Geom. Giofrè
Vincenzo Pasquale

java2

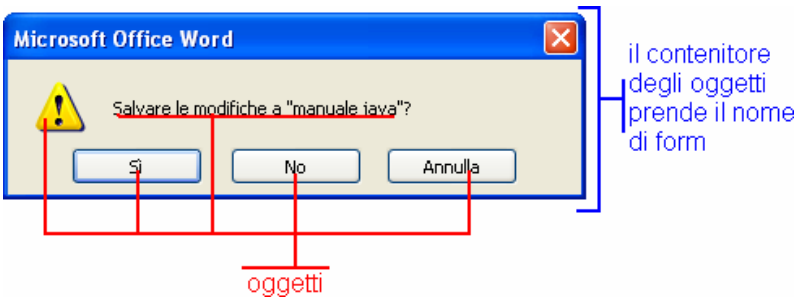
Il manuale per chi vuole imparare velocemente

Presentazione

Il linguaggio **Java** nasce nel 1995 dai programmatori della Sun Microsystem. La sua popolarità è dovuta al fatto che può girare su macchine diverse, ovvero e "Multi piattaforma", si può usare su ***Windows***, ***Linux*** , ***Mac OS*** e tanti altri sistemi operativi. La programmazione in java copre molti campi, dalla classica programmazione per computer a quella più compatta per i dispositivi mobili come i cellulari. Java inoltre nasce come linguaggio **OOP** "***O****rientato alla* ***P****rogrammazione ad* ***O****ggetti*".

Gli oggetti

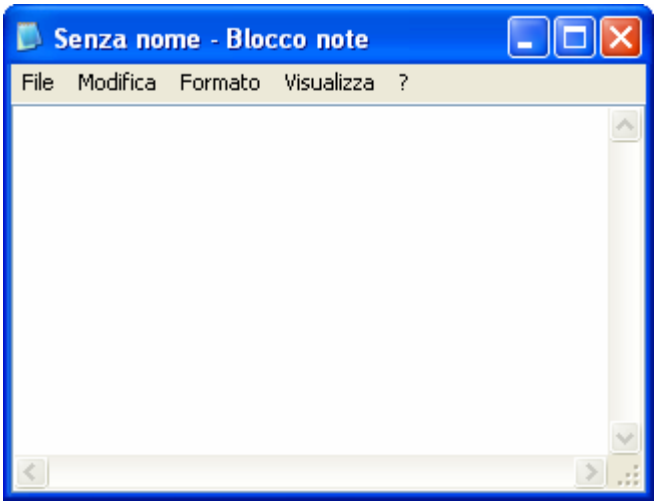
Gli oggetti che intendiamo in programmazione sono del tutto simili aglio oggetti che ci circondano nella realtà. Se prendiamo in esame un semplice messaggio che ci mostra il computer, vedremo che:



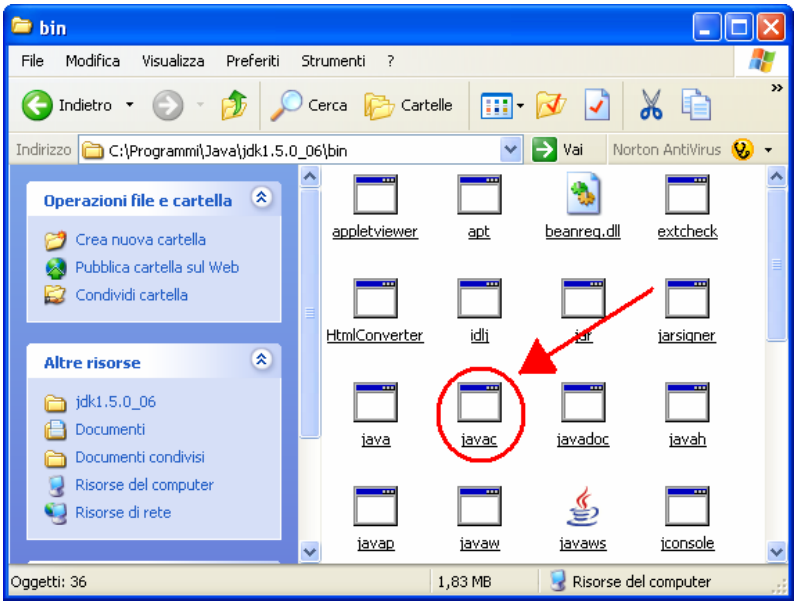
Editor e Compilatore

Bisogna fare la distinzione fondamentale fra Editor e Compilatore.

- Un ***editor*** può essere anche il classico ***Block Notes***, il ***TexPad***, il ***JCreator*** o qualsiasi altro programma che consente di scrivere testo. Un editor non è Java, e solo con un editor non si può programmare in java.



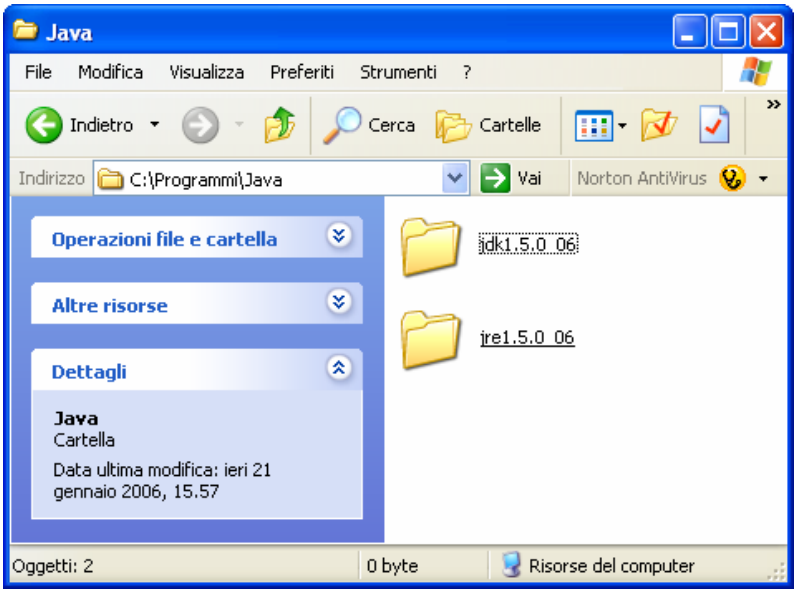
- Il ***compilatore*** invece è quel programma che trasforma il nostro codice, scritto in un editor, in linguaggio macchina e nel caso di java il compilatore e il programma ***javac*** che si trova nel ***jdk***. Il compilatore di java trasforma il codice in bitecode.



Jdk e jre

Non su tutti i computer è installato java, per esempio Windows XP non lo implementa più. Un'altra distinzione da fare e tra jdk e jre.

- Il ***jdk*** è l'insieme dei programmi che servono per poter programmare in java.
- Il ***jre*** è solo la virtual machine, ovvero serve solo per vedere i programmi fatti in java.



Per cui se sul vostro computer è installato solo il jre, non potete programmare in java.

Trovare il jdk

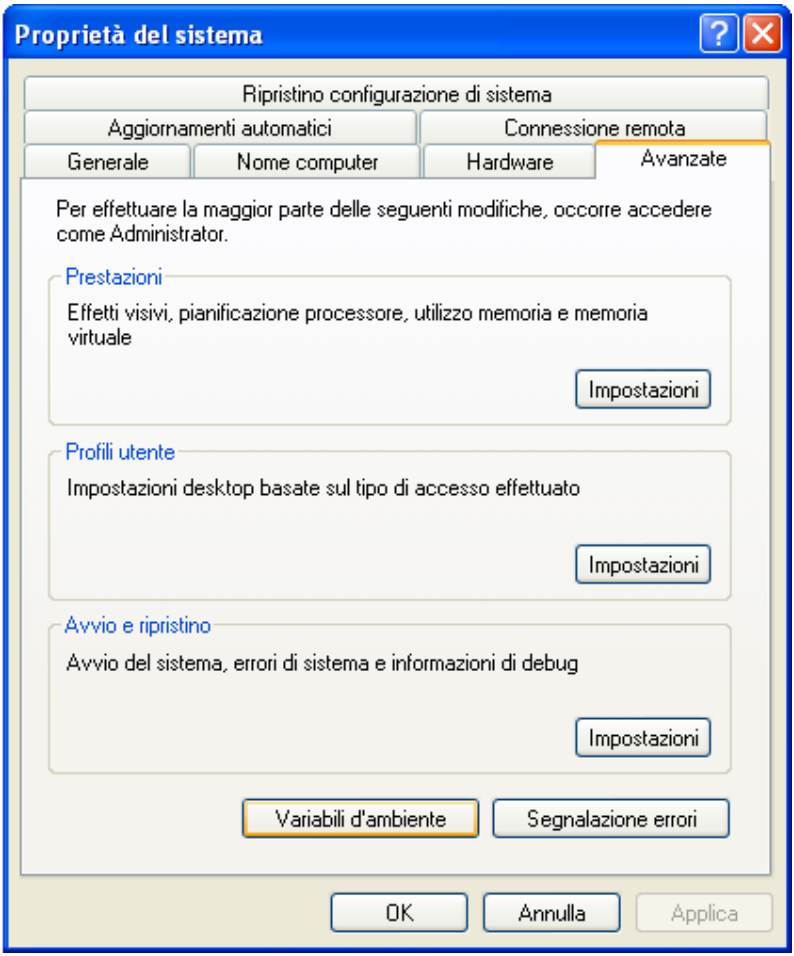
Trovare il jdk è semplice, basta andare sul sito della Sun Microsystem o più semplicemente in un qualsiasi motore di ricerca digitate "download jdk", e ricordatevi di scaricare sempre la versione più recente.

Configurare il jdk

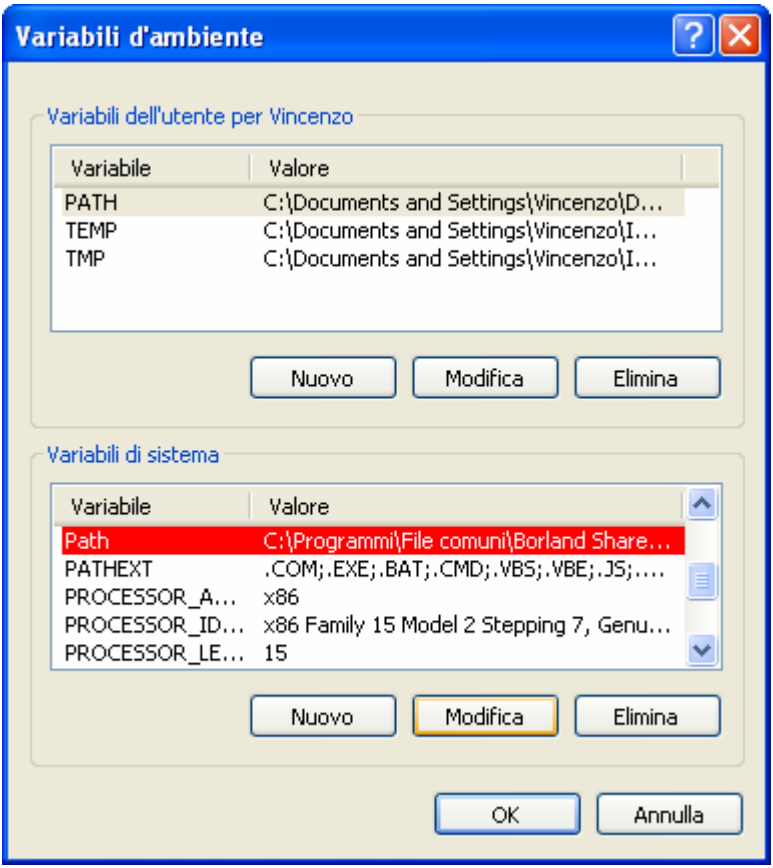
Se utilizzate Windows, dopo aver scaricato (*si trova*

anche sui CD-Rom delle riviste di programmazione) il ***jdk***, installatelo seguendo la procedura guidata. Da notare che quando installate il jdk, installate anche automaticamente anche il jre. Dopo l'installazione ciccate col tasto destro su ***risorse del computer*** e cliccare sulla voce ***proprietà***.

Nella nuova schermata, selezionare la voce ***avanzate*** e cliccare sul pulsante ***variabili d'ambiente.***



Nella sezione ***variabili di sistema*** selezionate la voce ***Path*** che nell'immagine è evidenziata in rosso.



All'inizio del campo ***Valore variabile***, inserite il percorso del jdk terminante con il "**;**", nell'esempio in questione, il percorso è

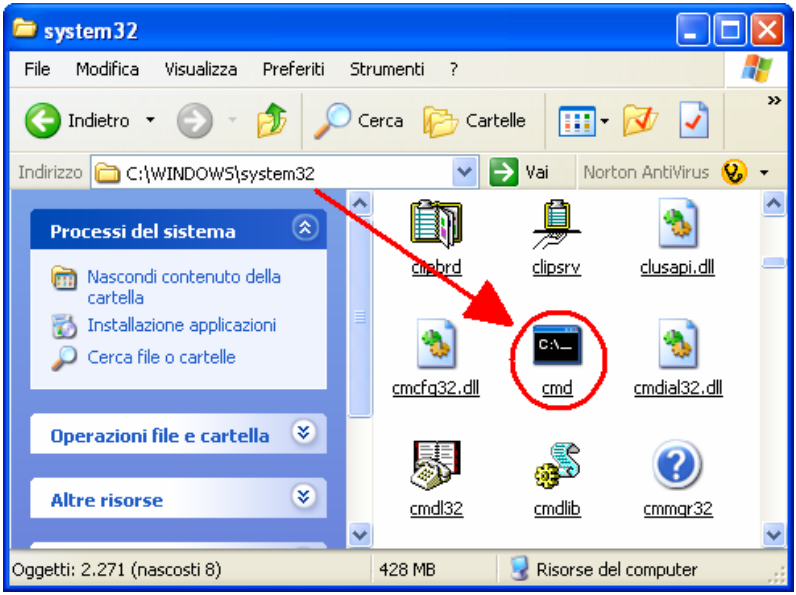
C:\Programmi\Java\jdk1.5.0_06\bin;

Sul vostro computer, la cartella dovrebbe essere quasi la stessa, potrebbero cambiare solo i numeri di versione del jdk, ma l'importante è che il sistema

trovi la cartella ***bin*** del jdk. Ed infine cliccate sul pulsante ***ok***

Il comando di linea

Per compilare e vedere i vostri file java, avrete bisogno di quella piccola applicazione che si chiama ***CMD***, e si trova nella cartella ***C:\Windows\System32***, cercatela e copiatela sul desktop, o nella cartella dove intendete lavorare con i vostri file java.



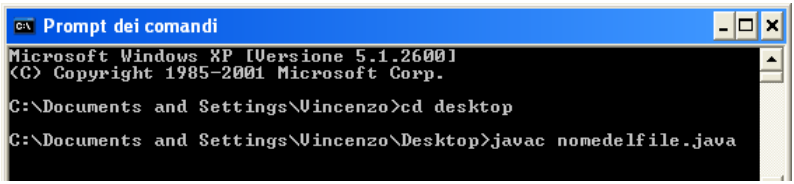
Compilazione / esecuzione

Per ***compilare*** i vostri file java, quelli salvati con estensione ***.java***, sarà sufficiente copiare il ***CMD***

nella cartella dove ci sono questi file, aprirlo e digitare:

javac nomedelfile.java

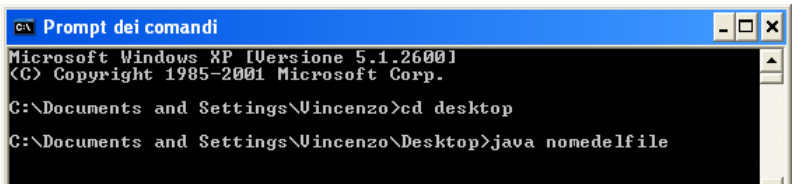
e se l'operazione è andata a buon fine, ovvero se il compilatore non riscontrerà errori, creerà un nuovo file, con estensione ***.class***, ovvero il bitecode.



Per ***eseguire*** un'applicazione appena creata o che ci è stata data da un amico, sarà sufficiente copiare il ***CMD*** nella cartella dove c'è il file ***.class***, aprirlo e digitare:

java nomedelfile

senza applicare nessuna estensione al file .class.



Input e Output

Questa è la parte più importante da capire, perché serve per interagire direttamente con l'utente. Immaginate che sul vostro schermo venga stampata la scritta:

inserisci il primo numero

all'ora questo insieme di parole formano l'***output***, quindi possiamo dire che l'***output*** sono tutte quelle cose che il computer ci mostra sullo schermo, siano parole o immagini, ***o meglio che il programmatore decide di farci vedere***. Se prendiamo in esame un altro esempio che ci chiede di inserire il nostro nome:

inserisci il tuo nome ß output
Vincenzo ß input

Quindi possiamo dire che l'***input*** sono tutte quelle richieste che ci fa il computer, nell'esempio precedente, il computer stampa sullo schermo l'***output*** "*inserisci il tuo nome*" e richiede come l'***input*** *il nostro nome*.

Variabili

Per spiegare cosa sono le variabili, prendiamo in esame un altro esempio

```
Int x = 3;
Int y = 2;
Int R = x + y;
```

in questo caso la parola “Int” rappresenta il tipo di variabile, in questo caso rappresenta un numero intero, mentre x, y, R rappresentano le variabili, a cui noi associamo i valori.

Tipi di Variabili

Nome	da	a	rappresenta
Boolean	False	True	Vero o falso
Char	Unicode	Unicode	Tutti i caratteri della tastiera come testo
String	Unicode	Unicode	Uguale a Char ma migliore
Byte	-128	128	Numeri interi
Short	-2^15	2^15	Numeri interi
Int	-2^31	2^31	Numeri interi
Long	-2^63	2^63	Numeri interi
Float	-IEE754	IEE754	Numeri decimali
Double	-IEEE754	IEEE754	Numeri decimali
BigInteger	-------	-------	Numeri grandi interi
BigDecimal	-------	-------	Numeri grandi decimali
File	-------	-------	Si usa per le operazioni con i file

Da notare che se impostiamo una variabile di tipo testo, il computer riconoscerà anche i numeri come se fossero solo del testo, per esempio:

```
String x = “3”;
String y = “2”;
String r = x + y;
```

Allora il valore assegnato a r non sarà la somma dei numeri e quindi uguale a “5”, ma la soma dei caratteri e quindi uguale a “32”.

Commenti

Il commento non è altro che una semplice annotazione che serve al programmatore per ricordare qualcosa.

```
// Commento su una riga

/* Commento
        Su più
        Righe */
```

Da notare che il commento non viene letto dal compilatore, anzi lo ignora.

Librerie

Le librerie sono una serie di procedure semplificate, che racchiudono centinaia di funzioni tipo la classica "System.out.println("");", e si richiamano all'inizio della nostra applicazione tramite il comando:

```
import nomelibreria.*;
```

l'esempio classico e:

```
import java.util.*;
```

Modello di applicazione

```
public class nomedelprogramma
{
        public static void main(String[] args)
        {
                /* qui ci va il codice della nostra
                applicazione */
        }
}
```

Possiamo ipotizzare il programma racchiuso in due contenitori, uno dentro l'altro, rappresentati dai simboli **{ }.** Sul primo contenitore, quello esterno viene riportato il ***nomedelprogramma***, che va scritto tutto attaccato, il secondo contenitore, il ***main***, racchiude il codice della nostra applicazione.

Output

Se abbiamo intenzione di far visualizzare al computer una scritta o una variabile sullo schermo, dobbiamo utilizzare questo comando:

System.out.println(“Testo che vogliamo vedere”);

oppure posiamo usare una forma composta:

```
Int x = 3;
Int y = 2;
Int R = x + y;
System.out.println(“R vale = ” + R + “ metri ”);
```

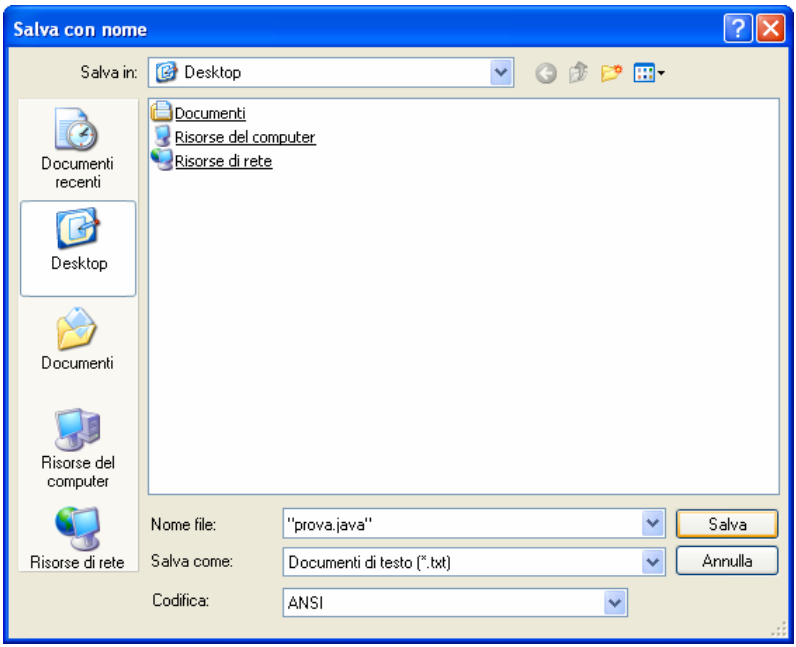
Come si crea un programma

Inizieremo creando un semplice programma che visualizza una scritta sullo schermo. Per prima cosa, aprire il Block Notes e scrivi:

```
import java.util.*;
public class prova
{
   public static void main(String[] args)
    {
      System.out.println("Questa e un’applicazione");
    }
}
```

dove import java.util.*; e la libreria che ci serve, prova è il nome del nostro programma. Da ricordare che in java ogni riga deve terminare col "**;**" eccetto alcune righe predefinite. Ora ***salviamo*** il file sul desktop col nome del nostro programma, aggiungendo le virgolette e l'estensione ***.java***:

"prova.java"



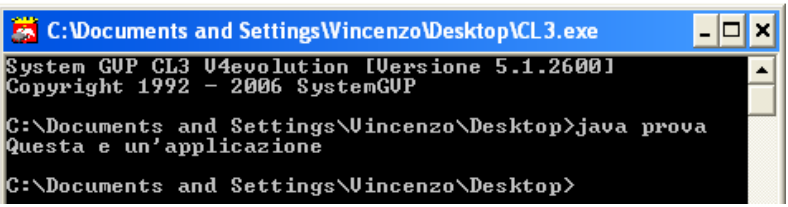
Copiamo il ***CMD*** sul desktop, apriamolo e scriviamo:

javac prova.java

e se l'operazione è andata a buon fine, ovvero se il compilatore non riscontrerà errori, creerà un nuovo file ***prova.class***, ovvero il bitecode. Per ***eseguire*** l'applicazione appena creata è sufficiente aprire il ***CMD*** e digitare:

java prova

il risultato dovrebbe essere:



Input

Se vogliamo che l'utente inserisca una variabile, numerica o testo, per prima cosa dobbiamo impostare lo scanner della variabile:

Scanner in = new Scanner(System.in);

Questa riga nasce con la versione 1.5 o 5 di java, ed è una serie di procedure semplificate. Va inserita una volta sola, indipendentemente da quante variabili richiediamo.
Per richiede un input di tipo numerico intero digitiamo:

Int nomedellavariabile = in.nextInt();

Per richiede un input di tipo numerico decimale digitiamo:

```
Double nomedellavariabile = in.nextDouble();
```

Per richiede un input di tipo testo digitiamo:

```
String nomedellavariabile = in.nextLine();
```

Da notare che queste operazioni hanno bisogno della libreria import java.util.*; Un piccolo esempio di come usare le variabili:

```
import java.util.*;
public class prova
{
   public static void main(String[] args)
    {
      Scanner in = new Scanner(System.in);

      System.out.println("Inserisci il primo numero");
      Int x = in.nextInt();
      System.out.println("Inserisci il secondo numero");
      Int y = in.nextInt();

      System.out.println("Inserisci il tuo nome");
      String nome = in.nextLine();

      //questo è un commento
      Int R = x + y;
      System.out.println("Ciao " + nome " R = " + R);
    }
}
```

Condizioni if/else

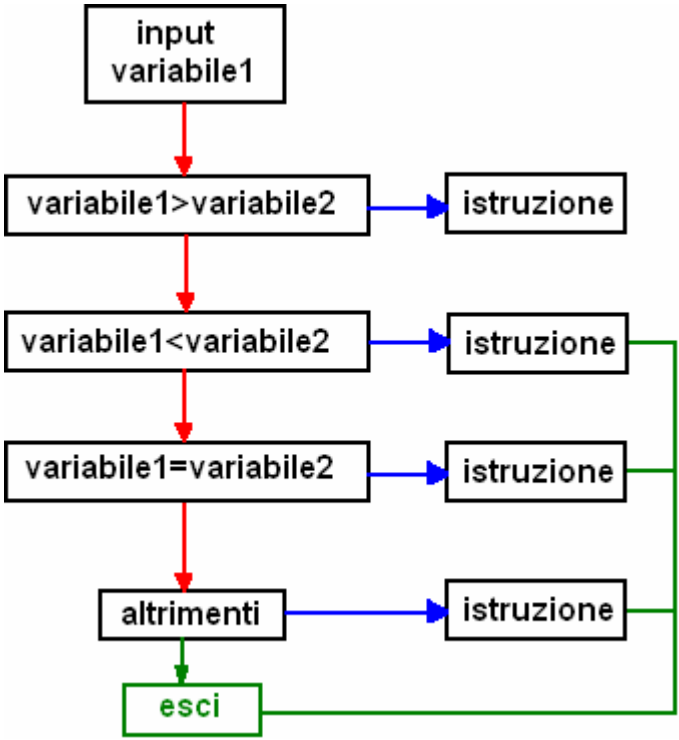
Le parole if e else significano se e altrimenti, ovvero se vogliamo realizzare una condizione potremo usarle in modi diversi:

```
if (variabile1 = variabile2)
 { //serie di istruzioni}
else
 { //altra serie di istruzioni}
```

Altrimenti se ci sono più condizioni:

```
if (variabile1 > variabile2)
 { //serie di istruzioni}
else if (variabile1< variabile2)
 { //serie di istruzioni}
else if (variabile1= variabile2)
 { //serie di istruzioni}
else
 { //altra serie di istruzioni}
```

Ogni buon programmatore, prima di usare questa istruzione (*anche se converrebbe farlo prima di creare una qualsiasi applicazio*ne), si crea un flowcart ovvero un albero decisionale:



Esempio di if/else

Vi siete mai chiesti come funziona la schermata di Windows quando vi cerca la password? Tutte le applicazioni che cercano un codice o una password funzionano grazie a una semplice istruzione if/else con quattro righe di codice:

```
import java.util.*;
public class esempioif
{
```

```
   public static void main(String[] args)
{
 String codice = “123ABC”;

 Scanner in = new Scanner(System.in);

 System.out.println(“inserisci la password”);
 String password = in.nextLine();

  if (password = codice)
    { System.out.println(”Password corretta”);
    }
  else
    { System.out.println(”Password non corretta”);
    }
}
}
```

Cicli For

Se vogliamo ripetere un’istruzione per un certo numero di volte possiamo utilizzare l’istruzione ***for***. Se ad esempio vogliamo stampare sullo schermo per ***n*** volte la parola ***ciao*** scriveremo:

```
for (int i = 1;  i <= n;  i++)
 { System.out.println(”Ciao”);}
```

- L’istruzione int i = 1; definisce una variabile che si chiama i e che vale 1.

- L’istruzione i <= n; imposta che la variabile i deve arrivare ad un numero massimo di n dove n deve essere un numero intero, esempio 23
- L’istruzione i++ imposta che la variabile i deve aumentare da 1 a n di una unità per volta.

Operazioni con i numeri grandi

Le operazioni con i numeri eccessivamente grandi non sono simili a quelle con i numeri piccoli, per esempio se impostiamo ipoteticamente:

```
BigInteger x = 10000000000000000000000000;
BigInteger y = 10000000000000000000000000;
```

Le operazioni possibili sono

```
BigInteger R = x.add(y); // R= x + y
BigInteger R = x.subtract(y); // R= x - y
BigInteger R = x.multiply(y); // R= x * y
BigInteger R = x.divide(y); // R= x : y
```

Le stesse operazioni sono possibili con
BigDecimal

Funzioni matematiche

Molte volte nelle nostre applicazioni dobbiamo ricorrere a funzioni di tipo matematico o trigonometrico, queste possono essere utilizzate con ***Math***:

```
Math.sqrt(variabile);      //radice quadrata
Math.sinvariabile);        //seno
Math.cos(variabile);       //coseno
Math.tan(variabile);       //tangente
Math.atan(variabile);      //arco tangente
Math.atan2(variabile);
Math.exp(variabile);       //esponenziale
Math.log(variabile);       //logaritmo
Math.PI;                   //valore di pigreco
Math.E;                    //valore di e
```

Da notare che con Math bisogna usare variabili di tipo double, per esempio

```
Double x = 4;
double y = Math.sqrt(x);
System.out.println(y);
```

L'alternativa a if/else

Molte volte la funzione if/else può risultare scomoda, quando si ha a che fare con delle selezioni multiple che prevedono diverse alternative, in questo caso possiamo utilizzare il comando ***switch*** associato a ***case n***, per esempio:

```
import java.util.*;
public class esempioswitch
{
 public static void main(String[] args)
{
 Scanner in = new Scanner(System.in);

 System.out.println("scegli l'opzione");
 Int scelta = in.nextInt();

  switch (scelta)
    {
       CASE 1;
          System.out.println("hai scelto l'opzione 1");
          Break;   //chiude l'istruzione
       CASE 2;
          System.out.println("hai scelto l'opzione 2");
          Break;   //chiude l'istruzione
       CASE 3;
          System.out.println("hai scelto l'opzione 3");
          Break;   //chiude l'istruzione
    }
}
}
```

Per scrivere su un file

Se vogliamo salvare una variabile o una stringa su un file dobbiamo includere la libreria java.io.*; e ricordarci di inserire le parole ***Throws IOException*** al contenitore ***Main***, nell'esempio viene salvata la variabile ***testo*** come testo scritto nel file ***filediuscita.txt***:

```
import java.util.*;
public class esempiotxt
{
public static void main(String[]args)Throws IOException
{
        File FileOutput = new File("filediuscita.txt");
        File Writer out = new FileWriter(FileOutput);

        String testo = "Testo scritto nel file";

        out.write(testo);
        out.close();
} }
```

Info di sistema

Molte volte abbiamo la necessità di richiamare delle informazioni di sistema, come la versione del sistema operativo o il nome dell'utente che lo sta utilizzando. Per farlo possiamo impostare una

variabile di tipo string a ricevere queste informazioni.

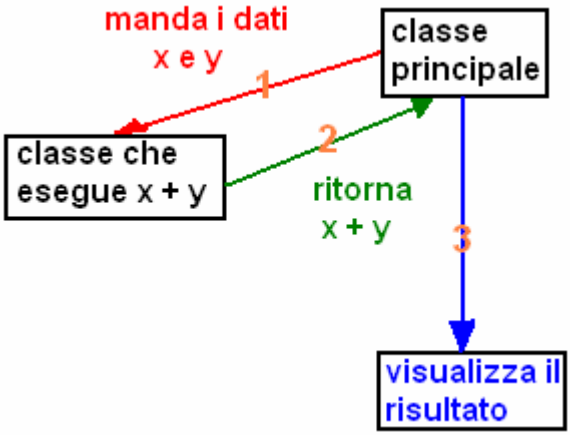
```
String variabile = System.getProperty("tipo.info");
System.out.println(variabile);
```

Una tabbella riassuntiva dei richiami possibili:

tipo info	richiamo
File.separator	Simbolo utilizzato come separatore
Java.class.path	Percorso di classe
Java.class.version	Versione della classe
Java.home	Cartella dov'è installato java
Java.vendor	Nome del produttore
Java.vendor.url	Indirizzo URL del produttore
Java.version	Versione di java
Line.separator	Separatore di riga
Os.arch	Architettura del Sistema Operativo
Os.name	Nome del Sistema Operativo
Os.version	Versione del Sistema Operativo
Path.separator	Separatore utilizzato nelle stringhe
User.dir	Cartella di lavoro corrente
User.name	Nome dell'utente
User.home	Cartella dell'utente

Classi

In un'applicazione possiamo creare più classi o richiamare classi esterne e farle interagire fra di loro. Se ipotizziamo una classe principale che per fare x + y ha bisogno di una istruzione contenuta in un'altra classe:



Se ipotizziamo una classe principale ***classe1:***

```
public class classe1
{
        public static void main(String[] args)
        {
                /* qui ci va il codice della nostra prima
                classe */
        }
}
```

E una seconda classe scritta nello stesso file:

```
class classe2
{
        Public String richiama_funzione;

        public void funzione()
        {
                /* qui ci va il codice della nostra seconda
                classe*/
        }
}
```

Per richiamare nella ***classe1*** la funzione ***funzione()*** descritta nella ***classe1***, abbiamo bisogno della variabile "***Public String richiama_funzione;"*** che viene descritta nella ***classe2***. Nella classe principale ***classe1,*** per richiamare la funzione ***funzione(),*** bisognerà richiamare prima la ***classe2*** e dopo la ***variabile di richiamo*** :

```
Classe2 richiama_funzione = new classe2();
richiama_funzione.funzione();
```

Se invece vogliamo richiamare una funzione descritta in un altro file di classe, basterà aggiungere il nome del file fra le librerie import nomedelfile.*; e richiamare la funzione come precedentemente descritto.

Un breve esempio di questo utilizzo è il seguente:

```
import java.util.*;

public class scrittura
{
	public static void main(String[] args)
	{
		System.out.println("Inserisci il tuo nome");
		System.out.println();

		classe2 richiamo_dati = new classe2();
		richiamo_dati.dati();
	}
}

class classe2
{
	public String richiamo_dati;
	public void dati()
	{
		Scanner in = new Scanner(System.in);
		String nome = in.nextLine();
		System.out.println();
		System.out.println("Ciao " + nome);
	}
}
```

Note

Se volete utilizzare un buon editor che funge anche da compilatore e runnig, uno buono è il JCreator, da fare molta attenzione perché alcuni degli esempi che sono in questo manuale sono testati con il jdk1.5.0_06 e non e certo il loro funzionamento se il vostro jdk è più vecchio. Al momento che scrivo, il la versione di java disponibile è:


```
Versione 1.5.0 (build 1.5.0_06-b05)
Copyright 2005 Sun Microsystems, Inc.
Tutti i diritti riservati. L'uso è soggetto ai
termini della licenza.
Per ulteriori informazioni sulla tecnologia Java e
per informazioni sulle applicazioni Java disponibili,
visitare il sito http://www.java.com
```