Programmazione II

Marco Ronchetti

Obiettivi

Il corso introduce le tecniche e costrutti della programmazione ad oggetti come una evoluzione necessaria per affrontare il problema della crescente complessità degli artefatti software.

Verrà utilizzato il linguaggio Java.
(dopo aver fatto alcuni richiami di C++)

Il corso è prevalentemente teorico, ma avrà anche una parte pratica.
vi saranno alcune esercitazioni di introduzione a tool per l'uso di Java.

Impegno

1 credito = 25 ore di studio

6 crediti = 150 ore. In aula: 12x4=48 ore

⇒PER OGNI ORA DI LEZIONE IN AULA
OCCORRE STUDIARE (Studio, ripasso, esercizi)
DUE ORE FUORI AULA

Supporto

Materiale on-line

- *copia delle slides*

- *registrazione audio-video delle lezioni (on line – su CD - DVD)*

- *discussion board*

Accessibile da http://latemar.science.unitn.it

or google for "Marco Ronchetti"

Esame

Sul sito web trovate il materiale degli anni scorsi, compresi alcuni testi di esame.

Esame articolato in due fasi:

- **Primo scritto (40 min,**
  - **7 esercizi di lettura di codice,**
  - **10 domande a risposta multipla,**
  - **correzione immediata)**
- **Prova pratica (sviluppo di codice, 3 ore).**

Domande e risposte

ASSEGNAZIONE	DEADLINE	TASK
MARTEDI	MERCOLEDI 23:59	FAI UNA DOMANDA
GIOVEDI	LUNEDI 23:59	RISPONDI A UNA DOMANDA
MARTEDI	MERCOLEDI 23:59	VALUTA UNA RISPOSTA

http://persistence.disi.unitn.it:9894/t4e

1 punto per il completamento del 75% delle task
1 punto per il piazzamento nel top 33%

Programmazione industriale

Programming "in the large"

- *Suddivisione del lavoro tra persone/gruppi (divide et impera)*
- *Mantenibilità (che succede se voglio cambiare qualcosa tra un mese/un anno/...)*
- *Robustezza (che succede se sostituisco una persona?)*

Programmazione industriale

Le risposte:

Ingegneria del software
(corso del prossimo anno)

Buone tecniche di programmazione
(es. commenti up to date)

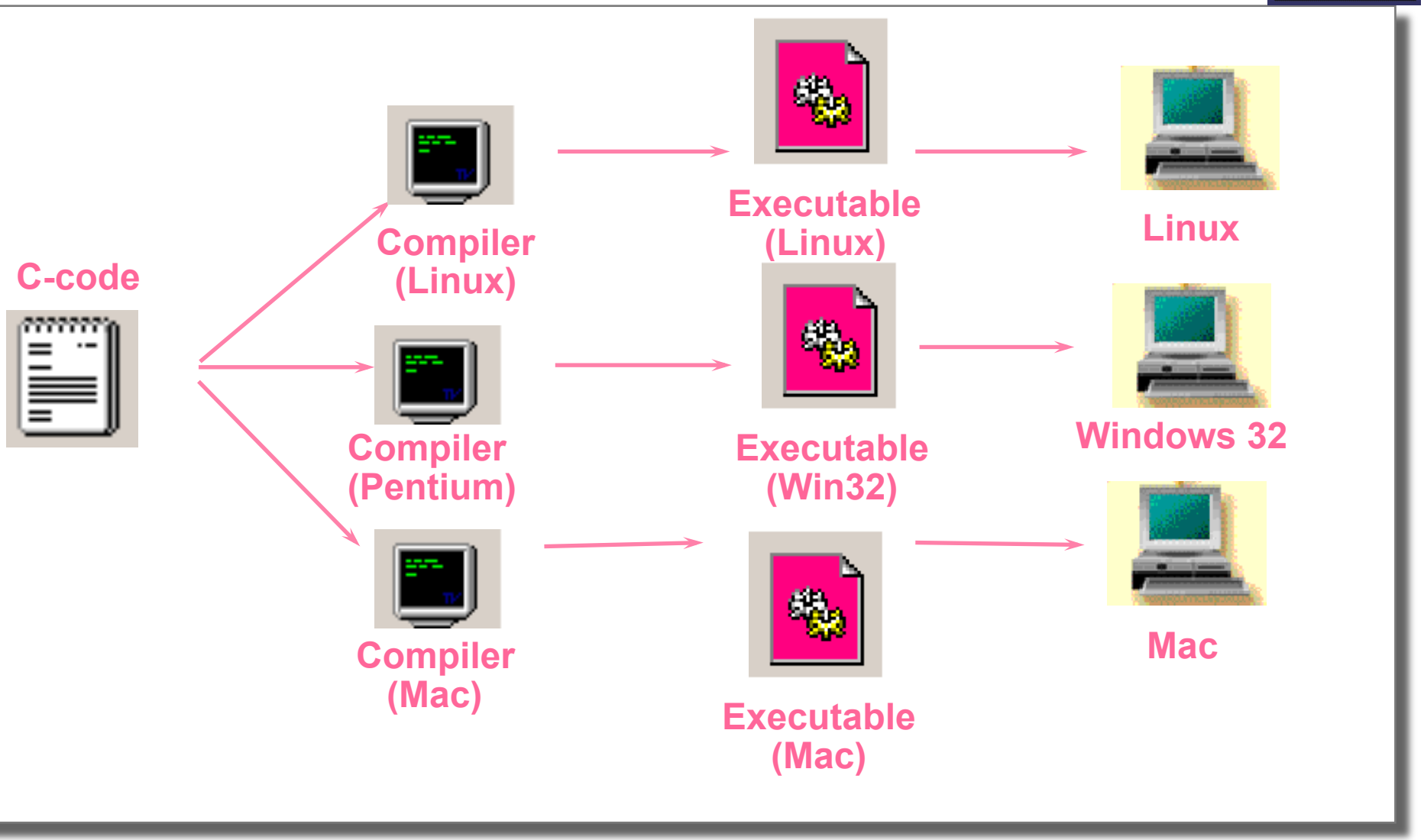
Supporto dal linguaggio:
Object Oriented Programming
(in C++)
(in Java)

Java

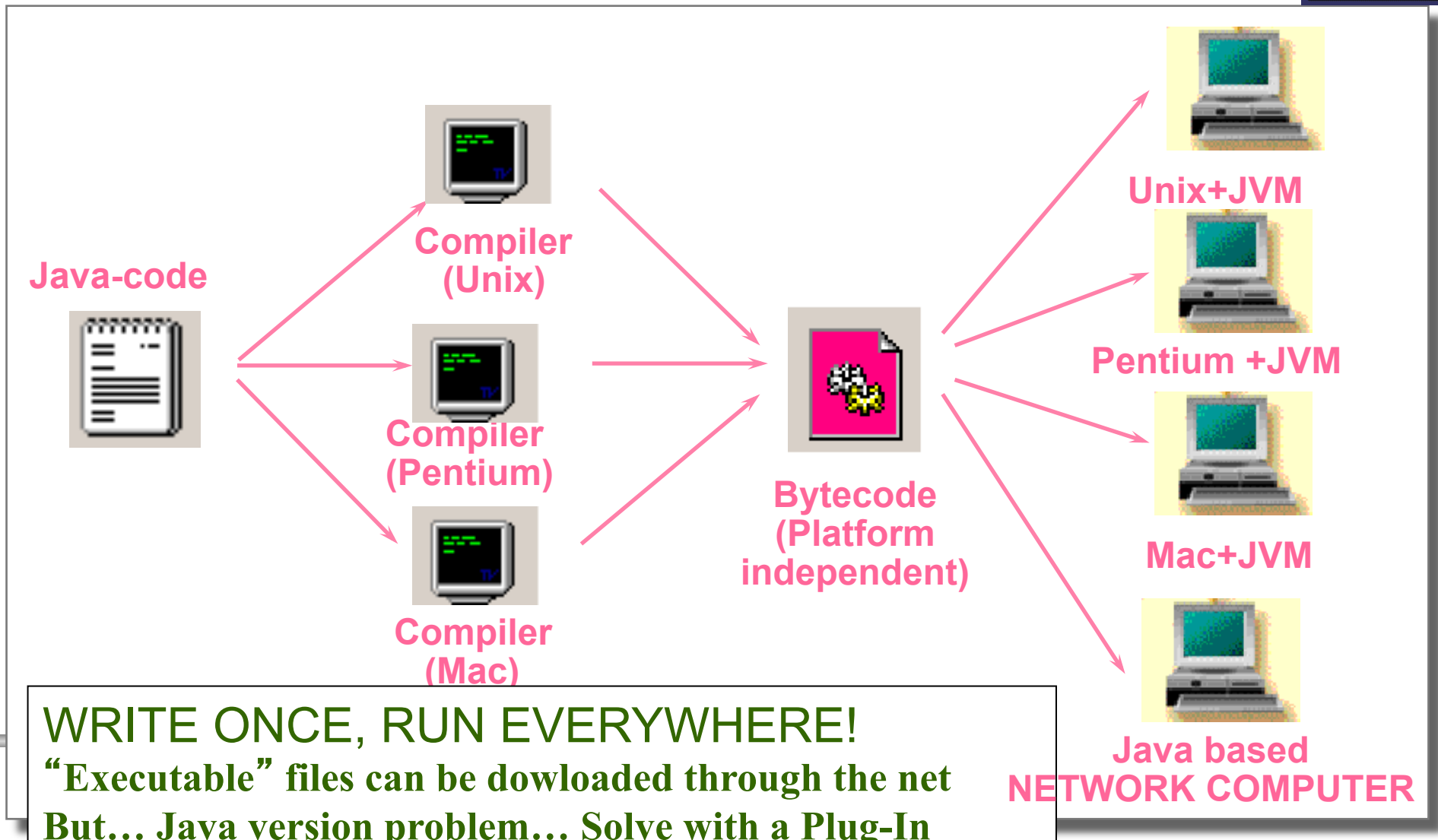
JAVA:

una introduzione

Traditional "portability" (ideal)



Portability of Java programs



WRITE ONCE, RUN EVERYWHERE!

"Executable" files can be dowloaded through the net

But… Java version problem… Solve with a Plug-In

Esecutori di bytecode

Java può essere eseguito:

- come **standalone program**
  - ✓ da interpreti java (o compilatori JIT, o Java Chips)
- come "**applet**":
  - ✓ da browsers Web:
  - ✓ da applicativi ad hoc:
- come "**add-on module**":
  - ✓ da server Web
  - ✓ da application server (Enterprise Java Beans)

Java - Introduction

Applications are built

in the frame of the

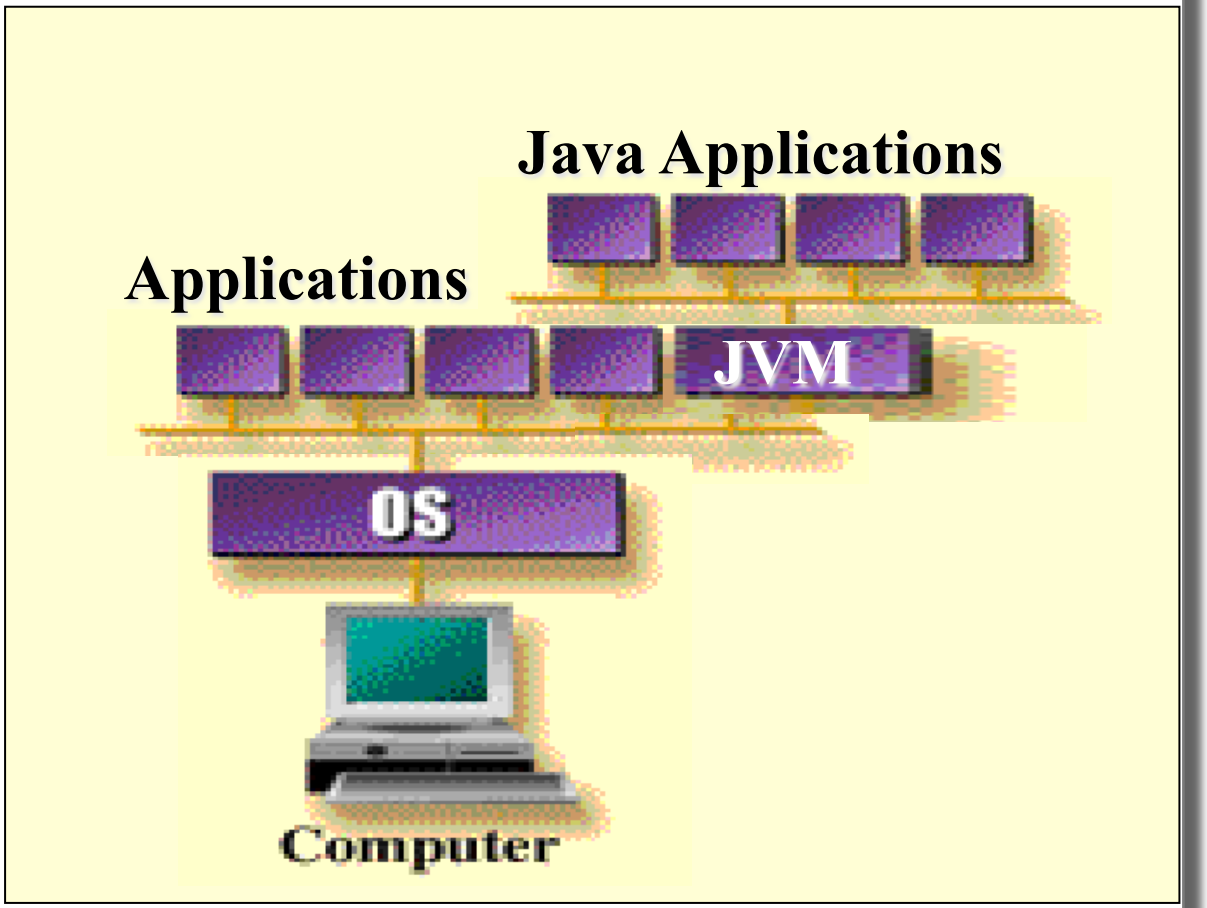
OPERATING SYSTEM

Which in turn is built over a particular

HARDWARE

Java - Introduction

Java defines a HW-OS neutral **SOFTWARE LAYER** on top of which its code runs



The Java Virtual Machine

The Software Layer is called

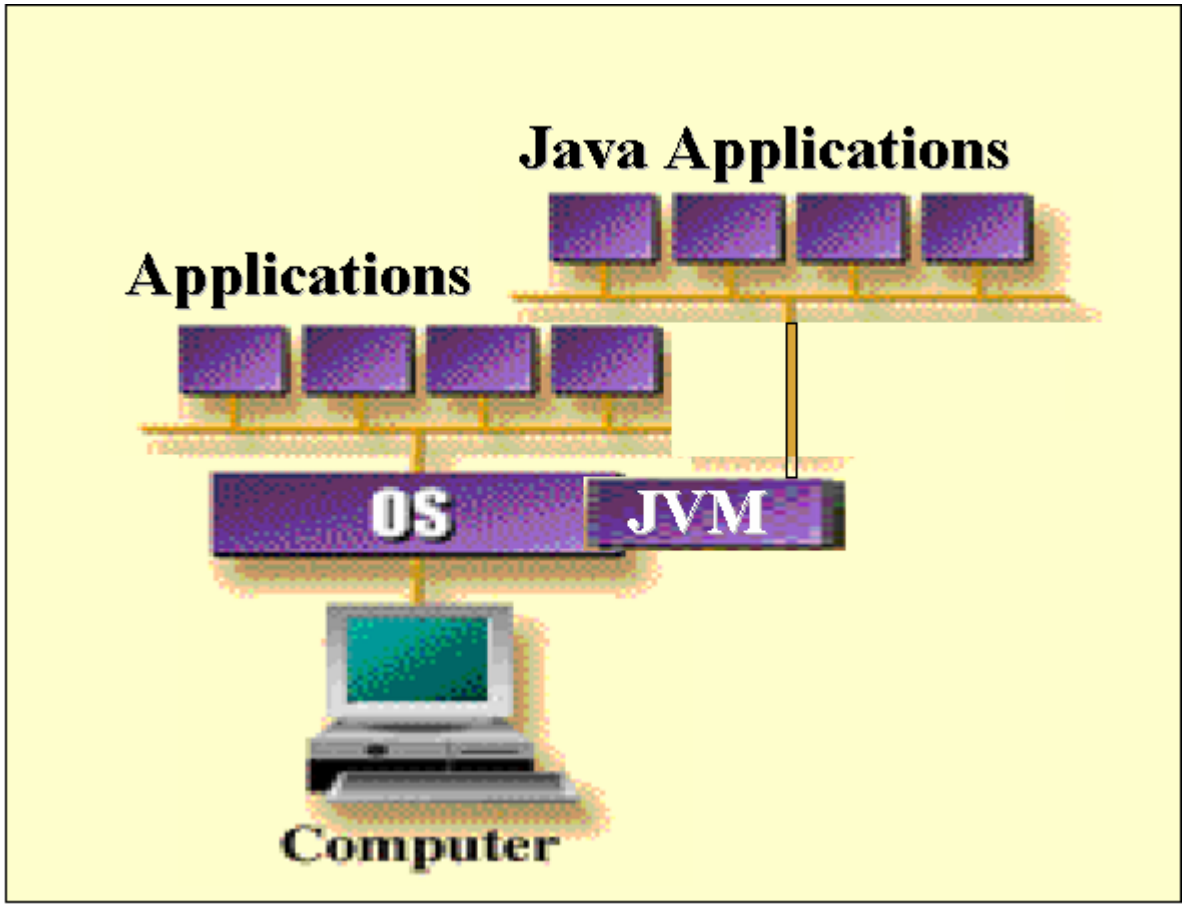
Java Virtual Machine

It is a (smart) *interpreter* of an assembly-like language called

ByteCode

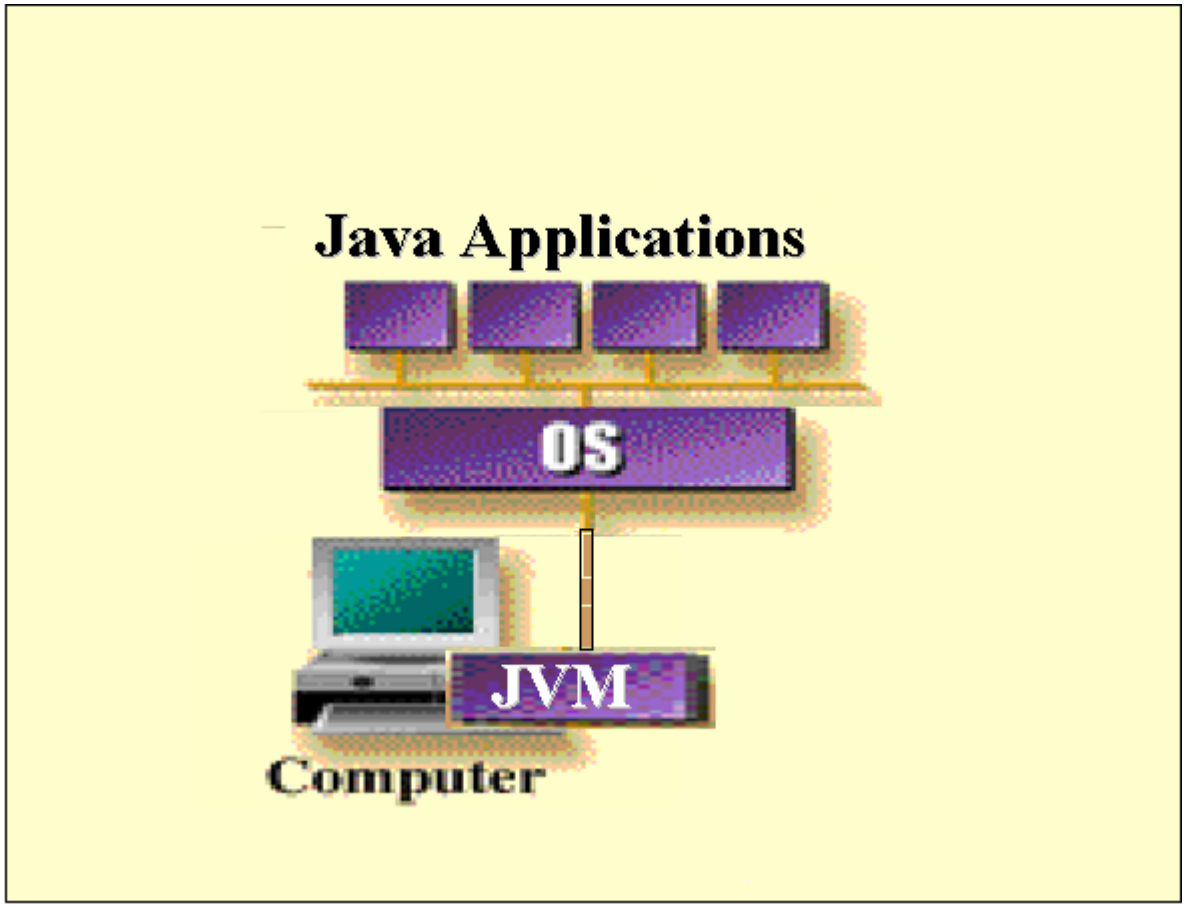
Java - Introduction

In principle the JVM could be a SW component ff the OS



Java - Introduction

In principle the JVM could be embedded in the Hardware!



"the first universal software platform"

Consists of:

The language **Easy!**

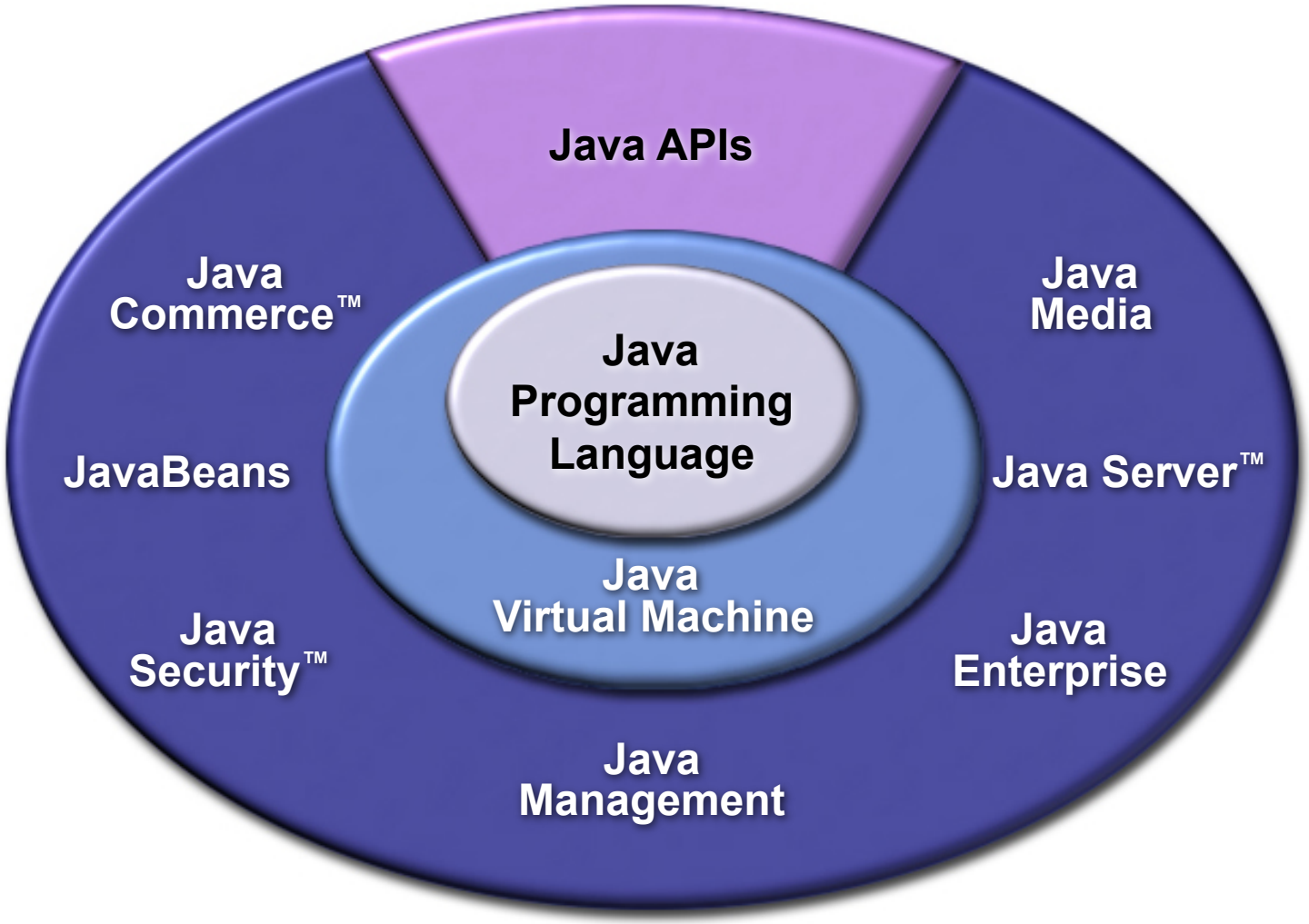
The Virtual Machine **You don't care!**

(Many) class libraries and API **That's the difficult part!**

Java: the platform for **"Internet Computing"**

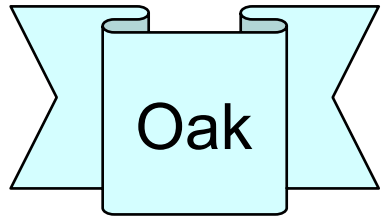
Hardware independent • Scalable • Open

The Java Platform

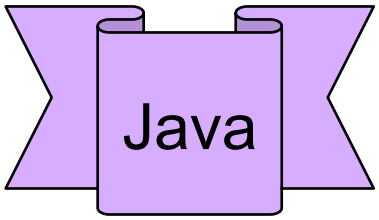


Storia di Java

- Inizio anni 90: Java nasce come "**Oak**"
  target: **intelligent consumer electronics.**
- Successivamente, nuovo target: **set top box**



- 1994: linguaggio per la "**Web**" (client side)
- 1996: la prospettiva é "**network computing**"



Oggi:

Successi

- **Device-independent GUI**
- **Web on the server side (Servlets, JSP, EJB, XML…)**
- **Android!**

In salsa microsoft…

Visual-J

C#

Applicazioni

Definizione:

Programmi stand-alone scritti in linguaggio Java.

Possono essere eseguiti da una Java Virtual Machine:

- Fisica: un processore il cui assembler e' il bytecode
- Virtuale: un interprete o Just In Time Compiler Java.

Prestazioni…

Inferiori al C++…

Tempo di sviluppo:

Inferiore al C++ ...

Hello World (application)

Lo schema MINIMO di ogni applicazione é:

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

Hello World (application)

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
   }
}
```

Uso di JDK

Compilazione:

$javac HelloWorld.java

Obbligatorio specificare l'estensione!

produce HelloWorld.class

(in realtà: un file class per ogni classe contenuta nel sorgente)

Esecuzione…

$java HelloWorld

Obbligatorio omettere l'estensione!

(la classe indicata deve contenere il main)

Basic tools

http://www.oracle.com/technetwork/java/javase/downloads/index.html/

Java™ Platform, Standard Edition (J2SE™)

The essential Java SDK, tools, runtimes, and APIs for developers writing, deploying, and running applets and applications in the Java programming language.

Advanced development tool

http://www.oracle.com/technetwork/java/javase/downloads/index.html/

Eclipse Project

jdt
java development tools subproject

http://www.eclipse.org/

free

Un buon libro…

Gratis in forma elettronica:
Thinking in Java
Bruce Eckel

http://www.mindview.net/Books

In Italiano:
Thinking in Java
Bruce Eckel
Ed. Apogeo
(in libreria)

"The" Tutorials and examples

http://docs.oracle.com/javase/tutorial/?frontpage-spotlight

More Tutorials and examples

MokaByte

La prima rivista web italiana dedicata a Java®

http://www.mokabyte.it/

Facilità

Java è basato sul C, come il C++.

- Java **TOGLIE** al C alcune caratteristiche difficili e pericolose (puntatori).

- Java **AGGIUNGE** al C le caratteristiche di un linguaggio object-oriented (classi, ereditarietà, messaggi).

- Java **INTRODUCE** una gerarchia di classi predefinite:
  AWT, IO, Lang(tipi, Math, Thread), Exeptions, Net,
  Utils(Vector, Dictionary, Date...)

Robustezza

La maggior parte degli errori sono legati alla gestione della memoria tramite i **PUNTATORI**:

- puntatori che puntano a locazioni illecite (non allocate)
- puntatori che puntano a locazioni lecite ma sbagliate
- indirizzi di vettori sbagliati
- memoria allocata e non più rilasciata (memory leaks)

Soluzione di Java:

- **ABOLIZIONE DEI PUNTATORI**
- **GARBAGE COLLECTION**

Differenze tra Java e C++

?(Java == ((C++)- -)++)

Forma di un programma

In Java tutto e' una "classe".

Lo scheletro minimo di un programma e':

```
import ...;
class myProgram {
  public static void main (String args[]) {
      System.out.println("Java is running!");
  }
}
```

import <= Include "intelligente"

(senza bisogno di #ifdef)

NON c'è precompilatore!

Tipi di dato primitivi

Type	Contains	Default	Size	Min/Max Value
boolean	true or false	false	1 bit	N.A. / N.A.
char	Unicode char	\u0000	16 bits	\u0000 / \uFFFF
Byte	signed integer	0	8 bits	-128 / 127
short	signed integer	0	16 bits	-32768 / 32767
int	signed integer	0	32 bits	-2147483648 / 2147483647
long	signed integer	0	64 bits	-9223372036854775808 / 9223372036854775807
float	IEEE 754 f.p.	0.0	32 bits	+/-3.40282347E+38 / +/-1.40239846E-45
double	IEEE 754 f.p.	0.0	64 bits	+/-1.79769313486231570E+308 / +/-4.94065645841246544E-324

Literals (costanti)

interi (sempre int, long se serve)

0777 ottale 0xFF esadecimale 77L long

reali

10.4 1.04E01 double 10.4F 1.04E01F float

boolean

true false

carattere

tutte le escape sequences del C sono riconosciute (\n \t \' \" \\ …)

Unicode: \u0022 has exactly the same meaning to the compiler as "

stringhe

"questa e' una stringa"

Arrays

E' possibile definire arrays di tutti i tipi di dati (elementari o classi). In fase di DEFINIZIONE non e' necessario specificare la dimensione del vettore.

Solo al momento della ALLOCAZIONE viene richiesto lo spazio desiderato.

```
String[ ] strings; // this variable can refer to any String array
strings = new String[10];   // one that contains 10 Strings
strings = new String[20];   // or one that contains 20.

float f[ ][ ] = new float[5][3]; //array bidimensionale

char s[]={'+','-','*','/','=','C'}; // array inizializzato in creazione
```

In C++…


```
int a=1;
if (a=0) a++;
cout << a;
```

Class String

java.lang

Class String

java.lang.Object
  |
  +--**java.lang.String**

All Implemented Interfaces:

CharSequence, Comparable, Serializable

---

public final class **String**
extends Object
implements Serializable, Comparable, CharSequence

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Class String

Constructor Summary

String()
Initializes a newly created String object so that it represents an empty character sequence.

String(byte[] bytes)
Constructs a new String by decoding the specified array of bytes using the platform's default charset.

String(byte[] ascii, int hibyte)
Deprecated. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String(byte[] bytes, int offset, int length)
Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.

String(byte[] ascii, int hibyte, int offset, int count)
Deprecated. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String(byte[] bytes, int offset, int length, String charsetName)
Constructs a new String by decoding the specified subarray of bytes using the specified charset.

String(byte[] bytes, String charsetName)
Constructs a new String by decoding the specified array of bytes using the specified charset.

String(char[] value)
Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

String(char[] value, int offset, int count)
Allocates a new String that contains characters from a subarray of the character array argument.

String(String original)
Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

String(StringBuffer buffer)
Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.

Class String

Method Summary

char	**charAt**(int index) Returns the character at the specified index.
int	**compareTo**(Object o) Compares this String to another Object.
int	**compareTo**(String anotherString) Compares two strings lexicographically.
int	**compareToIgnoreCase**(String str) Compares two strings lexicographically, ignoring case differences.
String	**concat**(String str) Concatenates the specified string to the end of this string.
boolean	**contentEquals**(StringBuffer sb) Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.
static String	**copyValueOf**(char[] data) Returns a String that represents the character sequence in the array specified.
static String	**copyValueOf**(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations.
byte[]	**getBytes**() Encodes this String into a sequence of bytes using the platform's default charset, storing the result into a new byte array.
void	**getBytes**(int srcBegin, int srcEnd, byte[] dst, int dstBegin) **Deprecated.** *This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the the getBytes() method, which uses the platform's default charset.*

Class String

Constructor Summary

String()
Initializes a newly created String object so that it represents an empty character sequence.

String(byte[] bytes)
Constructs a new String by decoding the specified array of bytes using the platform's default charset.

String(byte[] ascii, int hibyte)
Deprecated. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String(byte[] bytes, int offset, int length)
Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.

String(byte[] ascii, int hibyte, int offset, int count)
Deprecated. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String(byte[] bytes, int offset, int length, String charsetName)
Constructs a new String by decoding the specified subarray of bytes using the specified charset.

String(byte[] bytes, String charsetName)
Constructs a new String by decoding the specified array of bytes using the specified charset.

String(char[] value)
Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

String(char[] value, int offset, int count)
Allocates a new String that contains characters from a subarray of the character array argument.

String(String original)
Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

String(StringBuffer buffer)
Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.

Class String

Method Detail

length

```
public int length()
```

Returns the length of this string. The length is equal to the number of 16-bit Unicode characters in the string.

Specified by:
`length` in interface `CharSequence`

Returns:
the length of the sequence of characters represented by this object.

---

charAt

```
public char charAt(int index)
```

Returns the character at the specified index. An index ranges from `0` to `length() - 1`. The first character of the sequence is at index `0`, the next at index `1`, and so on, as for array indexing.

Specified by:
`charAt` in interface `CharSequence`

Parameters:
`index` - the index of the character.

Returns:
the character at the specified index of this string. The first character is at index `0`.

Throws:
`IndexOutOfBoundsException` - if the `index` argument is negative or not less than the length of this string.

String

Per trasformare il contenuto di una stringa in un intero:

int Integer.parseInt(String s)

Per trasformare il contenuto di una stringa in un float:

float Float.parseFloat(String s)

I parametri del main sono inclusi in un vettore di String

Parametri di ingresso


```
/* sum and average command lines */
class SumAverage {
  public static void main (String args[]) {
    int sum = 0;
    float avg = 0;
    for (int i = 0; i < args.length; i++) {
      sum += Integer.parseInt(args[i]);
    }
    System.out.println("Sum is: " + sum);
    System.out.println("Average is: "
        + (float)sum / args.length);
  }
}
```

Commenti

3 forme di commento:

```
/* C style */
/* Questo tipo di commento
può proseguire su pi linee */
/* NOTA: ATTENZIONE AI /*COMMENTI*/ NIDIFICATI! */
// C++ style
// Una intera riga commentata
a=a+3; // Commento su una linea di codice
/**documentation */
/**Stile di commento usato da JAVADOC
per la generazione automatica di
documentazione */
```

Nomi

I programmi Java includono nomi per identificare alcune entità di programmazione

(packages, classes, interfaces, methods, variables, statement)

Nomi validi sono composti da un numero illimitato di lettere e numeri UNICODE, iniziare con una lettera.

I nomi non possone essere Java keywords.

Unicode

Java characters, strings, and identifiers are composed of 16-bit Unicode characters. This makes Java programs relatively easy to internationalize for non-English-speaking users.

Most platforms cannot display all 38,885 currently defined Unicode characters

The Unicode character set is compatible with ASCII and the first 256 characters (0x0000 to 0x00FF) are identical to the ISO8859-1 (Latin-1) characters 0x00 to 0xFF.

Unicode \u escape sequences are processed before the other escape characters

Operatori

Gruppo	Funzione	Operatori
Arithmetic	comparazione	= =, !=, <, <=, >, >=
	unitari	+, -
	algebrici	+, -, *, /, %
	postfissi	++, --
Bit	shift	<<, >>, >>>
	bitwise comparison	~, &, |, ^
Boolean	relationali	= =, !=
	logici	!, &, |, ^, &&, ||
String	concatenazione	+

Operatori

Poiché Java non vi permette di manipolare i puntatori, non supporta gli operatori di dereferenziazione *, ->, e &.

L'operatore sizeof è pleonastico e quindi soppresso.

package

Una collezione di classi correlate

```
package myclasses;
class A {...};
class B {...};


import myclasses.A;

import myclasses.*;
```

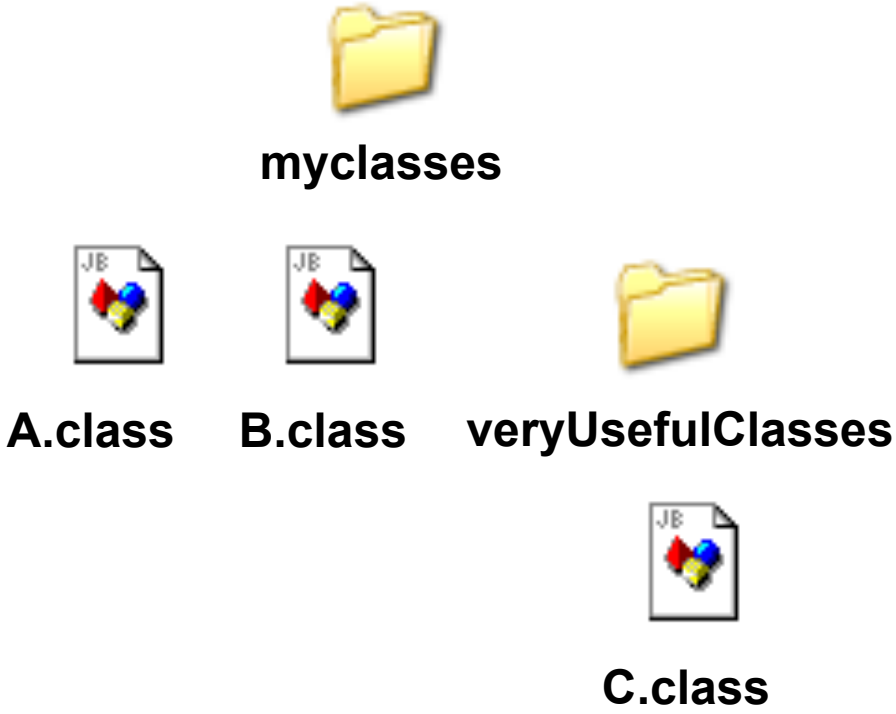
Annidamento di package


```
package myclasses;
class A {...};

package myclasses;
class B {...};

package myclasses.veryUsefulClasses;
class C {...};

import myclasses.*; // NON importa C!
```

Definizione suggerita di un nome univoco per i packages:
È basata sul nome internet (es.: it.unitn.science.mypackage)

Annidamento di package

I packages si riflettono in una struttura di directories



Keywords

Le keywords usate attualmente sono

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

Oltre a queste, alcune keywords sono riservate per usi futuri:

by value cast const future generic goto inner operator
outer rest var