Introduzione a tool per lo sviluppo e l'esecuzione di programmi in linguaggio Java

Indice

Introduzione all'ambiente di sviluppo
- Compilazione ed esecuzione di un programma
- Compilazione ed esecuzione di un programma usando NetBeans IDE

Esercizi

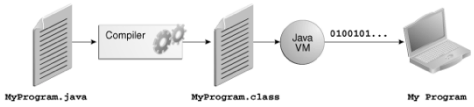
Compilazione ed esecuzione di un programma

Ref: The Java Tutorial,

Getting Started > The "Hello World!" Application > "Hello World!" for Solaris OS and Linux

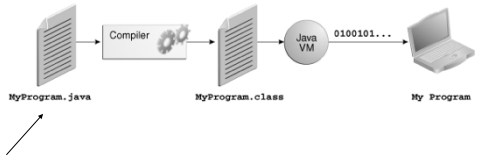
Compilazione ed esecuzione da riga di comando

Un APPLICAZIONE JAVA è un programma stand-alone scritto in linguaggio Java



immafine da: The Java Tutorial, www.oracle.com

Scrittura di un'applicazione Java

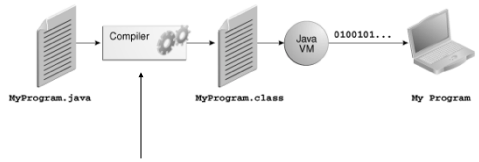


Creazione di un file sorgente.

Un file sorgente è un file di testo, scritto in linguaggio Java.

immafine da: The Java Tutorial, www.oracle.com

Compilazione

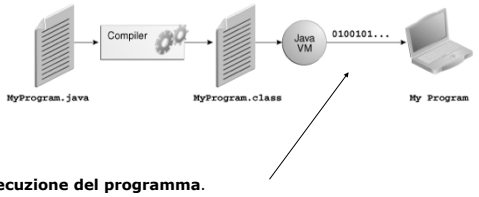


Compilazione del file sorgente in un file bytecode.

Il compilatore java, javac, traduce il testo contenuto nel file sorgente in istruzioni comprensibili alla Java Virtual Machine (Java VM) e le scrive in un file bytecode.

immafine da: The Java Tutorial, www.oracle.com

Esecuzione

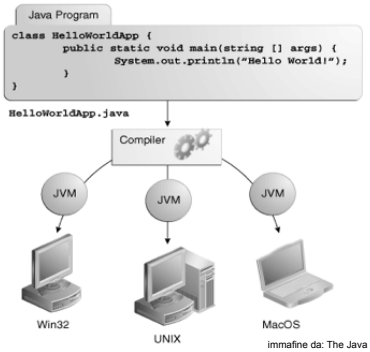


Esecuzione del programma.

La Java VM è implementata da un interprete Java, java. Traduce le istruzioni bytecode (indipendenti dalla piattaforma) in istruzioni comprensibili dal computer e le esegue.

immafine da: The Java Tutorial, www.oracle.com

Portabilità del bytecode Java



immafine da: The Java Tutorial, www.oracle.com

9 Programmazione 2 - Marco Ronchetti

Hello World (application)

Fac.Scienze – Università di Trento

Lo schema CONSIGLIATO di ogni applicazione é:

```
class Applicazione{
/* Hello World, my first Java application  - second version*/
public static void main (String args[]) {
        Applicazione p= new Applicazione();
   }
   Applicazione() {
        System.out.println("Hello World!");
        // qui va il resto del programma principale
   }
}
```

Classe Countdown

```
package it.unitn.prog2;

public class Countdown {
  public static void main(String[] args) {
      Countdown c = new Countdown ();
   }

   Countdown(){
      for (int i = 100; i >= 0; i--) {
         // stampo i
         System.out.println("Counting down: " + i);
      }
   }
}
```

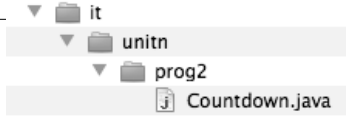
Classe Countdown

```
package it.unitn.prog2;

public class Countdown {
  public static void main(String[] args) {
      Countdown c = new Countdown ();
   }

   Countdown(){
      for (int i = 100; i >= 0; i--) {
         // stampo i
         System.out.println("Counting down: " + i);
      }
   }
}
```

Countdown.java

Gestione dei file sorgente (.java) e bytecode (.class)

```
package it.unitn.prog2;

public class Countdown{
 //...
}
```



Compilatore Java

`javac`

Linux: /usr/local/jdk1.7.0_45/bin/javac

Compilazione della classe Applicazione:

```
> javac Applicazione.java
```

Sintassi:

> javac [opzioni] file_sorgente

Alcune opzioni di **javac**

-classpath <classpath>
Specifica dove sono i file bytecode (.class)
-sourcepath <sourcepath>
Specifica dove sono i file sorgente (.java)
-d <directory>
Specifica in quale directory salvare i bytecode prodotti
-verbose
Stampa informazioni aggiuntive durante la compilazione

Esempio

> javac Applicazione.java

> javac –verbose
–d classes
src/it/unitn/prog2/Countdown.java

Esempio

Supponiamo di voler compilare un programma composto da due file contenuti nel folder src

> javac –verbose
–sourcepath src
–d classes
src/Messaggio.java

Path dei file sorgenti necessari

Directory di destinazione dei file bytecode

Nome del file da compilare

> javac –d classes –classpath classes
src/Saluto.java

> javac –d classes –sourcepath src
src/Saluto.java

Saluto dipende da Messaggio

Interprete Java

Linux:
/usr/local/jdk1.7.0_45/bin/java

`java`

Sintassi:
java [opzioni] *unaClasse* [argomenti]
(opzioni: -classpath –verbose –version –help)

Esempio:
> java Applicazione
> java -classpath classes it.unitn.prog2.Countdown

NB: *unaClasse* deve aver definito il metodo main:
public static void main(String[] argomenti)

Il metodo main

Il punto di entrata dell'esecuzione di un programma Java è il metodo *main:* quando un interprete java esegue un applicazione, chiama il metodo main della classe sulla quale è stato invocato eseguendone il codice

```
public static void main(String[] args)
```

Compilazione ed esecuzione di un programma utilizzando NetBeans

Ref: The Java Tutorial,

Getting Started > The "Hello World!" Application > "Hello World!" for the NetBeans IDE

Riferimenti e Approfondimenti

Compilazione, esecuzione usando i tool del JDK:

- Thinking in Java, cap. 2, par. 'Compiling and running'

Compilazione, esecuzione usando NetBeans IDE o tools JDK:

- Lesson: 'The "Hello World!" Application' http://download.oracle.com/javase/tutorial/getStarted/cupojava/index.html

Esercizi

Esercizio 1

Scrivere un'applicazione che calcoli i primi 20 numeri primi maggiori di due e ne stampi a video uno ogni quattro.

22

Traccia

Seguire lo schema consigliato per un'applicazione.

Usare un ciclo while o un ciclo for e una variabile booleana o una variabile intera per controllare il numero di iterazioni

Usare un costrutto if per verificare se un numero e' divisible per un'altro o se e' da stampare

Esercizio 1

```
public NumeriPrimi() {

    int numeriTrovati = 0;
    int numeroDaTestare = 3;

    while (numeriTrovati < 20) {
        //verifico se numeroDaTestare e' primo
        boolean primo = false;
        boolean divisibile = false;
        for(int i=2;i<numeroDaTestare && !divisibile;i++){
            divisibile = ((numeroDaTestare % i)==0);
        }
        primo = !divisibile;
        // se e' primo, incremento il numero dei trovati
        if(primo) numeriTrovati+=1;
        // se e' primo, controllo se e' da stampare
        if(primo && numeriTrovati%4==1)
            System.out.println(numeroDaTestare);
        //incremento di uno il numero da testare
        numeroDaTestare++;
    }

}
```

Leggere

Variabili / operatori :

The Java Tutorial, Learning the Java Language > Language Basics

> Variables -->

> Primitive Data Types, boolean int

> Assignment, Arithmetic, and Unary Operators

> Equality, Relational, and Conditional Operators

Leggere

Costrutti iterativi e condizionali:

The Java Tutorial, Learning the Java Language > Language Basics

> The if-then and if-then-else Statements

> The while and do-while Statements

> The for Statement

Riferimenti e Approfondimenti

Costrutti condizionali ed iterativi:

- http://download.oracle.com/javase/tutorial/java/nutsandbolts/flow.html
- Thinking in Java, cap. 3: 'Controlling Program Flow', par. 'Execution control'

Esercizio 2

Scrivere un'applicazione che calcoli la differenza tra un numero intero n ed il primo intero > n che sia contemporaneamente:

- divisibile per 9,
- multiplo di 5
- e che, diviso per 7, non dia resto 3.

Esercizio 3

Scrivere un'applicazione che definisca un array di caratteri e calcoli quale carattere e' presente il maggior numero di volte.

Utilizzare una variabile di tipo array di char.

Traccia

Array: The Java Tutorial, Learning the Java Language > Language Basics> Arrays

java-tutorial/java/nutsandbolts/arrays.html

char[] caratteri = { 'a', 'b', 'a', 'c', 'b', 'b'};

→

Esercizio 4

Scrivere un'applicazione che calcoli il prodotto degli elementi presenti sulla diagonale di una matrice quadrata.

Implementare la matrice tramite un array bidimensionale di valori interi.

Indice

Introduzione all'ambiente di sviluppo
- Compilazione ed esecuzione di un programma
- Compilazione ed esecuzione di un programma usando NetBeans IDE

Esercizi

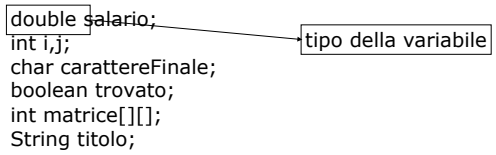
Java, sintassi

Dichiarazione di variabili

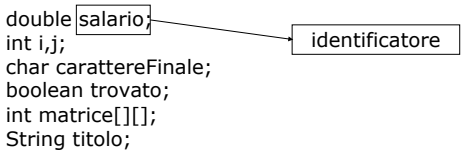
```
double salario;
int i,j;
char carattereFinale;
boolean trovato;
int matrice[][];
String titolo;
```

←

Dichiarazione di una variabile



Dichiarazione di una variabile



I nomi (identificatori) delle variabili sono di lunghezza arbitraria, devono iniziare con una lettera (o con uno dei caratteri _ $) e possono contenere caratteri alfanumerici.

Non possono essere delle parole-chiave del linguaggio.

Keywords

In Java sono parole riservate (o parole-chiave):

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Dichiarazione di una variabile

```
double salario;
int i,j;
char carattereFinale;
boolean trovato;
int matrice[][];
String titolo;
```

tipi di dato semplici (o primitivi)

Dichiarazione di una variabile

```
double salario;
int i,j;
char carattereFinale;
boolean trovato;
int matrice[][];
String titolo;
```

tipi di dato composti

Assegnamento di un valore ad una variabile

```
salario = 1500.00;
i = 4;
carattereFinale = 'L';
trovato = false;
```

←

Tipi di dati primitivi

	dimensione	valori	esempio
boolean	-	true / false	false
char	16 bit	A single character	'c'
byte	8 bit	[-128, +127]	3
short	16 bit	[-32768, 32767]	8
int	32 bit	[-2^31, +2^31-1]	187
long	64 bit	[-2^63, +2^63-1]	8834L
float	32 bit		87.363F
double	64 bit		26.77e3

←

Tipi di dati primitivi

	dimensione	valori	esempio
boolean	-	true / false	false

```
boolean a, b, c;
a = false;
b = !a;      //negazione

c = a | b;      //valuta sia a che b
c = a && !b;   //non sempre valuta b
```

←

Tipi di dati primitivi

	dimensione	valori	esempio
char	16 bit		'c'

```
char aChar;
aChar = 'S';

char anotherChar = 'B';
```

Tipi di dati primitivi

	dimensione	valori	esempio
byte	8 bit	[-128, +127]	3
short	16 bit	[-2^15, +2^15-1]	8
int	32 bit	[-2^31, +2^31-1]	187
long	64 bit	[-2^63, +2^63-1]	8834L

```
int i, j, k;
long n = 178L;
i = 0; j=++i; k = -j;

k= 512 % 22; //resto della divisione
k +=4; // equivale a k=k+4;
```

←

Tipi di dati primitivi

	dimensione	valori	esempio
float	32 bit		87.363F
double	64 bit		26.77e3

```
double a, b;
float c;

a = 37.266;
b = 37.266D;
c = 87.363F;
```

Blocchi e visibilità delle variabili

Un blocco di istruzioni è una lista di istruzioni racchiusa tra parentesi graffe che può essere usata ovunque al posto di una singola istruzione. I blocchi definiscono lo scope delle variabili.

Lo *scope* (o ambito di visibilità) di una variabile è la sezione di codice in cui una variabile può essere utilizzata. Esempi:

```
public static void main(String[] args) {
    int n;
    {
      int k;
    }
    k = 3; //errore: k non è più visibile
}
```

Blocchi e visibilità delle variabili

Un blocco di istruzioni è una lista di istruzioni racchiusa tra parentesi graffe che può essere usata ovunque al posto di una singola istruzione. I blocchi definiscono lo scope delle variabili.

Lo *scope* (o ambito di visibilità) di una variabile è la sezione di codice in cui una variabile può essere utilizzata. Esempi:

```
public static void main(String[] args) {
    int n;
    {
      int k;
      int n;        // errore!! variabile già definita
    }
}
```

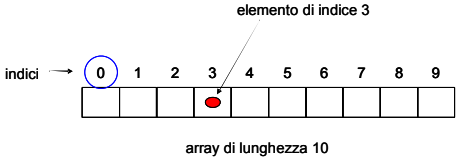
Commenti

3 tipi di commenti:

```
/*        questo è un commento
          su più righe      */

// questo è un commento su una riga sola

/**
  *  anche questo è un commento su più righe
  */
```

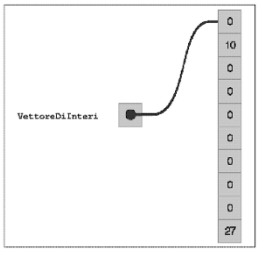
Array

Un array è una struttura dati che raccoglie una collezione di valori dello stesso tipo. A ciascun valore si accede tramite un indice intero. Esempio: se *a* è un array di int, *a[i]* è l'i-esimo intero dell'array.



Esempi

```
int[] vettoreDiInteri;
vettoreDiInteri = new int[10];
//dereferenziazione
vettoreDiInteri[1] = 10;
vettoreDiInteri[9] = 27;

int[] anni = { 2010,
2011, 2012, 2013 };
```



Riferimenti e Approfondimenti

Variabili, tipi di dati primitivi, operatori:
Lezione: 'Language Basics' sul Java Tutorial:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

51

Strutture di controllo

Costrutti condizionali:
- if-then
- if-then-else
- switch

Costrutti iterativi:
- ciclo for
- ciclo while
- ciclo do .. while

Costrutti condizionali

I costrutti condizionali permettono di specificare una sezione di codice che va eseguita solo in particolari condizioni.

Piu' precisamente, permettono di definire dei flussi di istruzioni alternativi all'interno di un programma.

Costrutti iterativi

I costrutti iterativi permettono di specificare una sezione di codice che va eseguita zero, una o piu' volte consecutivamente.