

Thread in Java

Il linguaggio Java consente di creare e gestire Thread mettendo a disposizione del programmatore la classe *Thread* e l'interfaccia *Runnable*. Il programmatore potrà quindi implementare dei thread mediante la scrittura di classi che estendono la classe *Thread* o implementano l'interfaccia *Runnable*. Anche se in un programma Java non viene definito esplicitamente alcun thread, l'esecuzione del metodo *main* (necessaria per far partire il programma) istanzia implicitamente un thread relativo all'esecuzione del programma.

Supponiamo di aver scritto una semplice classe *Printer* nel modo seguente

```
public class Printer {  
    int min, max;  
  
    public Printer(int min, int max) {  
        this.min=min;  
        this.max=max;  
    }  
  
    public void esegui() {  
        for (int i=min; i<=max; i++)  
            System.out.println(i);  
    }  
}
```

Istanziando un oggetto di tipo *Printer* con parametri *min* e *max*, e invocando su esso il metodo *esegui*, verranno scritti su video, uno per linea, i numeri compresi tra i valori *min* e *max*. Quindi, il risultato di questa porzione di codice

```
...  
Printer p=new Printer(4,6);  
p.esegui();  
System.out.println("Finito");  
...
```

produrrà il seguente output

```
4  
5  
6  
Finito
```

Ciò avviene perché l'istruzione che scrive su output la stringa "Finito" viene eseguita solo dopo che le istruzioni contenute nel metodo *esegui* sono state completamente eseguite.

In particolare, il ciclo di *for* contenuto nel metodo *esegui*, non viene interrotto poiché non vi è alcun altro thread in esecuzione.

Scriviamo ora una classe simile a *Printer*, ma che effettua la stampa dei numeri come thread distinto da quello del *main*.

```
public class ThreadPrinter extends Thread {  
    int min, max;  
  
    public ThreadPrinter(int min, int max) {
```

```

        this.min=min;
        this.max=max;
    }

    public void run() {
        for (int i=min; i<=max; i++)
            System.out.println(i);
    }
}

```

Istanziando un oggetto di tipo `ThreadPrinter` costruiamo un nuovo thread, che ancora però non è in esecuzione.

La classe `Thread` esporta il metodo `start()`, che serve per segnalare alla Java Virtual Machine che il thread può essere posto in esecuzione. Quando lo scheduler della JVM metterà in esecuzione il thread, saranno eseguite sequenzialmete le istruzioni definite nel metodo `run`.

Quindi, se in un programma scriviamo

```

...
ThreadPrinter tp=new ThreadPrinter(4,6);
tp.start();
System.out.println("Finito");
...

```

il thread `tp` sarà pronto per essere eseguito, ma in modo distaccato dal thread del main, che continuerà con l'istruzione che scrive su output la stringa "Finito". Pertanto, non possiamo prevedere in modo esatto quale sarà l'output su video. Tre possibilità sono le seguenti

```

4
5
6
Finito

```

```

4
5
Finito
6

```

```

Finito
4
5
6

```

Nel primo caso il thread main lascia il controllo al thread `tp`, che viene quindi posto in esecuzione fino al termine delle istruzioni contenute nel metodo `run`. Anche nel secondo caso il thread main lascia il controllo al thread `tp`, ma questa volta, prima che `tp` esaurisca le sue operazioni, il controllo viene ripassato al main, viene quindi eseguita l'istruzione di scrittura su video di "Finito", e quindi il controllo ripassa a `tp`, che termina le sue operazioni. Infine, nel terzo caso, il controllo non viene subito passato a `tp`, ma viene prima effettuata la scrittura su video di "Finito", e quindi viene posto in esecuzione `tp`, fino al completamento delle sue operazioni.

Ricapitolando, quando un programmatore vuole implementare un thread, può creare una classe che estende `Thread`, ridefinire il metodo `run`, specificando quali sono le operazioni che il thread deve compiere. Successivamente, è possibile istanziare un oggetto del tipo definito e porlo in esecuzione mediante il metodo `start`. Tenere presente che essendo il metodo `run` pubblico, esso potrà essere

invocato esplicitamente dal programmatore. Tuttavia, in tal caso, non verrebbe avviata l'esecuzione del nuovo thread creato, ma sarà esclusivamente eseguito il metodo run dal thread che lo invoca.

Come accennato in precedenza, è possibile utilizzare i thread di Java anche utilizzando l'interfaccia `Runnable`. Questo meccanismo è indispensabile qualora si voglia creare una nuova classe con funzionalità di thread estendendo una classe diversa da `Thread`. Ad esempio, avendo implementato la classe `Printer`, potremmo implementare una classe `Printer2`, che scriva i numeri in un thread a se stante, estendendo `Printer` ed implementando `Runnable`. L'interfaccia `Runnable` dichiara il metodo `run`, che pertanto dovrà essere definito da ogni classe che implementa quest'interfaccia.

```
public class Printer2 extends Printer implements Runnable {  
    public Printer2(int min, int max) {  
        super(min,max);  
    }  
    public void run() {  
        esegui();  
        System.out.println("Finito");  
    }  
}
```

Il metodo `run` di `Printer2` invoca il metodo `esegui` definito nella classe `Printer`, estesa da `Printer2`. Creando un oggetto di tipo `Printer2`, non abbiamo ancora la possibilità di avviarlo come thread. Infatti, il metodo `start` è esportato dalla classe `Thread`, e `Printer2` non è di tipo `Thread` (è di tipo `Printer` e `Runnable`). Java, però, consente di creare un oggetto `Thread` a partire da un oggetto `Runnable`. Infatti, `Thread` ammette un costruttore che riceve come parametro un oggetto `Runnable`. Possiamo quindi creare un thread basato su `Printer2` nel seguente modo

```
...  
Printer2 p=new Printer2(1,100);  
Thread tp=new Thread(p);  
tp.start();  
...
```

Viene così creato e posto in attesa d'esecuzione un thread che eseguirà il metodo `run` dell'oggetto `p`, ovvero dell'oggetto di tipo `Runnable` passato come parametro al costruttore.

Si noti che l'esecuzione del thread `tp` può essere interrotta. Tuttavia, ciò che è contenuto nel metodo `run` viene eseguito sequenzialmente. Pertanto, verranno prima stampati su video i numeri compresi tra 1 e 100, e poi sarà stampata la stringa "Finito".