UNIVERSITÀ DELLA CALABRIA
Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Corso di Reti di Calcolatori

Introduzione ai Thread in Java

A.A. 2009/2010

Thread

- Un ***thread*** è un flusso sequenziale di esecuzione di istruzioni all'interno di un **processo**.

- In un programma si possono far partire più thread che sono eseguiti concorrentemente. Nei computer a singola CPU la concorrenza viene simulata con una ***politica di scheduling*** che alterna l'esecuzione dei singoli thread.

- Tutti i thread eseguono all'interno del contesto di esecuzione di un solo processo, ovvero condividono le stesse variabili del programma.

- Ciascun ***processo*** tradizionale ha invece il proprio contesto di esecuzione.

Thread in Java

- Tutti i programmi Java comprendono almeno un thread. Anche un programma costituito solo dal metodo main viene eseguito come un singolo thread. Inoltre, Java fornisce strumenti che consentono di creare e manipolare thread aggiuntivi nel programma

Esistono due modi per implementare thread in Java:

- Definire una sottoclasse della classe **Thread**.

- Definire una classe che implementa l'interfaccia **Runnable**. Questa modalità è più flessibile, in quanto consente di definire un thread che è sottoclasse di una classe diversa dalla classe Thread.

Definire una sottoclasse della classe Thread

1. Si definisce una nuova classe che estende la classe Thread. La nuova classe deve ridefinire il metodo ***run()*** della classe Thread.

2. Si crea un'istanza della sottoclasse tramite ***new***.

3. Si chiama il metodo ***start()*** sull'istanza creata. Questo determina l'invocazione del metodo run() dell'oggetto, e manda in esecuzione il thread associato.

Esempio di sottoclasse di Thread

```
class Saluti extends Thread {
  public Saluti(String nome) {
    super(nome);
  }
  public void run() {
    for (int i = 0; i < 10; i++)
      System.out.println("Ciao da "+getName());
  }
}
public class ThreadTest {
  public static void main(String args[]) {
    Saluti t = new Saluti("Primo Thread");
    t.start();
  }
}
```

Definire una classe che implementa Runnable

1. Si definisce una nuova classe che implementa l'interfaccia Runnable. La nuova classe deve implementare il metodo ***run()*** dell'interfaccia Runnable.

2. Si crea un'istanza della classe tramite ***new***.

3. Si crea un'istanza della classe Thread, passando al suo costruttore un riferimento all'istanza della nuova classe definita.

4. Si chiama il metodo ***start()*** sull'istanza della classe Thread creata, determinando l'invocazione del metodo run() dell'oggetto Runnable associato.

Esempio di classe che implementa Runnable

```
class Saluti implements Runnable {
  private String nome;
  public Saluti(String nome) {
    this.nome = nome;
  }
  public void run() {
    for (int i = 0; i < 10; i++)
      System.out.println("Ciao da "+nome);
  }
}
public class RunnableTest {
  public static void main(String args[]) {
    Saluti s = new Saluti("Secondo Thread");
    Thread t = new Thread (s);
    t.start();
  }
}
```

La classe Thread (1)

Costruttori principali:

Thread (): crea un nuovo oggetto Thread.

Thread (String name): crea un nuovo oggetto Thread con nome **name**.

Thread (Runnable target): crea un nuovo oggetto Thread a partire dall'oggetto **target**.

Thread (Runnable target, String name): crea un nuovo oggetto Thread con nome **name** a partire dall'oggetto **target**.

Metodi principali:

String getName(): restituisce il nome di questo Thread.

void join() *throws InterruptedException*: attende fino a quando questo Thread non termina l'esecuzione del proprio metodo run.

La classe Thread (2)

void join(long millis) *throws InterruptedException*: attende, per un tempo massimo di **millis** millisecondi, fino a quando questo Thread non termina l'esecuzione del proprio metodo run.

void run(): specifica le operazioni svolte dal Thread. Deve essere ridefinito dalla sottoclasse, altrimenti non effettua alcuna operazione. Se il Thread è stato costruito a partire da un oggetto Runnable, allora verrà invocato il metodo run di tale oggetto.

static void sleep(long millis) *throws InterruptedException*: determina l'interruzione dell'esecuzione del Thread corrente per un tempo di **millis** millisecondi.

void start(): fa partire l'esecuzione del Thread. Viene invocato il metodo run di questo Thread.

Un programma sequenziale (1)

```
class Printer {
  private int from;
  private int to;

  public Printer (int from, int to) {
    this.from = from;
    this.to = to;
  }

  public void print () {
    for (int i = from; i <= to; i++)
      System.out.print (i+"\t");
  }
}

public class PrinterApp {
  public static void main (String args[]) {
    Printer p1 = new Printer (1,10);
    Printer p2 = new Printer (11,20);
    p1.print();
    p2.print();
    System.out.println ("Fine");
  }
}
```

Un programma sequenziale (2)

L’esecuzione di **PrinterApp** genera sempre il seguente output:

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
Fine

I metodi:

p1.print()

p2.print()

System.out.println(“Fine”)

sono eseguiti in modo sequenziale.

Un programma threaded (1)

```
class TPrinter extends Thread {
  private int from;
  private int to;

  public TPrinter (int from, int to) {
    this.from = from;
    this.to = to;
  }

  public void run () {
    for (int i = from; i <= to; i++)
      System.out.print (i+"\t");
  }
}

public class TPrinterApp {
  public static void main (String args[]) {
    TPrinter p1 = new TPrinter (1,10);
    TPrinter p2 = new TPrinter (11,20);
    p1.start();
    p2.start();
    System.out.println ("Fine");
  }
}
```

Un programma threaded (2)

L'esecuzione di **TPrinterApp** potrebbe generare il seguente output:

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
Fine

oppure:

Fine
1 2 3 **11** 4 **12** 5 **13** 6 **14**
7 **15** 8 **16** 9 **17** 10 **18** **19** **20**

oppure:

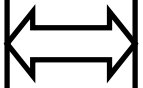
1 2 **11** **12** 3 4 ***Fine***
5 **13** **14** 6 7 **15** 8 **16** **17** 9
18 **19** **20** 10

e così via: non è possibile fare alcuna assunzione sulla velocità relativa di esecuzione dei thread. L'unica certezza è che le operazioni all'interno di un dato thread procedono in modo sequenziale.

Imporre la sequenzialità

Per imporre la sequenzialità nell'esecuzione dei diversi thread si può fare uso del metodo join().

```
public class TPrinterApp {
  public static void main (String args[]) {
    TPrinter p1 = new TPrinter (1,10);
    TPrinter p2 = new TPrinter (11,20);
    p1.start();
    try {
      p1.join();
    } catch (InterruptedException e) {
      System.out.println(e);
    }
    p2.start();
    try {
      p2.join();
    } catch (InterruptedException e) {
      System.out.println(e);
    }
    System.out.println ("Fine");
  }
}
```



```
public class PrinterApp {
  public static void main (String args[]) {
    Printer p1 = new Printer (1,10);
    Printer p2 = new Printer (11,20);
    p1.print();
    p2.print();
    System.out.println ("Fine");
  }
}
```

Somma in concorrenza (1)

```
class Sommatore extends Thread {
  private int da;
  private int a;
  private int somma;
  public Sommatore (int da, int a) {
    this.da = da;
    this.a = a;
  }
  public int getSomma() {
    return somma;
  }
  public void run () {
    somma = 0;
    for (int i = da; i <= a; i++)
      somma += i;
  }
}
```

Somma in concorrenza (2)

```
public class Sommatoria {
  public static void main (String args[]) {
    int primo = 1;
    int ultimo = 100;
    int intermedio = (primo+ultimo)/2;
    Sommatore s1 = new Sommatore (primo,intermedio);
    Sommatore s2 = new Sommatore (intermedio+1,ultimo);
    s1.start();
    s2.start();
    try {
      s1.join();
      s2.join();
    } catch (InterruptedException e) { System.out.println (e); }
    System.out.println (s1.getSomma()+s2.getSomma());
  }
}
```

Thread con attività ciclica (1)

Ciclo finito:

```
public void run ()
{
  for (int i = 0; i < n; i++)
  {
     istruzioni
  }
}
```

Ciclo infinito:

```
public void run ()
{
  while (true)
  {
     istruzioni
  }
}
```

Thread con attività ciclica (2)

Ciclo con terminazione:

```
class  myThread extends Thread {
  private boolean continua;
  public myThread () {
    continua = true;
  }
  public void termina ()  {
    continua = false;
  }
  public void run () {
    while (continua)
    {
       istruzioni
    }
  }
}
```