Il linguaggio Java

Programmi d'esempio

I thread

Il linguaggio Java

Programmi d'esempio

La classe Thread e l'interfaccia Runnable

Classe Thread

```
public class SimpleThread extends Thread {
  public SimpleThread(String str) {
  super(str);
  }

  public void run() { // sovrapposizione
      for (int i = 0; i < 10; i++) {
      System.out.println(i + " " +
                         getName());
          try {
              sleep((long)(Math.random()*1000));
          } catch(InterruptedException e) {}
  }
  System.out.println("FINITO! " + getName());
  }
  public static void main (String[] args) {
      new SimpleThread("Pippo").start();
      new SimpleThread("Pluto").start();
 }
}
```


L'interfaccia Runnable

```
public class SimpleThread implements Runnable {
  private Thread thr;

  public SimpleThread(String str) {
    thr = new Thread(this, str);
    thr.start();
  }

  public void run() { // implementazione
    for (int i = 0; i < 10; i++) {
      System.out.println(i + " " + thr.getName());
      try {
        Thread.sleep((long)(Math.random()*1000));
      } catch (InterruptedException e) {}
    }
    System.out.println("FINITO! " + thr.getName());
  }

  public static void main (String[] args) {
    new SimpleThread("Pippo");
    new SimpleThread("Pluto");
  }
}
```

Il metodo join

```
public class JoinDemo extends Thread {
   public JoinDemo(String name) {
      super(name);
   }

   public static void main(String[] args)
                                        throws InterruptedException {
      System.out.println("main start");
      Thread t = new JoinDemo("pippo");
      t.start();
      t.join(); // comment this line
      System.out.println("main end");
   }

   public void run() {
      System.out.println(getName() + " start");
      for (int tick = 0; tick < 10000; tick++);
      System.out.println(getName() + " end");
   }
}
// output          output senza join
// main start      main start
// pippo start     main end
// pippo end       pippo start
// main end        pippo end
```


Il metodo yield

```
public class MyThread extends Thread {
   private int tick = 1;
   private int num;
   private final static int NUMTHREAD = 2;

   public MyThread(int num) {
      this.num = num;
   }

   public void run() {
      while (tick < 400000) { // CPU-intensive code
         tick++;
         if ((tick % 50000) == 0) {
            System.out.println("Thread #" + num +
                                                ", tick = " + tick);
//          yield(); // (1)
         }
      }
   }

   public static void main(String[] args) {
      MyThread[] runners = new MyThread[NUMTHREAD];
      for (int i = 0; i < NUMTHREAD; i++) {
         runners[i] = new MyThread(i);
//       runners[i].setPriority(NORM_PRIORITY+i); // (2)
      }
      for (int i = 0; i < NUMTHREAD; i++) runners[i].start();
   }
}
```

Lock rientranti

```
public class Rientrante {
   public synchronized void a() {
      b();
      System.out.println("Sono in a()");
   }
   public synchronized void b() {
      System.out.println("Sono in b()");
   }
   public static void main(String[] args) {
      Rientrante r = new Rientrante();
      r.a();
   }
}

// output
// Sono in b()
// Sono in a()
```


Il linguaggio Java

Programmi d'esempio

Produttori e consumatori

Produttore e consumatore (I)

```
// Il produttore

public class Consumatore extends Thread {
   private Buffer buffer;
   private int number;

   public Consumatore(Buffer b, int numero){
      buffer = b;
      this.number = numero;
   }

   public void run() {
      int valore = 0;
      for (int i = 0; i < 10; i++) {
         valore = buffer.get();
         System.out.println("Consumatore #" + this.number +
                  " got: " +  valore);
      }
   }
}
```


Produttore e consumatore (II)

```
// Il consumatore

public class Produttore extends Thread {
   private Buffer buffer;
   private int numero;

   public Produttore(Buffer b, int numero) {
      buffer = b;
      this.numero = numero;
   }

   public void run() {
      for (int i = 0; i < 10; i++) {
         buffer.put(i);
         System.out.println("Produttore #" + this.numero +
                  " put: " + i);
      }
   }
}
```

Produttore e consumatore (III)

```
// Il buffer
public class Buffer {
   private int valore;
   private boolean disponibile = false;

   public syncronized int get() {
      int v;
      while (disponibile == false)
         try {
            wait();
         } catch(InterruptedException e){}
      disponibile = false;
      v = valore;
      notifyAll();
      return v;
   }

   public synchronized void put(int value) {
      while (disponibile == true)
         try {
            wait();
         } catch(InterruptedException e){}
      valore = value;
      disponibile = true;
      notifyAll();
   }
}
```


Il linguaggio Java

Programmi d'esempio

Gestore di un pool di risorse equivalenti

Gestore (1)

```
public class Gestore {

  // numero totale di risorse
  private final int N;
  // se risorse[i] = true, la risorsa i è disponibile
  private boolean[] risorse;
   // numero delle risorse disponibili
  private int disponibili;

  public Gestore(int numRisorse) {
     N = numRisorse;
     disponibili = N;
     risorse = new boolean[N];
     for (int i = 0; i < N; i++) risorse[i] = true;
  }

  public Gestore() {this(2);}

  // continua
```


Gestore (2)

```
  synchronized int richiesta() {
     if (disponibili <= 0)
        try {
           wait();
        } catch(InterruptedException e){}
     int i = 0;
     while ((i < N) && !risorse[i]) i++;
     risorse[i] = false;
     disponibili--;
     System.out.println("Alloco risorsa " + i);
     return i;
  }

  synchronized void rilascio(int i) {
     System.out.println(“Rilascio risorsa " + i);
     risorse[i] = true;
     disponibili++;
     notify();
  }
} // class
```

Gestore (3)

```
class MioThread extends Thread {
   final int TIMES = 2;
   Gestore g;

   public MioThread(String name, Gestore gest) {
      super(name);
      g = gest;
   }
   public void run() {
      int r;
      for (int i = 0; i < TIMES; i++) {
         r = g.richiesta();
         try{
            sleep((long)(Math.random()*1000));
         } catch(InterruptedException e){}
         g.rilascio(r);
      }
   }
}
////////////////////////////////////////////////
public class DriverGestore {
   public static void main(String[] args) {

      final  int NUMTHREADS = 3;
      final  int NUMRISORSE = 2;


      Gestore g = new Gestore(NUMRISORSE);
      for (int i = 0; i < NUMTHREADS; i++)
         new MioThread("thread[" + i + "]", g).start();

   }
}
```


Il linguaggio Java

Programmi d'esempio

Buffer di lunghezza limitata

Buffer (I)

```
public class Buffer { // FIFO

  private Object[] buffer; // il buffer
  private final int SIZE;  // capacità del buffer
  private int testa; // punto di inserimento
  private int coda; // punto di estrazione
  private int cont; // num oggetti presenti nel buffer

  public Buffer() {
    this(10);
  }

  public Buffer(int sz) {
    SIZE = sz;
    buffer = new Object[SIZE];
    cont = 0; testa = 0; coda = testa;
  }

  synchronized void put(Object elem) {
    while (cont >= SIZE)
      try {
        wait();
      } catch(InterruptedException e){}
      buffer[testa] = elem;
      testa = (testa + 1) % SIZE;
      cont++;
      notifyAll();
  }

  synchronized Object get() {
    Object elem;
    while (cont <= 0)
      try {
        wait();
      } catch(InterruptedException e){}
    elem = buffer[coda];
    coda = (coda + 1) % SIZE;
    cont--;
    notifyAll();
    return elem;
  }
}
```


Buffer (II)

```
class Produttore extends Thread {
  final int TIMES = 2;
  Buffer buf;

  public Produttore(String name, Buffer b) {
    super(name);
    buf = b;
  }
  public void run() {
    int r;
    for (int i = 0; i < TIMES; i++) {
      String s = getName() + ": string " + i;
      System.out.println(s);
      buf.put(s);
      try{
        sleep((long)(Math.random()*1000));
      } catch(InterruptedException e){}
    }
  }
}
```

Buffer (III)

```
class Consumatore extends Thread {
  final int TIMES = 2;
  Buffer buf;

  public Consumatore(String name, Buffer b) {
    super(name);
    buf = b;
  }
  public void run() {
    int r;
    for (int i = 0; i < TIMES; i++) {
      String s;
      s = (String)buf.get();
      System.out.println(getName() +
                         " consuma " + s);
      try{
        sleep((long)(Math.random()*1000));
      } catch(InterruptedException e){}
    }
  }
}
```


Buffer (IV)

```
public class DriverBuffer {
  public static void main(String[] args) {

  final int NUMPROD = 3;
  final int NUMCONS = 3;

  Buffer buf = new Buffer();

  for (int i = 0; i < NUMCONS; i++)
    new Consumatore("cons[" + i + "]",
                                    buf).start();
  for (int i = 0; i < NUMPROD; i++)
    new Produttore("prod[" + i + "]",
                                    buf).start();
  }
}
```

Il linguaggio Java

Programmi d'esempio

Lettori e scrittori

Scrittore

```
class Scrittore extends Thread {
  final int TIMES = 5;
  ReadWrite rw;

  public Scrittore(String name,
                                  ReadWrite rw) {
    super(name);
    this.rw = rw;
  }
  public void run() {
    for (int i = 0; i < TIMES; i++) {
      String name = getName();
      rw.beginWrite();
      System.out.println(name +
                          " begin write");
      try{
        sleep((long)(Math.random()*1000));
      } catch(InterruptedException e){}
      System.out.println(name +
                          " end write");
      rw.endWrite();
    }
  }
}
```

Lettore

```
class Lettore extends Thread {
  final int TIMES = 3;
  ReadWrite rw;

  public Lettore(String name,
                              ReadWrite rw) {
    super(name);
    this.rw = rw;
  }

  public void run() {
    String name = getName();
    for (int i = 0; i < TIMES; i++) {
      rw.beginRead();
      System.out.println(name +
                         " begin read");
      try{
        sleep((long)(Math.random()*1000));
      } catch(InterruptedException e){}
      System.out.println(name +
                         "   end read");
      rw.endRead();
    }
  }
}
```


DriverReadWrite

```
public class DriverReadWrite {
  public static void main(String[] args)
                    throws InterruptedException {

    final  int NUMREADERS = 2;
    final int NUMWRITERS = 2;
    Thread[] t =
      new Thread[NUMREADERS + NUMWRITERS];
    ReadWrite rw = new ReadWrite();

    System.out.println("MAIN: BEGIN");
    for (int i = 0; i < NUMREADERS; i++) {
      t[i] =
        new Lettore("lettore[" + i + "]", rw);
      t[i].start();
    }
    for (int i = 0; i < NUMWRITERS; i++) {
      t[i + NUMREADERS] =
        new Scrittore("scrittore[" + i + "]",rw);
      t[i + NUMREADERS].start();
    }
    for (int i = 0; i < NUMREADERS+NUMWRITERS;
                                          i++)
      t[i].join();
    System.out.println("MAIN: END");
  }
}
```

ReadWrite (I)
(starvation sui lettori)

```
public class ReadWrite {

  private int aw = 0; // num active writers
  private int rr = 0; // num running readers
  // a writer is in
  private boolean busy_writing = false;

  public ReadWrite() {}

  synchronized void beginRead() {
    while (aw > 0)
      try {
        wait();
      } catch(InterruptedException e){}
    rr++;
  }

  synchronized void endRead() {
    rr--;
    notifyAll();
    }

  synchronized void beginWrite() {
    aw++;
    while (busy_writing || (rr > 0))
      try {
        wait();
      } catch(InterruptedException e){}
    busy_writing = true;
  }

  synchronized void endWrite() {
    busy_writing = false;
    aw--;
    notifyAll();
  }
}
```


ReadWrite (II)
(in assenza di starvation)

```
public class ReadWrite {

    private boolean okReader = true; // readers can proceed
    private boolean okWriter = true; // a writer can proceed
    private int aw = 0; // active writers
    private int rr = 0; // running readers
    private int ar = 0; // active readers
    private boolean busy = false; // a writer is in

public ReadWrite() {}
    synchronized void  beginRead() {
        ar++;
        while ((aw > 0) && okWriter)
            try {wait();} catch(InterruptedException e){}
        ar--;
        rr++;
    }

    synchronized void endRead() {
        rr--;
        if (aw > 0) {
            okWriter = true;
            okReader = false;
        }
        if (rr == 0) notifyAll();
    }

    synchronized void beginWrite() {
        aw++;
        while (busy || (rr > 0) || ((ar > 0) && okReader))
            try {wait();} catch(InterruptedException e){}
        busy = true;
    }

    synchronized void endWrite() {
        busy = false;
        aw--;
        if (ar > 0) {
            okReader = true;
            okWriter = false;
        }
        notifyAll();
    }
}
```

Semaforo

```
public class Semaphore {
   private int s;
 public Semaphore(int v) {
        s = v;
 }

 // a mutex by default
 public Semaphore() { this(1); }

 synchronized public void p() {
   while (s <= 0)
     try{
      wait();
     } catch(InterruptedException e){}
   s--;
 }

 synchronized public void v() {
   s++;
   notifyAll();
 }
}
```


ReadWrite (III)
(assenza di starvation)

```
public class ReadWrite {
   private int ar = 0; // active readers
   private boolean aw = false; // active writers
   private int br = 0; // blocked readers
   private int bw = 0; // blocked writers
   private Semaphore rs = new Semaphore(0); // priv. sem.
   private Semaphore ws = new Semaphore(0); // priv. sem.

   public void beginRead() {
      synchronized(this) {
         if (!aw && (bw == 0)) {
            rs.v();
            ar++;
         } else br++;
      }
      rs.p(); // potentially blocking; outside of mutex
   }
   synchronized public void endRead() {
      ar--;
      if ( ar == 0 && bw > 0) {
         aw = true;
         bw--;
         ws.v();
      }
   }

// continua
```

ReadWrite (III)
(assenza di starvation)

```
public void beginWrite() {
      synchronized(this){
         if (ar == 0 && !aw) {
            ws.v();
            aw = true;
         }
         else bw++;
      }
      ws.p(); // potentially blocking; outside of mutex
   }
   synchronized public void endWrite() {
      aw = false;
      if (br > 0)
         while (br >0) {
            br--;
            ar++;
            rs.v();
         }
      else if (bw > 0) {
         aw = true;
         bw--;
         ws.v();
      }
   }

} // class ReadWrite
```


Il linguaggio Java

Programmi d'esempio

Esempio di server multithread

Il server multi-thread

```
import java.io.*;
import java.net.*;

class ServerThread extends Thread {
  private Socket socket;
  private BufferedReader in;
  private PrintWriter out;

  public ServerThread(Socket s)
                              throws IOException {
    socket = s;
    in = new BufferedReader(
           new InputStreamReader(
             socket.getInputStream()));
    out = new PrintWriter(new BufferedWriter(
             new OutputStreamWriter(
              socket.getOutputStream())), true);
    start();
  }

  public void run() {
    try {
      while (true) {
         String str = in.readLine();
        if ( str.equals("END") ) break;
        System.out.println("Echo: " + str);
          out.println(str);
      }
    } catch(IOException e) {
    } finally {
      try{
        socket.close();
      } catch(IOException e) {}
    }
  }
} // continua
```


Il server multi-thread

```
public class Server {
  static final int PORT = 8080;

  public static void main(String[] args)

    throws IOException {
    ServerSocket s =

    new ServerSocket(PORT);
    try {
      while ( true ) {
         Socket socket = s.accept();
         try {
           new ServerThread(socket);
         } catch(IOException e) {
           socket.close();
         }
      }
    } finally {
      s.close();
    }
  }
}
```

Il linguaggio Java

Programmi d'esempio

I timer task

TimerTask

```
import java.util.Timer;
import java.util.TimerTask;

public class Reminder {
  class RemindTask extends TimerTask {
    public void run() {
      System.out.println("E' l'ora!");
      timer.cancel();
    }
  }
  Timer timer;
  public Reminder(int seconds) {
    timer = new Timer(); // creazione timer
    // creazione e scheduling di un timer thread
    timer.schedule(new RemindTask(), seconds*1000);
  }

  public static void main(String args[]) {
    System.out.println("Sto per schedulare il task.");
    new Reminder(5);
    System.out.println("Task schedulato.");
  }
} // class Reminder
```

ReminderBeep

```
import java.util.Timer;
import java.util.TimerTask;
import java.awt.Toolkit;

public class ReminderBeep {
  Toolkit toolkit;
  Timer timer;
  public ReminderBeep(int seconds) {
    toolkit = Toolkit.getDefaultToolkit();
    timer = new Timer();
    timer.schedule(new RemindTask(),
                                seconds*1000);
  }
  class RemindTask extends TimerTask {
    public void run() {
      System.out.println("E' l'ora");
      toolkit.beep();
       //timer.cancel(); // JVM non termina
      System.exit(0);
    }
  }
  public static void main(String args[]) {
    System.out.println("Sto per schedulare
                                  il task.");
    new ReminderBeep(5);
    System.out.println("Task schedulato.");
  }
}
```


Esecuzione periodica di un thread

```
import java.util.Timer;
import java.util.TimerTask;

public class PeriodicReminder {
  Timer timer;
  public PeriodicReminder() {
    timer = new Timer();
    timer.schedule(new RemindTask(),
                        0, //ritardo iniziale
                        1*1000);  //periodo
  }
  class RemindTask extends TimerTask {
    int numVolte = 10;
    public void run() {
      if (numVolte > 0) {
        System.out.println("E' l'ora");
        numVolte--;
      } else {
        System.out.println("E' l'ora");
        timer.cancel();
      }
    } // run
  } // RemindTask
  // main
}
```