UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Programmazione 3
e
Laboratorio di Programmazione 3

Gestione dei Thread

Proff. Angelo Ciaramella – Emanuel Di Nardo

Thread

- Definizione
  - Un thread è un processore virtuale che esegue codice su determinati dati
- multithreading
  - definire più di un thread
    - ognuno ha compiti da eseguire parallelamente
    - possono dialogare tra loro allo scopo di spartirsi nella maniera ottimale l'utilizzo delle risorse del sistema

Thread

- La Virtual Machine offre uno strato d'astrazione per gestire il multithreading direttamente dal linguaggio

- I meccanismi della gestione
  - Classe `Thread` e interfaccia `Runnable` (`java.lang`)
  - Classe `Object` (`java.lang`)
  - JVM e nella keyword `synchronized`

Thread

- I meccanismi della gestione
  - Classe `Thread` e interfaccia `Runnable` (`java.lang`)
  - Classe `Object` (`java.lang`)

- Cosa utilizzare?
  - Interfaccia `Runnable` quando abbiamo bisogno di estendere una qualche classe
  - Classe `Thread` quando non abbiamo bisogno di altre tipologie di relazioni

Classe Thread

```
public class ThreadExists {
public static void main(String args[]) {
      Thread t = Thread.currentThread();
      t.setName("Thread principale");
      t.setPriority(10);
      System.out.println("Thread in esecuzione: " + t );
      try {
      for (int n = 5; n > 0; n--) {
         System.out.println("" + n);
         t.sleep(1000);
         }
    }
    catch (InterruptedException e) {
        System.out.println("Thread interrotto");
    }
   }
}
```

Indirizzo dell'oggetto Thread

Priorità da 1 a 10 (default 5)

Pausa di 1000 millesecondi

Esempio di esecuzione di un thread

Thread

- più thread
  - istanziarne altri dalla classe Thread
  - passare al costruttore un'istanza di una classe che implementa l'interfaccia `Runnable`
    - quando sarà fatto partire (mediante la chiamata al metodo `start()`), eseguirà il codice del metodo `run()` dell'istanza associata
- interfaccia `Runnable`
  - implementazione del solo metodo `run()` che definisce il comportamento di un thread
  - avvio di un thread si ottiene con la chiamata del metodo `start()`

Codice di riferimento

`ThreadCreation.java`

Thread

- La classe Thread stessa implementa l'interfaccia Runnable
  - implementazione vuota del metodo `run()`
  - è possibile fare eseguire ad un thread il metodo `run()` definito all'interno dello stesso oggetto thread

```
public class CounterThread extends Thread {
   public void run() {
      for (int i = 0; i<10; ++i)
       System.out.println(i);
      }
}

// Esecuzione di un thread
CounterThread thread = new CounterThread ();
thread.start();
```


Sincronyzed

- Quando due o più thread necessitano contemporaneamente dell'accesso ad una fonte di dati condivisa
  - bisogna che accedano ai dati uno alla volta
    - i loro metodi vanno sincronizzati (synchronized)

```
synchronized (nomeOggetto) {

   ... Blocco sincronizzato

}
```


Sincronyzed

- La sincronizzazione si basa sul concetto di lock (o monitor)

- Ogni oggetto ha un lock associato con esso:
  - Deriva dalla classe Object
  - Ogni thread per accedere ad un oggetto deve acquisire il lock sull'oggetto e rilasciarlo al termine dell'operazione

Sincronyzed

- Quando due o più thread necessitano contemporaneamente dell'accesso ad una fonte di dati condivisa
  - bisogna che accedano ai dati uno alla volta

- Livelli di sincronizzazione:
  - Metodi
  - Blocchi
    - Preferibile su oggetti final
  - Variabili
    - Dipende dalla prima tipologia - Atomic
  - Static

Codice di riferimento

`ThreadSynchronization`

Esercizio

- Abbiamo una conto condiviso con la nostra famiglia ed abbiamo N carte associate al nostro conto. La banca deve tutelarsi per evitare frodi nei prelievi e depositi.
- Modificare BankAccount e scegliere due metodi diversi di sincronizzazione tra quelli possibili:
  - Sincronizzare il metodo deposit ed il metodo withdraw per proteggere la variabile balance
  - Creare N thread che effettuano la stessa operazione