Java Threads

esempi

Creazione

```
public class ProvaThread {

    public static void main(String[] args) {

        Runnable r = new MyRunnable();

        Thread t = new Thread(r);
        .
        .
        .
        .
    }
}
```

```
class MyRunnable implements Runnable {

    public void run() {
        // codice del thread
    }

}
```

Creazione

```
public class ProvaThread {

    public static void main(String[] args) {

        Runnable r = new MyRunnable();

        Thread t1 = new Thread(r, "pippo");
        Thread t2 = new Thread(r, "pluto");
        .
        .
        .
      }
}
```

Nomi diversi a thread diversi

Lo stesso runnable viene usato per i due threads

```
class MyRunnable implements Runnable {

    public void run() {

        Thread t = Thread.currentThread();
        System.out.println("il mio nome e' " + t.getName());
        System.out.println("il mio id e' " + t.getId());
        System.out.println("t = " + t.toString());

        // resto del codice del thread
    }
}
```

restituisce il thread che sta eseguendo MyRunnable

Output

```
il mio nome e' pluto
il mio id e' 8
t = Thread[pluto,5,main]
```

Creazione

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(11, so);
        Runnable r2 = new MyRunnable(22, so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

    }
}

class MyRunnable implements Runnable {
int num;
OggettoCondiviso so;

MyRunnable(int num, OggettoCondiviso so) {
    this.num = num;
    this.so = so;
}

public void run() {
    // codice del thread: utilizza so e num
}
```

i parametri vengono passati al costruttore di MyRunnable

lo stesso oggetto viene passato a due threads

Priorità

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(11, so);
        Runnable r2 = new MyRunnable(22, so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

        t1.setPriority(Thread.MIN_PRIORITY);
        t2.setPriority(Thread.MAX_PRIORITY);

        System.out.println("ora t1 ha priorita' " + t1.getPriority());
        System.out.println("ora t2 ha priorita' " + t2.getPriority());


    }
}
```

Output

```
ora t1 ha priorita' 1
ora t2 ha priorita' 10
```

Lancio

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(11, so);
        Runnable r2 = new MyRunnable(22, so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

        t1.start();
        t2.start();



    }
}
```

Attesa Terminazione

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(so);
        Runnable r2 = new MyRunnable(so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
        }
        catch (Exception e){
            System.err.println("errore");
        }
    }
}
```

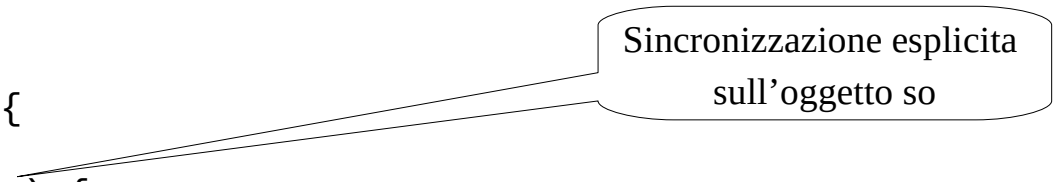
Sincronizzazione Esplicita

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;

    MyRunnable(int iNum, OggettoCondiviso so) {
        this.iNum = iNum;
        this.so = so;
    }

    public void run() {

        synchronized(so) {
            so.conta = so.conta + 7;
        }
    }
}
```



Sincronizzazione Esplicita

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;

    MyRunnable(int iNum, OggettoCondiviso so) {
        this.iNum = iNum;
        this.so = so;
    }

    public void run() {

        synchronized(OggettoCondiviso.class) {
            OggettoCondiviso.iQuanti++;
        }
    }
}
```

Sincronizzazione esplicita sulla classe OggettoCondiviso per campi static

Sincronizzazione Implicita

```
class OggettoCondiviso {
    static int iQuanti = 0;
    int conta;

    OggettoCondiviso(int conta) {
        this.conta = conta;
        iQuanti++;
    }

    synchronized void decrem(int dec) {
        conta-=dec;
    }
}
```

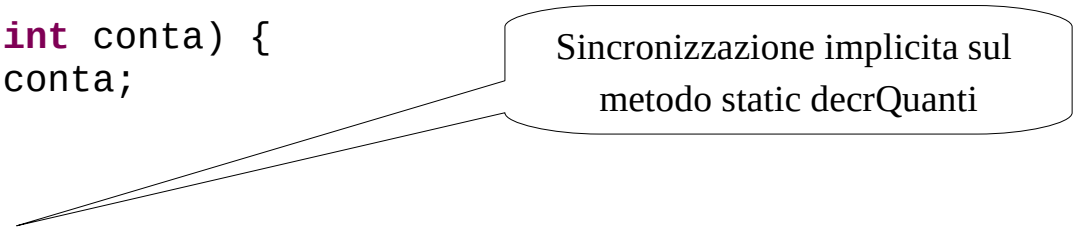
Sincronizzazione implicita sul metodo decrem: l'oggetto su cui avviene la sincronizzazione e' quello su cui viene invocato il metodo decrem

Sincronizzazione Implicita

```
class OggettoCondiviso {
    static int iQuanti = 0;
    int conta;

    OggettoCondiviso(int conta) {
        this.conta = conta;
        iQuanti++;
    }

    static synchronized void decrQuanti() {
        iQuanti--;
    }
}
```



Wait

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;


    public void run() {

        try {
            synchronized(so) {
                while (so.conta < 1)
                    so.wait();

                so.conta-=2;
            }
        }
        catch (InterruptedException e) {
            System.out.println("uscito dalla wait per interruzione");
        }
    }
}
```

Sospende il thread fino a che un altro thread non esegue una notify sullo stesso oggetto

Un altro thread ha lanciato un interrupt() a questo thread

Notify

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;


    public void run() {

        synchronized(so) {

            so.increm(2);

            so.notify();
        }
    }
}
```

risveglia un thread che aveva eseguito una wait sullo stesso oggetto

notifyAll() risveglia tutti i threads che avevano eseguito una wait sullo stesso oggetto

Interrupt

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(11, so);
        Runnable r2 = new MyRunnable(22, so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

        t1.start();
        t2.start();


        t1.interrupt();



    }
}
```

Yield

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;

    MyRunnable(int iNum, OggettoCondiviso so) {
        this.iNum = iNum;
        this.so = so;
    }

    public void run() {

        // altre istruzioni

        Thread.yeld();

        // altre istruzioni
    }
}
```

Il thread che sta eseguendo MyRunnable cede il processore ad un altro thread

Sleep

```
class MyRunnable implements Runnable {
    int iNum;
    OggettoCondiviso so;

    MyRunnable(int iNum, OggettoCondiviso so) {
        this.iNum = iNum;
        this.so = so;
    }

    public void run() {

        // altre istruzioni

        Thread.sleep(1000);

        // altre istruzioni
    }
}
```

Il thread che sta eseguendo MyRunnable cede il processore ad un altro thread e dorme 1000 millis

Stato

NEW: A thread that has not yet started is in this state.

RUNNABLE: A thread in the runnable state is executing in the Java virtual machine but it may be waiting for other resources from the operating system such as processor.

BLOCKED: Thread state for a thread blocked waiting for a monitor lock. A thread in the blocked state is waiting for a monitor lock to enter a synchronized block/method or reenter a synchronized block/method after calling Object.wait.

WAITING: A thread is in the waiting state due to calling one of the following methods:

- `Object.wait` with no timeout
- `Thread.join` with no timeout

A thread in the waiting state is waiting for another thread to perform a particular action.

TIMED_WAITING: Thread state for a waiting thread with a specified waiting time. A thread is in the timed waiting state due to calling one of the following methods with a specified positive waiting time:

- `Thread.sleep`
- `Object.wait` with timeout
- `Thread.join` with timeout

TERMINATED: A thread that has exited is in this state.

Stato

```
public class ProvaThread {

    public static void main(String[] args) {

        OggettoCondiviso so = new OggettoCondiviso(33);

        Runnable r1 = new MyRunnable(11, so);
        Runnable r2 = new MyRunnable(22, so);

        Thread t1 = new Thread(r1, "primo_th");
        Thread t2 = new Thread(r2, "secondo_th");

        System.out.println("stato di t1 prima della start " + t1.getState());

        t1.start();
        t2.start();

        System.out.println("stato di t1 " + t1.getState());
        System.out.println("stato di t2 " + t2.getState());

        try {
            t1.join();
            System.out.println("stato di t1 dopo la join " + t1.getState());
        }
        catch (Exception e) {
            System.err.println("errore");
        }
    }
}
```

NON usare la getState() per le sincronizzazioni!!!

Output

```
stato di t1 prima della start NEW
stato di t1 TIMED_WAITING
stato di t2 RUNNABLE
stato di t1 dopo la join TERMINATED
```

Metodi Deprecated

```
stop()

suspend()

resume()

destroy()
```