Asserzioni

Strumento utile durante la fase di debug

Ci permettono di testare condizioni che noi sappiamo essere vere

Sintassi, due forme:

```
assert espressione_booleana

assert espressione_booleana : altra_espressione
```

l'espressione_booleana rappresenta la condizione che noi sappiamo essere vera

Date due variabili x e y se sappiamo che x deve essere uguale a y possiamo scrivere

```
assert x == y;
```

se il risultato dell'espressione booleana è false viene lanciato un

AssertionError

Se è presente altra_espressione il suo valore viene trasportato dall'AssertionError

```
assert x == y : "x vale " + x + " e y vale " + y;
```

Esempi d'uso:

```
void f() {
    =
    for( — ) {
        =
        return;
    }
    assert false;
}
```

oppure

```
double totale = getTotale();
cb1.trasferisci(cb2, 100.01);
double totaleDopo = getTotale();
assert totale == totaleDopo : totale + " "
                              + totaleDopo;
```

la JVM esegue il codice con le asserzioni disabilitate

per abilitarle:

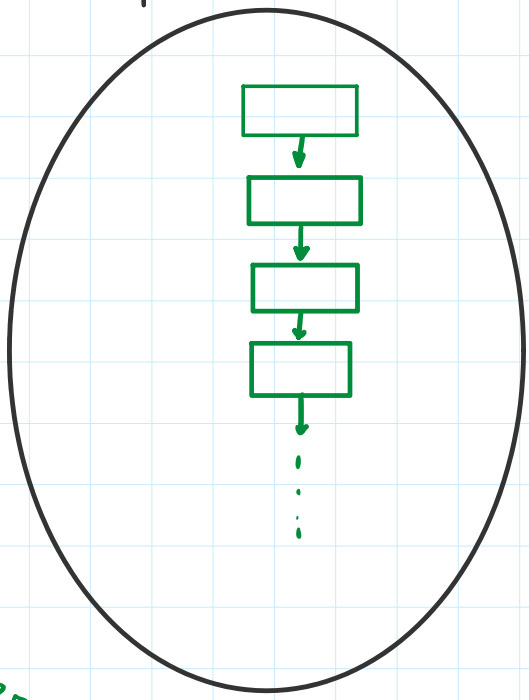
```
java -ea ....
java -enableassertions ....
```

Asserzioni ≠ Eccezioni

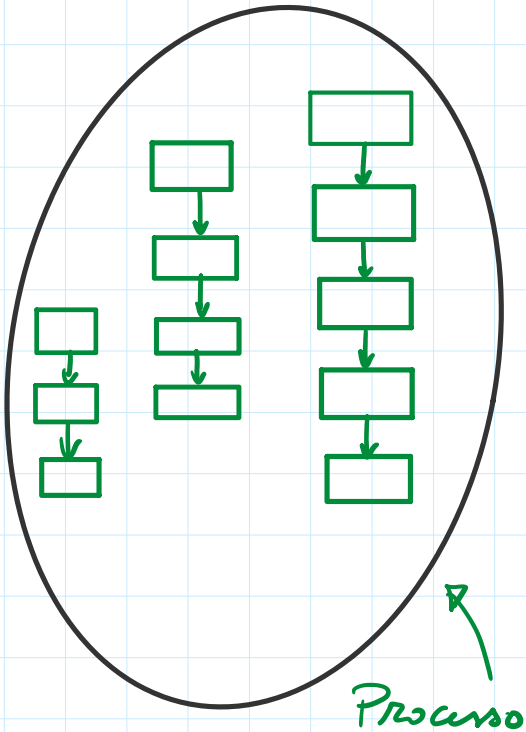
Thread

Un thread è un flusso di esecuzione indipendente

Possiamo avere più thread all'interno dello stesso processo.

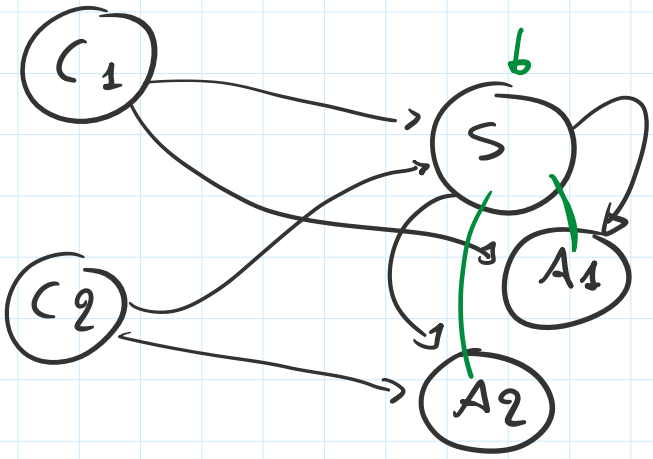


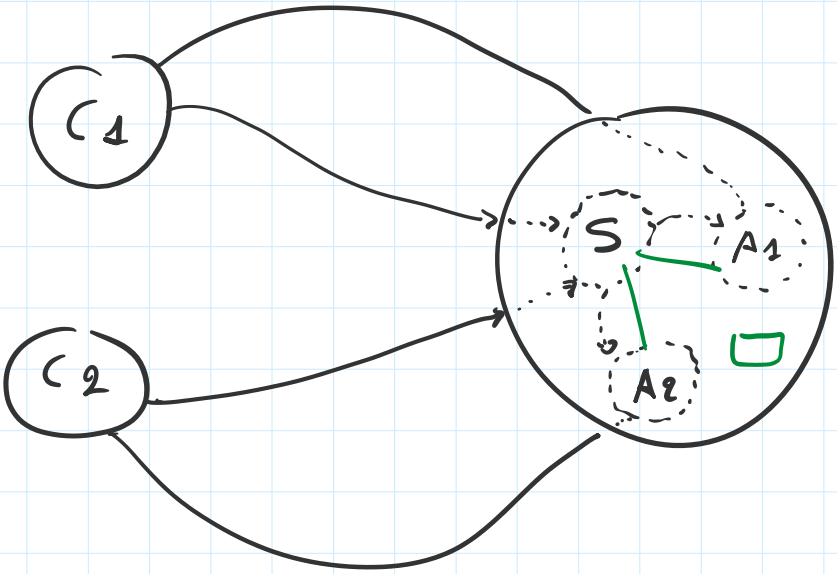
Processo con un solo flusso di esecuzione

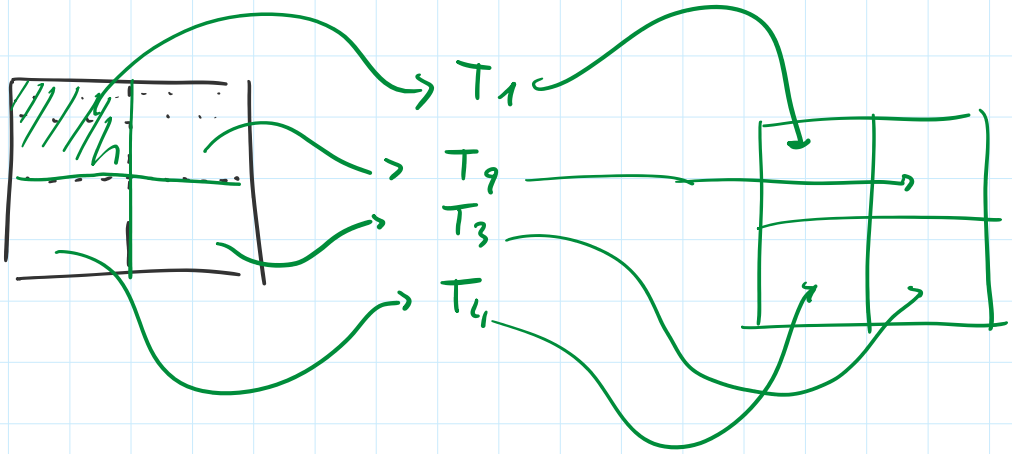


Processo con più flussi di esecuzione (più thread)

(più thread)

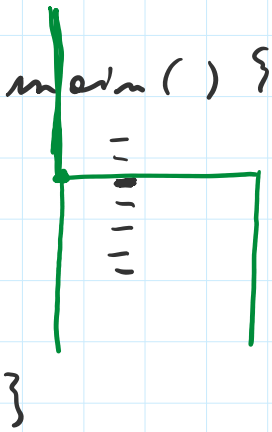






Quando mandiamo in esecuzione un programma Java la JVM crea un thread detto "main thread" che esegue il nostro codice

Il main thread può generare altri thread



in effetti un qualunque thread può generare altri thread

I thread sono rappresentati da istanze della classe java.lang.Thread

La classe Thread prevede un metodo che può essere ridefinito per specificare cosa deve fare il nuovo Thread.

Dobbiamo scrivere delle sottoclassi di Thread che ridefiniscono il metodo run()

```
public class UomoLupo extends Thread {

   private String urlo;

   public UomoLupo(String s) {
      urlo = s;
   }

   public void run(){
      for(int i=0; i<10000; i++) {
         System.out.println(urlo);
```

qui definisco il comportamento dei Thread di tipo UomoLupo

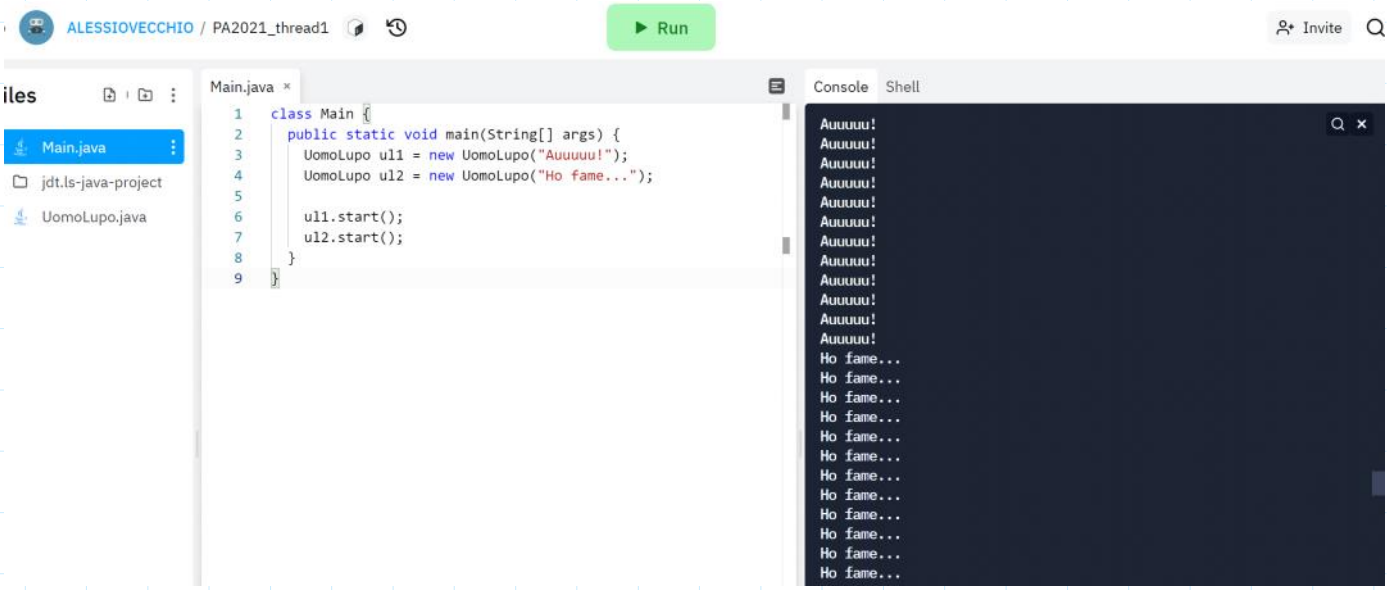
```
    public void run(){
      for(int i=0; i<10000; i++) {
        System.out.println(urlo);
      }
    }
}
```

dei UomoLupo

```
class Main {

  public static void main(String[] args) {

    UomoLupo ul1 = new UomoLupo("Auuuuu!");
    UomoLupo ul2 = new UomoLupo("Ho fame...");

    ul1.start();
    ul2.start();
  }
}
```

} i flussi di esecuzione vengono attivati con start()

PA2021_thread1



Per far partire un thread dobbiamo chiamare il suo metodo start()

(NOTA: non il metodo run())

Gli oggetti thread non sono soggetti a garbage collection (fino a che sono in vita) anche se non conserviamo il riferimento

Il processo di garbage collection considera raggiungibili tutti gli oggetti nello heap usando come punti di partenza gli stack di tutti i thread in vita

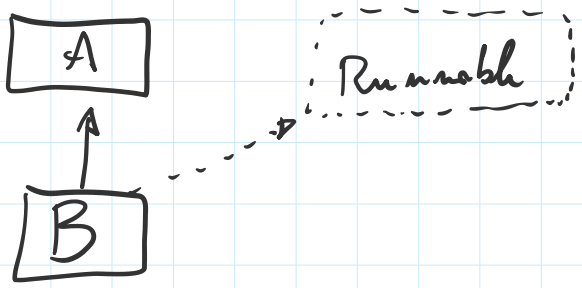
Ogni thread ha il suo stack

Siccome in Java l'eredità è singola, il dover estendere la classe Thread può essere una forte limitazione

Per risolvere questo problema è possibile seguire una strada alternativa che si basa sull'uso dell'interfaccia Runnable

L'interfaccia Runnable è fatta così:

```
public interface Runnable {
    void run();
}
```



Quindi dobbiamo creare un oggetto della classe in questione
Lo dobbiamo passare al costruttore della classe thread
Facciamo partire il flusso di esecuzione invocando il metodo start sull'oggetto thread

Esempio

```
public class ComportamentoMostro implements Runnable {

   private String nome;

   public ComportamentoMostro(String n) {
      nome = n;
   }

   public void run(){
      for(int i=0; i<10000; i++) {
         System.out.println(nome + ": Auagagaaaaaah");
            // Se metto yield() il comportamento e' "educato"
            // Thread.yield();
      }
   }
}
```

Quindo creo gli oggetti e li passo al costruttore di Thread:

```
class Main {
   public static void main(String[] args) {

      ComportamentoMostro cm1 = new ComportamentoMostro("Homo Iuno");
```

```
class Main {
    public static void main(String[] args) {

        ComportamentoMostro cz1 = new ComportamentoMostro("Uomo lupo");
        ComportamentoMostro cz2 = new ComportamentoMostro("Zombie");

        Thread t1 = new Thread(cz1);
        Thread t2 = new Thread(cz2);

        t1.start();
        t2.start();
    }
}
```

← Vengono attivati i miei flussi

Stati di un thread: