1 Lezione del 14-10-25

1.1 Asserzioni

Le **asserzioni** in Java permettono di ineserire condizioni presunte vere (utili al debug) da verificare a tempo di esecuzione. La sintassi è `assert <espressione>`, dove l'espressione deve essere booleana e valutare a *vera*.

Il fallimento di un'asserzione genera un errore di tipo *AssertionError*.

Un'altra possibile sintassi per le asserzioni è `assert <condizione1> : <condizione2>`, dove la condizione 1 è analoga alla precedente, mentre la condizione 2 è una stringa viene valutata se la prima vale falso per riportare informazione di diagnistica.

Ad esempio, si potrà dire:

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

Il blocco incodizionato del codice (magari se si salta ad un frammento di codice che dovrebbe essere irraggiungibile) si fa semplicemente con `assert false`.

Di norma, la JVM (`java ...`) esegue il codice con le asserzioni disabilitate. Se si vogliono abilitare, bisogna fornire l'argomento `-enableassertions` (abbreviato `-ea`).

1.2 Thread

I **thread** in Java sono un astrazione per i flussi di esecuzioni indipendenti.

I thread sono flussi diversi che vivono all'interno dello stesso *processo*: ergo hanno accesso alle stesse risorse e allo stesso spazio di indirizzamento. In questo possono passarsi oggetti fra di loro senza doversi appellare ai servizi di **IPC** (*Inter Process Communication*) forniti dal S/O.

Un esempio di utilizzo di thread può essere in un'applicazione server, dove ad ogni client che effettua richieste si associa (finché possibile) un suo thread dedicato. Questo semplifica sia lo sviluppo lato server (un singolo thread gestisce un singolo client), che l'esperienza lato client (si andrà a parlare con un singolo thread).

1.2.1 Thread e multithreading

Ricordiamo che i thread sono un'astrazione. Nei moderni processori, sappiamo di aver a disposizione più di un unità di elaborazione (*core*), per cui l'esecuzione parallela di istruzioni è effettivamente possibile. Questo non si traduce direttamente nell'esecuzione di ognuno dei nostri thread in Java su un core diverso: è compito del S/O capire se delegare a più core l'esecuzione dei thread, o se eseguirli tutti su un solo core in *time-slicing*.

1.2.2 Thread default

Quando mandiamo in esecuzione un programma Java, la JVM crea un thread detto *main thread* (cioè il thread *principale* o *di default*) che esegue il codice definito nel metodo `main` della classe specificata.

Questo metodo può poi definire, attraverso il suo flusso di esecuzione, altri thread che si evolveranno quindi parallelamente a quello principale.

1.2.3 Oggetto thread

Il modo idiomatico in Java di realizzare la funzionalità dei thread è quello di sfruttare un apposito oggetto, l'oggetto *Thread*.

L'oggetto Thread definisce un metodo, `run()`, all'interno del quale possiamo definire il flusso di esecuzione proprio del thread.