- get the bundle and extract data by key
      ……
    }
  };

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_1);
    ……
  }

  @Override
  protected void onStart() {
    super.onStart();
    // create a runnable object
    Runnable runnable = new Runnable() {
      public void run() {
      ……
        // new message object
        Message msg=handler.obtainMessage();
        // or new message object
        //Message msg = new Message();
        // creation of a new data bundle
        Bundle b = new Bundle();
        b.putString("My Key", "My Value”);
        msg.setData(b);
        // send message to the handler
        handler.sendMessage(msg);
      }
    };
    Thread thread = new Thread(runnable);
    thread.start();
  }
}
```

Επικοινωνία με sendMessage() και handleMessage()



Εναλλακτικός τρόπος επικοινωνίας με αντικείμενο runnable και post()



```
……
public class Activity_1 extends Activity {
  // new handler
    Handler handler = new Handler();
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_1);
    ……
  }

  @Override
  protected void onStart() {
    super.onStart();
    // create a runnable object
    Runnable runnable = new Runnable() {
      public void run() {
        // TODO changes in UI
        ……
      }
    };
```

```
    // send Runnable to the handler
    handler.post(runnable);
    Thread thread = new Thread(runnable);
    thread.start();
  }
}
```

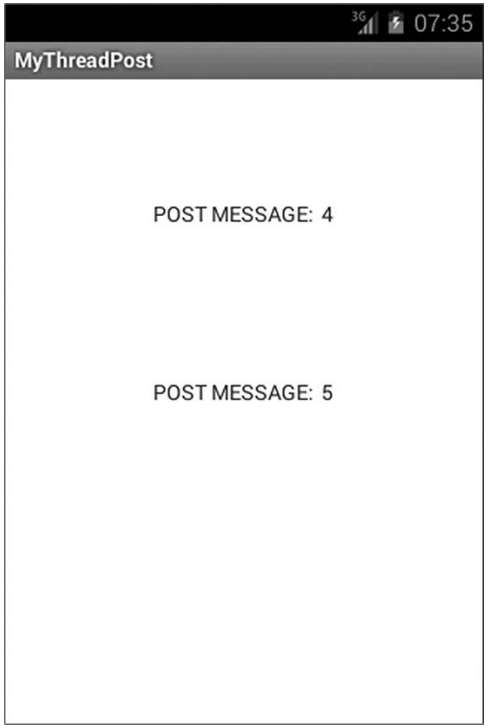
Παράδειγμα MyThreadMessage

Δημιουργία νέου νήματος και εμφάνιση στην οθόνη του THREAD MESSAGE κάθε δευτερόλεπτο μέσω του χειριστή και των μεθόδων *sendMessage()* και *handleMessage()*.

```
package gr.mybook.mythreadmessage;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.TextView;

public class Activity_1 extends Activity {
  TextView text;
  //new handler
  Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
      // get the bundle and extract data by key
      Bundle b = msg.getData();
      String key = b.getString("My Key");
      text.setText(key);
    }
  };

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_1);
    text = (TextView) findViewById(R.id.textView1);
  }
```

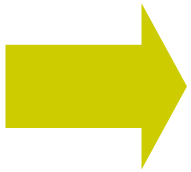
```
  @Override
  protected void onStart() {
    super.onStart();

    // create a new thread
    Runnable runnable = new Runnable() {
      public void run() {
        for (int i = 0; i < 10; i++) {
          try {
            Thread.sleep(1000);
            Message msg = handler.obtainMessage();
            // OR Message msg = new Message();
            Bundle b = new Bundle();
             b.putString("My Key", "THREAD MESSAGE:  " + String.valueOf(i));
            msg.setData(b);
            // send message to the handler with the current message handler
            handler.sendMessage(msg);
          }
          catch (InterruptedException e) {
            e.printStackTrace();
          }
        }
      }
    };
     Thread thread = new Thread(runnable);
     thread.start();
  }
}
```

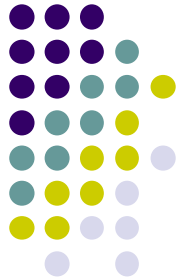
Παράδειγμα MyThreadPost



Νέο νήμα όπου εμφανίζει κάθε δευτερόλεπτο αρίθμηση από 1-9 και ταυτόχρονα από 9-1, με τη μέθοδο post().

```
package gr.mybook.mythreadpost;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.TextView;
public class Activity_1 extends Activity {
  TextView text1, text2;
  //new handler
  Handler handler = new Handler();
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_1);
    text1 = (TextView) findViewById(R.id.textView1);
    text2 = (TextView) findViewById(R.id.textView2);
  }
  @Override
  protected void onStart() {
    super.onStart();
    final int[] i = new int[1];
    // new thread
    new Thread() {
      public void run() {
        for (i[0] = 0; i[0] < 10; i[0]++) {
          try {
            Thread.sleep(1000);
```

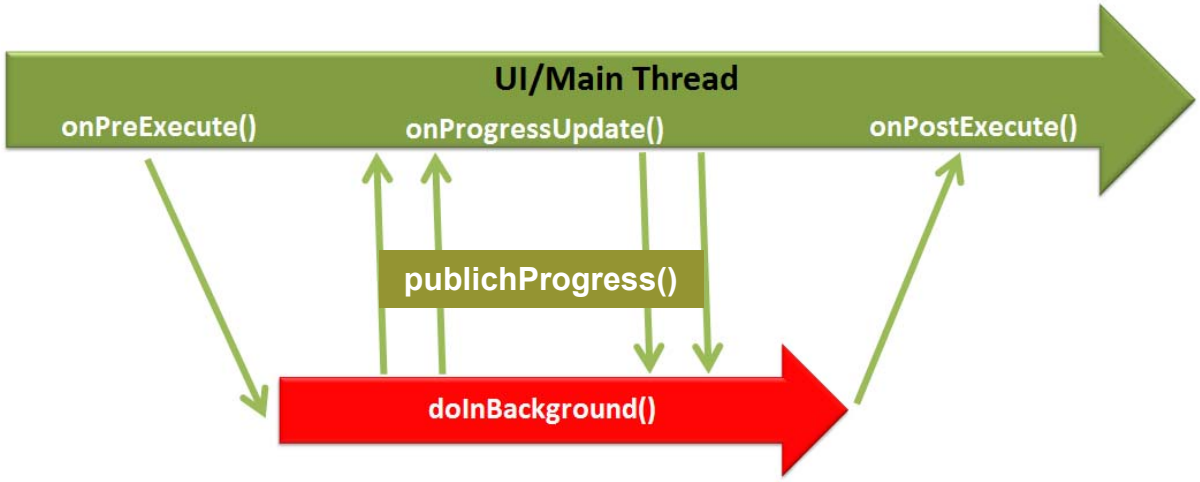




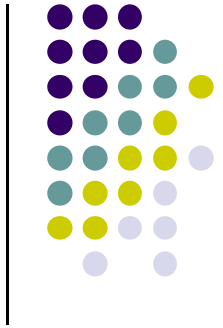
```
            handler.post(new Runnable() {
              public void run() {
                text1.setText("POST MESSAGE:  " + String.valueOf(i[0]));
              }
            });
            handler.post(new Runnable() {
              public void run() {
                text2.setText("POST MESSAGE:  " + String.valueOf(9-i[0]));
              }
            });
          }
          catch (InterruptedException e) {
            e.printStackTrace();
          }
        }
      }
    }.start();
  }
}
```

Η κλάση AsyncTask

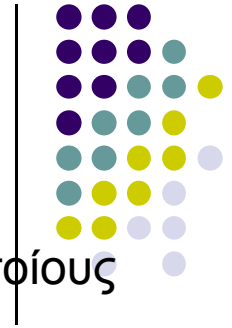
- Αποφυγή προβλημάτων συγχρονισμού ενημέρωσης του UI.
- Η κλάση AsyncTask διαχειρίζεται υπηρεσίες παρασκηνίου που ενημερώνουν το κύριο νήμα.
- Αντί νημάτων και χειριστών δημιουργούμε μια υποκλάση της AsyncTask με τις μεθόδους επανάκλησης.



AsyncTask in Android



```
……
……
public class mainActivity extends Activity {
……
……
  //** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ……
    ……
    new myAsyncTask().execute(x);
  }
  private class myAsyncTask extends AsyncTask<params, progress, result> {
    ……
    ……
    @Override
    protected void onPreExecute() {
    ……
    ……
    }
    @Override
    protected res doInBackground(params…x){
      ……
      ……
      publishProgress(var);
      return(res);
    }
    @Override
     protected void onProgressUpdate(progress…var) {
       ……
       ……
     }
     @Override
     protected void onPostExecute(result res) {
       ……
       ……
     }
  }
}
```



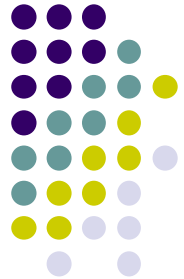
Η κλάση *AsyncTask* χρησιμοποιεί τρεις τύπους παραμέτρων, από τους οποίους οι δύο πρώτοι είναι πίνακες αντικειμένων:

- ***Params*** - Ο πρώτος τύπος παραμέτρων της *AsyncTask* χαρακτηρίζει τον πίνακα αντικειμένων που μεταφέρονται στη μέθοδο *doInBackground()* για την εκτέλεση της εργασίας παρασκηνίου.

- ***Progress*** - Ο δεύτερος τύπος παραμέτρων της *AsyncTask* χαρακτηρίζει τον πίνακα αντικειμένων που μεταφέρονται από τη μέθοδο *doInBackground()* στη μέθοδο *onProgressUpdate()* μέσω της μεθόδου *publishProgress()*, για να ενημερωθεί το κύριο νήμα για την πρόοδο της εργασίας παρασκηνίου.
  Το αποτέλεσμα μεταφέρεται στη μέθοδο *postExecute(),* η οποία αποτελεί την έξοδο από την εργασία παρασκηνίου.

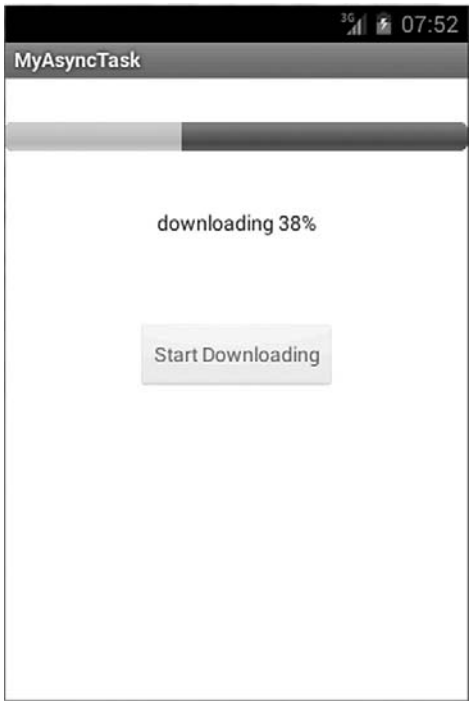
- ***Result*** - Ο τρίτος τύπος παραμέτρων της *AsyncTask* χαρακτηρίζει το αποτέλεσμα μετά την ολοκλήρωση της εργασίας παρασκηνίου, δηλαδή στο τέλος της μεθόδου *doInBackground().*

Εάν κάποιος τύπος δεν χρησιμοποιείται, τότε στην αντίστοιχη θέση τοποθετείται το Void.



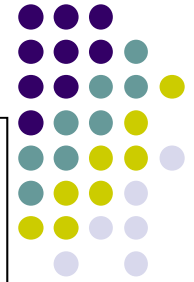
- ***onPreExecute()*** – Η μέθοδος αυτή εκτελείται από το κύριο νήμα πριν αρχίσει η λειτουργία παρασκηνίου. Χρησιμοποιείται για την αρχικοποίηση μεγεθών που θα χρησιμοποιηθούν στην εργασία παρασκηνίου (π.χ. την αρχικοποίηση μιας μπάρας προόδου).
- ***doInBackground()*** - Η μέθοδος αυτή εκτελείται στο παρασκήνιο και εκεί τοποθετείται όλη η εργασία που πρόκειται να γίνει. Δέχεται τον πρώτο τύπο παραμέτρων της *AsyncTask*, επί των οποίων θα λειτουργήσει (π.χ. ένα URL για πρόσβαση στο διαδίκτυο). Η μέθοδος αυτή επιστρέφει μια παράμετρο του τρίτου τύπου, η οποία μεταφέρεται στη μέθοδο *onPostExecute()* για έξοδο από την εργασία παρασκηνίου.
- ***publishProgress()*** – Η μέθοδος αυτή, η οποία καλείται από την προηγούμενη, ενημερώνει περιοδικά το κύριο νήμα. Οι παράμετροι αυτής της μεθόδου μεταφέρονται στην *onProgressUpdate()*.
- ***onProgressUpdate()*** - Η μέθοδος αυτή ενημερώνει το κύριο νήμα όποτε η *doInBackground()* καλεί την *publishProgress()*. Η μέθοδος αυτή λαμβάνει πληροφορίες από την εργασία παρασκηνίου (π.χ. αύξηση της μπάρας προόδου) του δεύτερου τύπου παραμέτρων.
- ***onPostExecute()*** – Η μέθοδος αυτή εκτελείται από το κύριο νήμα όταν ολοκληρωθεί η εργασία παρασκηνίου. Λαμβάνει μια παράμετρο μόλις ολοκληρωθεί η μέθοδος *doInBackground()* (π.χ. επιτυχής/ανεπιτυχής εργασία) και πραγματοποιεί την έξοδο από το παρασκήνιο (π.χ. εξαφάνιση της μπάρας προόδου).

Παράδειγμα MyAsyncTask

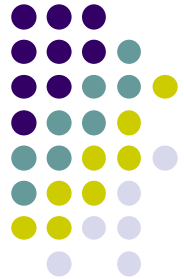



```
public class Activity_1 extends Activity {
    Button btn;
    TextView text;
    ProgressBar bar;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_1);
        btn = (Button) findViewById(R.id.button1);
        bar = (ProgressBar) findViewById(R.id.progress);
        text = (TextView) findViewById(R.id.textView1);
        btn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
              btn.setEnabled(false);
              new myAsyncTask().execute();
          }
        });
     }
```

```
private class myAsyncTask extends AsyncTask<Void, Integer, Void> {
    int progress_status;
    @Override
    protected void onPreExecute() {
        progress_status = 0;
        text.setText("downloading 0%");
    }
    @Override
    protected Void doInBackground(Void...unused){
     while(progress_status<100){
        progress_status += 2;
        publishProgress(progress_status);
        SystemClock.sleep(300);
     }
     return(null);
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        bar.setProgress(values[0]);
        text.setText("downloading " +values[0]+"%");
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
        text.setText("download complete");
        btn.setEnabled(true);
    }
  }
}
```



Εργασία



- Δημιουργείστε το project MyThreadHandler στο οποίο θα εμφανίζεται ένα πεδίο κειμένου (θα γράφει “MY PROJECT”) και ένα πλήκτρο (θα γράφει “DOWNLOAD”).
- Όταν πατηθεί το πλήκτρο θα δημιουργείται ένα νέο νήμα στο οποίο θα πραγματοποιείται χρονοκαθυστέρηση 20 δευτερολέπτων.
- Στο τέλος του χρόνου, το πεδίο κειμένου θα εμφανίσει το μήνυμα “END OF DELAY” μέσω ενός χειριστή μηνυμάτων.