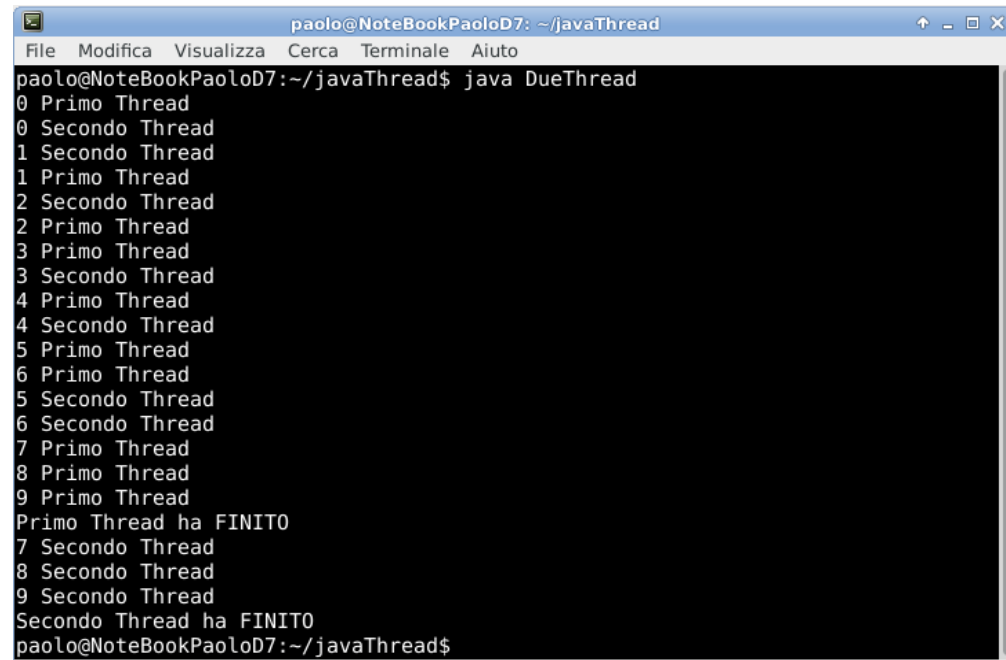


Un esempio banale di Thread in Java preso da internet:

```
class DueThread {  
    public static void main (String args[]) {  
        new SimpleThread("Primo Thread").start();  
        new SimpleThread("Secondo Thread").start();  
    }  
}
```

```
class SimpleThread extends Thread {  
    public SimpleThread(String str) {  
        super(str);  
    }  
    public void run() {  
        for (int i = 0; i < 10; i++) {  
            System.out.println(i + " " + getName());  
            try {  
                sleep((int)(Math.random() * 1000));  
            } catch (InterruptedException e) {}  
        }  
        System.out.println( getName()+ " ha FINITO");  
    }  
}
```

L'esecuzione produce :



```
paolo@NoteBookPaoloD7: ~/javaThread
File Modifica Visualizza Cerca Terminale Aiuto
paolo@NoteBookPaoloD7:~/javaThread$ java DueThread
0 Primo Thread
0 Secondo Thread
1 Secondo Thread
1 Primo Thread
2 Secondo Thread
2 Primo Thread
3 Primo Thread
3 Secondo Thread
4 Primo Thread
4 Secondo Thread
5 Primo Thread
6 Primo Thread
5 Secondo Thread
6 Secondo Thread
7 Primo Thread
8 Primo Thread
9 Primo Thread
Primo Thread ha FINITO
7 Secondo Thread
8 Secondo Thread
9 Secondo Thread
Secondo Thread ha FINITO
paolo@NoteBookPaoloD7:~/javaThread$
```

“IMPORTIAMOLO “ in ANDROID

*file : MainActivity.java*

```
package com.example.appthread00;

import android.os.Bundle;
...

public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        new SimpleThread("primoTask").start();
        new SimpleThread("secondoTask").start();
    }
    .....
}
```

*file : SimpleThread.java (è identico!!)*

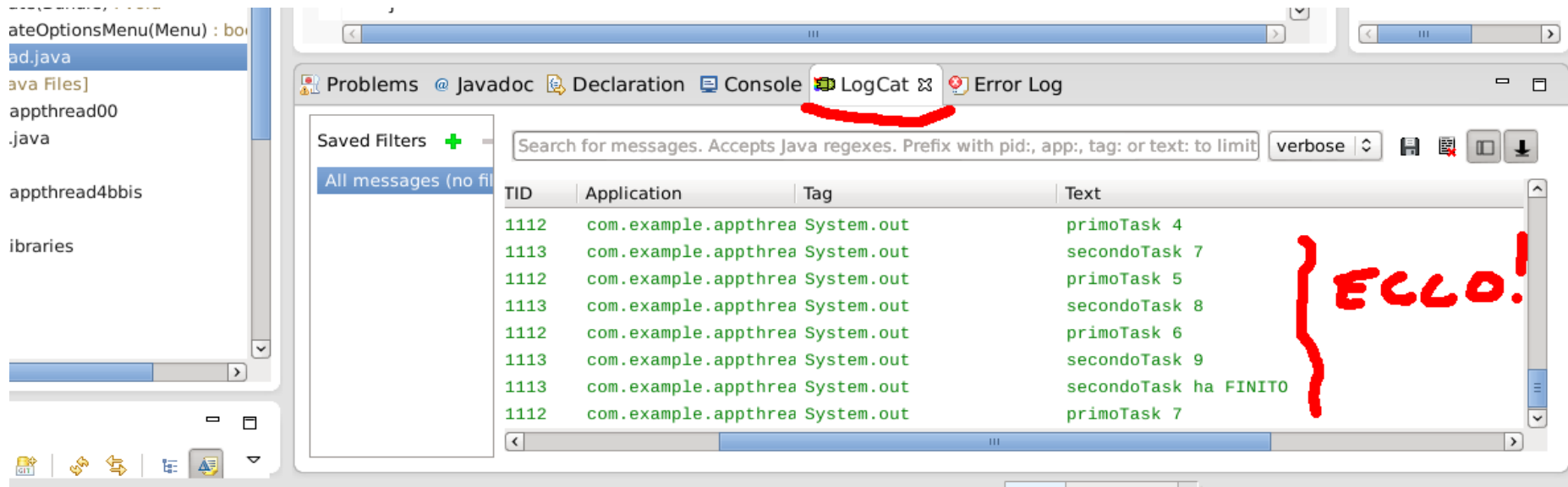
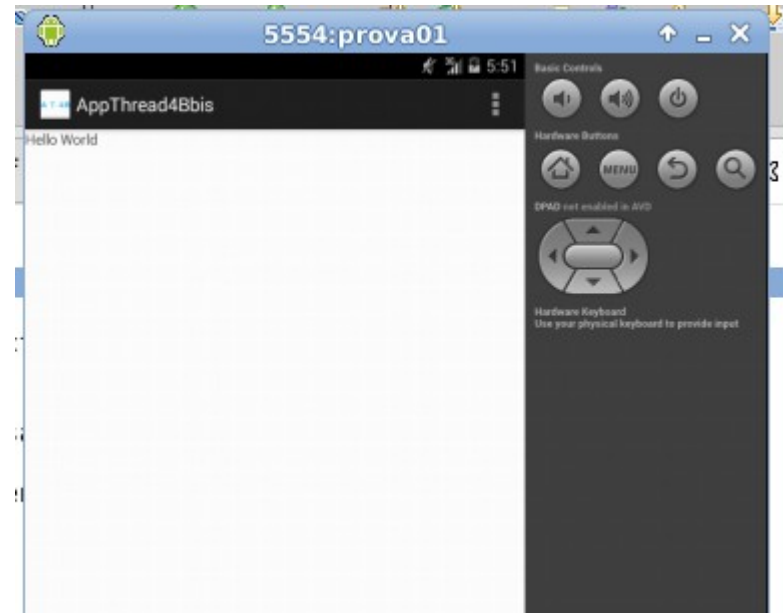
```
class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((int)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println( getName()+ " ha FINITO");
    }
}
```

l'esecuzione produce :  
HelloWorld!?

(forse il prof. Genera sempre Hello Word dal wizard!)

E i Thread dove hanno prodotto gli output?  
Dove è `System.out.println` ?

Uno sguardo all'ADT :



System.out.println va a finire qui! LogCat !  
E pre visualizzarlo su un dispositivo reale?

MA NON E' questo che vogliamo.

Noi vorremo che i Thread scrivessero sulla UI. Magari al posto di "hello World";

Modifichiamo quindi MainActivity.java :

```
/**
 * @author paolo
 *
 * @version 00.1 - Tentativo di fornire ai Thread il campo di testo dove poter
 *              scrivere
 *
 */

package com.example.appthread00;

import com.example.appthread00.R;

. . . . .

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView t = (TextView) findViewById(R.id.testo1);

        new SimpleThread("primoTask", t ).start();
        new SimpleThread("secondoTask", t ).start();
    }

    . . . . .
}
```

## e SimpleThread.java:

```
/**
 * @author paolo
 *
 * @version 00.1 - Tentativo di fornire ai Thread il campo di testo dove poter
 *               scrivere
 */
package com.example.appthread00;

import android.widget.TextView;

public class SimpleThread extends Thread {

    TextView t;

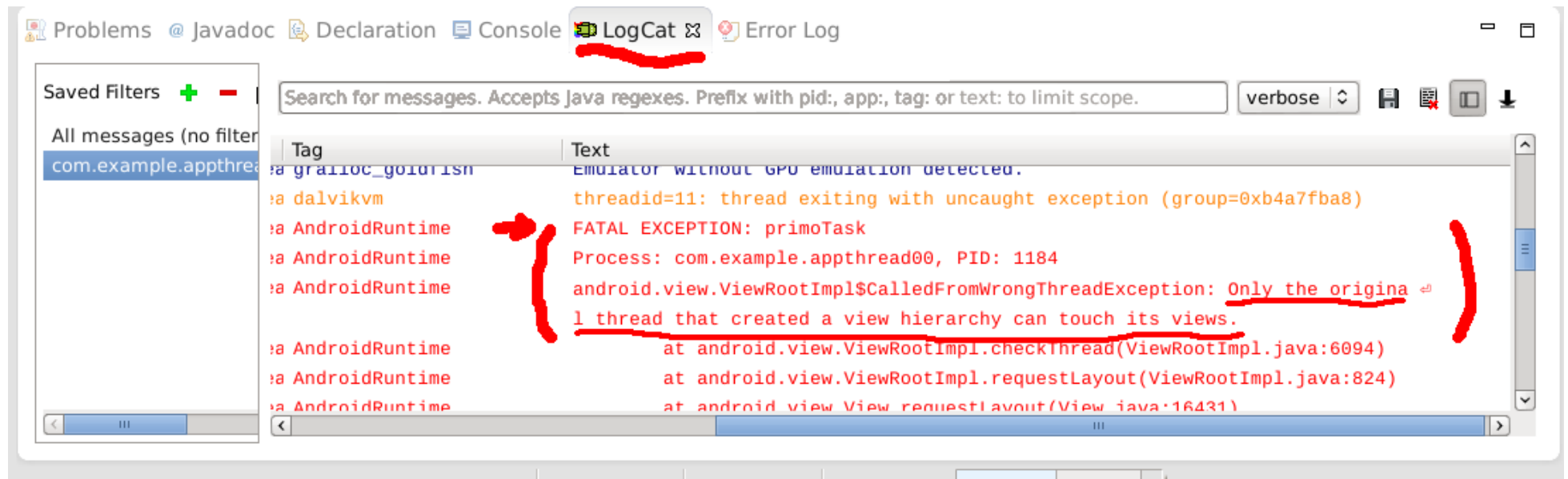
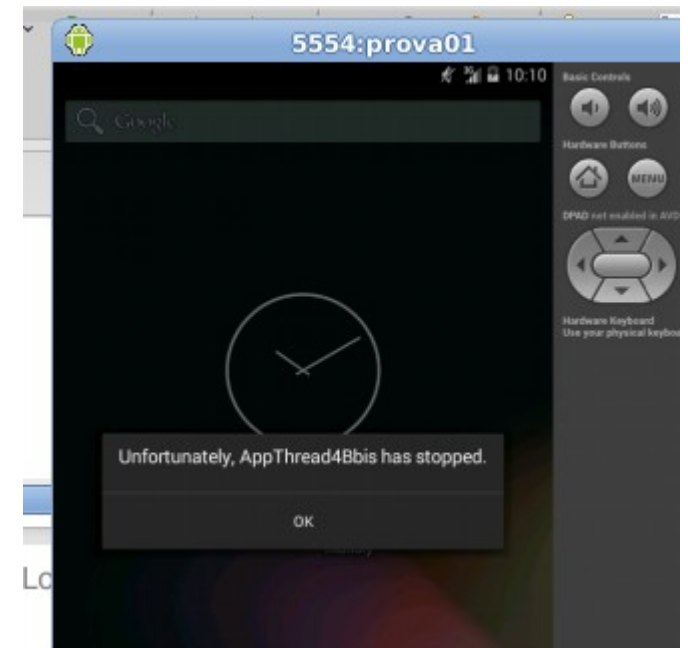
    public SimpleThread(String threadName, TextView t ) {
        super(threadName);
        this.t = t;
    }

    public void run()
    {
        for (int i = 0; i < 10; i++)
        {
            // System.out.println(this.getName()+" "+ i );
            CharSequence old = t.getText();
            t.setText( old + "\n" + this.getName()+" "+ i);

            try
            {
                sleep((int)(Math.random() * 1000));
            }
            catch (InterruptedException e) {}
        }
        // System.out.println( getName()+ " ha FINITO");
        CharSequence old = t.getText();
        t.setText( old + "\n" + this.getName()+" ha FINITO");
    }
}
```

MA L'ESECUZIONE SI BLOCCA E VA IN ERRORE !!!

Guardando ancora all'ADT in LogCat troviamo:





"Only the original thread that created a view hierarchy can touch its views."

Che vuol dire ?

Che in android le componenti della UI li può toccare solo chi le ha create!!

Nel nostro caso MainActivity !

Forse è il caso di "studiare" un po' di teoria...

<http://developer.android.com/guide/components/processes-and-threads.html>

si trova:

When an application is launched, the system creates a thread of execution for the application, called "main." This thread is very important because it is in charge of dispatching events to the appropriate user interface widgets, including drawing events. It is also the thread in which your application interacts with components from the Android UI toolkit (components from the `android.widget` and `android.view` packages). As such, the main thread is also sometimes called the UI thread.

...

Additionally, the Android UI toolkit is *not* thread-safe. So, you must not manipulate your UI from a worker thread—**you must do all manipulation to your user interface from the UI thread**. Thus, there are simply two rules to Android's single thread model:

1. Do not block the UI thread
2. Do not access the Android UI toolkit from outside the UI thread

Tra le soluzioni proposte c'è :

However, as the complexity of the operation grows, this kind of code can get complicated and difficult to maintain. To handle more complex interactions with a worker thread, you might consider using a [Handler](#) in your worker thread, to process messages delivered from the UI thread.

Visto che le altre soluzioni sono semplici affrontiamo questa.

DEFINIAMO UN HANDLER "COLLEGATO" AI THREAD , QUESTI ULTIMI COMUNICHERANNO CON L'HANDLER PER MEZZO DI MESSAGGI .

L'HANDLER PER MEZZO DEL "SOVRASCRITTO" METODO HANDLEMESSAGE() GESTIRÀ L'AGGIORNAMENTO DELLA UI.

```

/**
 * @author paolo
 *
 * @version 00.2 - Risoluzione del problema con Handler
 */

package com.example.appthread00;

import .....

@SuppressWarnings("HandlerLeak") //NECESSARIO : TOGLIERE LA RIGA E CAPIRE COSA CHIEDE IL COMPILATORE
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Handler mioHandler = new Handler(){
            @Override
            public void handleMessage(Message msg) {

                TextView t = (TextView) findViewById(R.id.testo1);

                CharSequence old = t.getText();
                t.setText( old + "\n" + (String) msg.obj);
            }
        };

        new SimpleThread("primoTask", mioHandler ).start();
        new SimpleThread("secondoTask", mioHandler ).start();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        .....
    }
}

```

```

/**
 * @author paolo
 *
 * @version 00.2 - Risoluzione del problema con Handler
 */
package com.example.appthread00;

import android.os.Handler;
import android.os.Message;

public class SimpleThread extends Thread {

    Handler h ;

    public SimpleThread(String threadName, Handler h ) {
        super(threadName);
        this.h = h;
    }

    public void run()
    {
        for (int i = 0; i < 10; i++)
        {
            // System.out.println(this.getName()+" "+ i );

            Message messaggio = Message.obtain();
            messaggio.obj = new String(this.getName()+" "+ i);
            h.sendMessage(messaggio);

            try { sleep((int)(Math.random() * 1000)); }
            catch (InterruptedException e) {}
        }
        // System.out.println( getName()+ " ha FINITO");

        Message messaggio = Message.obtain();
        messaggio.obj = new String(this.getName()+" ha FINITO");
        h.sendMessage(messaggio);

    }
}

```

AD OGNI VARIAZIONE DI VERSIONE CORRISPONDE UNA COMMIT .

ESERCIZI:

CREARE LO STESSO PROGRAMMA CON :