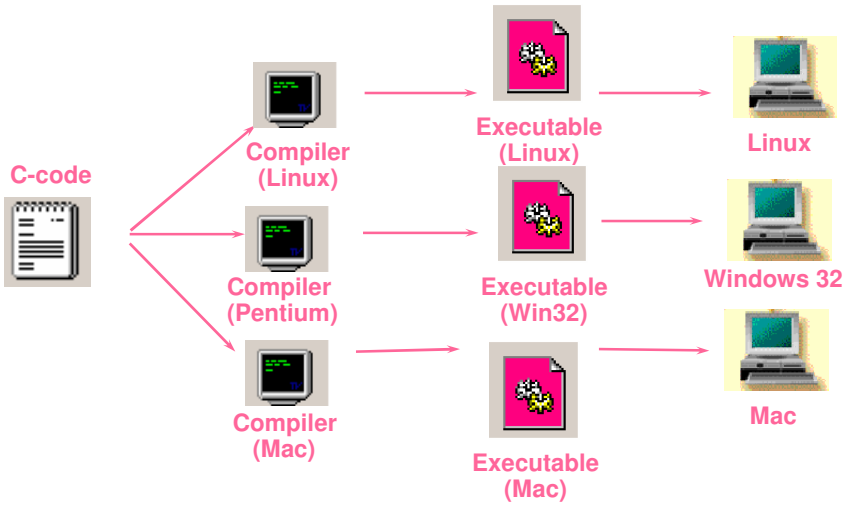
Java

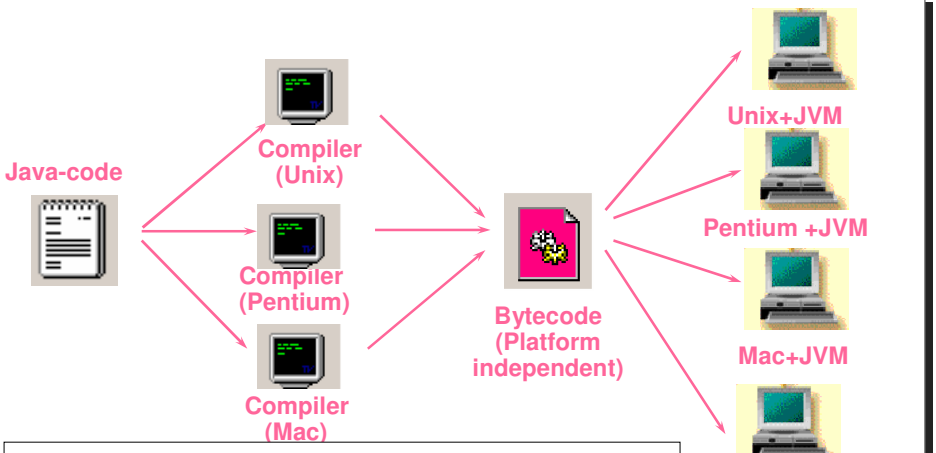
JAVA:

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

Java - Introduction

Applications are built in the frame of the

OPERATING SYSTEM

Which in turn is built over a particular

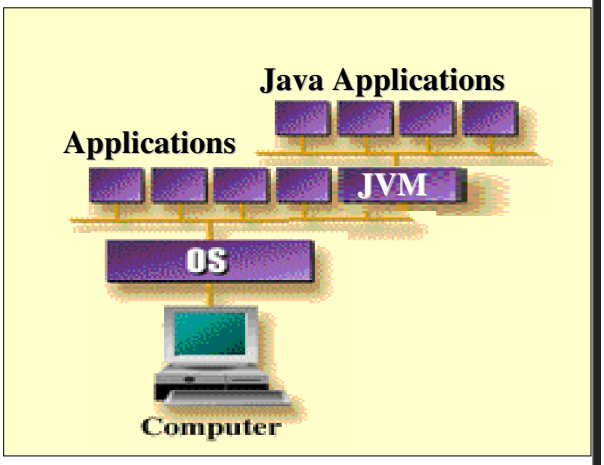
HARDWARE

5 Programmazione 2 - Marco Ronchetti

Java - Introduction

Fac.Scienze – Università di Trento

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



6 Programmazione 2 - Marco Ronchetti

The Java Virtual Machine

Fac.Scienze – Università di Trento

The Software Layer is called **Java Virtual Machine**

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

The Java Virtual Machine

The Java Virtual Machine can:

be an application

live inside an application (e.g. a Browser)

live inside the Operating System (e.g. JavaOS)

Cos'e' un eseguibile Java?

E' un codice "ByteCode": istruzioni (assembler) di una macchina virtuale (Java Virtual Machine).

Non esiste un processore con bytecode ad hardware!
(Sun lo aveva annunciato)

Il "Java processor" viene emulato via software.
Il bytecode viene "interpretato", o eseguito da "Just In Time" Compilers.

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)

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.

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.java.sun.com/j2se

Java™ 2 Platform, Standard Edition (J2SE™)

The essential Java 2 SDK, tools, runtimes, and APIs for developers writing, deploying, and running applets and applications in the Java programming language. Also includes earlier Java Development Kit versions JDK™ 1.1 and JRE 1.1

Advanced development tool

JBuilder®

The leading Java™ development solution

http://www.borland.com/products/downloads/download_jbuilder.html

Personal edition is free

Advanced development tool

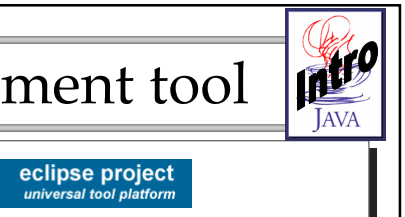
Sun[tm] ONE Studio 4 update 1

Sun[tm] ONE Studio 4 update 1 is the latest release in the Sun ONE Studio line of Integrated Development Environments (IDEs) for Java[tm] technology developers. Based on the NetBeans[tm] Tools Platform, Sun ONE Studio software continues to drive the market forward by providing the latest support for Java and Industry Standards in the development of enterprise class applications and Web services.

http://wwws.sun.com/software/sundev/jde/index.html

Community edition is free

Advanced development tool



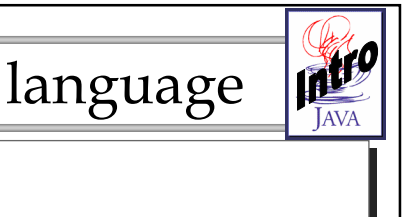
Eclipse Project

jdt
java development tools subproject

http://www.eclipse.org/jdt/index.html

free

Why Java? A safer language



A *clean* object-oriented programming language

No pointer arithmetic

Automatic Memory Management (Garbage Collection)

Automatic array and string bounds check

19 Programmazione 2 - Marco Ronchetti

"the first universal software platform"

Fac.Scienze – Università di Trento

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

20 Programmazione 2 - Marco Ronchetti

Facilità

Fac.Scienze – Università di Trento

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**

Java ha:

- eccezioni (Ada, C++)
- Garbage Collection (LISP, Smalltalk)
- libreria di classi (da Smalltalk, Objective-C)

Le specifiche del linguaggio e della Java Virtual Machine sono PUBBLICHE

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[]) {
      ...
  }
}
```

import <= Include "intelligente"
(senza bisogno di #ifdef)
NON c'è precompilatore!

25 Programmazione 2 - Marco Ronchetti

Nomi

Fac.Scienze – Università di Trento

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.

26 Programmazione 2 - Marco Ronchetti

Keywords

Fac.Scienze – Università di Trento

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

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 */
```


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

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

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"

String Literals

Strings in Java are not a primitive type, but are instances of the String class.

However, because they are so commonly used, string literals may appear between quotes in Java programs, just as they do in C:

"pippo"

When the compiler encounters such a string literal, it automatically creates the necessary String object.

Operatori

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

Operatori

Since Java does not allow you to manipulate pointers directly, it does not support the reference and dereference operators *, ->, and &, nor the sizeof operator.

Java also adds some new operators:

The + operator applied to String values concatenates them. If only one operand of + is a String, the other one is converted to a string.

Java does not support operator overloading--the language designers decided (after much debate) that overloaded operators were a neat idea, but that code that relied on them became hard to read and understand.

Operatori

The instanceof operator returns true if the object o on its left-hand side is an instance of the class C or implements the interface I specified on its right-hand side.

It also returns true if o is an instance of a subclass of C or is an instance of a subclass of some class that implements I. instanceof returns false if o is not an instance of C or does not implement I.

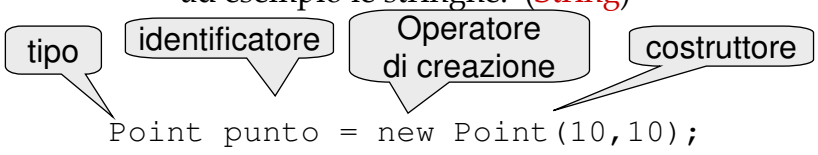
It also returns false if the value on its left is null.

If instanceof returns true, it means that o is assignable to variables of type C or I. The instanceof operator has the same precedence as the <, <=, >, and >= operators.

Tipi di dato derivati (reference data)

Java, come tutti i linguaggi OO, permette di definire NUOVI TIPI DI DATO (classi).

Alcuni tipi di dato (classi) sono predefinite:
ad esempio le stringhe. (String)



No Structures or Unions

Java does not support C struct or union types. Note, however, that a class is essentially the same thing as a struct, but with more features. And you can simulate the important features of a union by subclassing.

"Java non ha i puntatori"

Ma è vero?

```
Point punto = new Point(10,10);
```

l'identificatore di un oggetto ("punto")
sembra proprio un puntatore!

Quel che Java non ha è
l'aritmetica dei puntatori

Confronto dell'operatore new


```
in C++:  Point * punto = new Point(10,10);

in Java:  Point  punto = new Point(10,10);
```

punto.x di Java equivale a punto->x del C++

In Java gli oggetti sono accessibili
SOLO per referenza

memory management

La gestione (dinamica) della memoria e' automatica, tramite
- la creazione (operatore new) e
- la distruzione (garbage collection) di oggetti.

GC interviene quando serve memoria.

GC elimina gli oggetti per i quali non vi sono piu' riferimenti attivi.

GC puo' essere attivato su richiesta esplicita: System.gc()

memory management

Operazioni da eseguirsi alla nascita di un oggetto vanno definite nel metodo "costruttore".

Ogni classe deve avere uno (o piu') costruttori.

Operazioni da associarsi con l'eliminazione di un oggetto possono essere definite nel metodo "distruttore" finalize() (opzionale)

Arrays

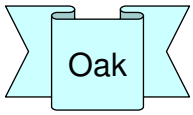
E' possibile definire arrays di tutti i tipi di dati (elementari o classi). In fase di DEFINIZIONE non e' necessario specificare la dimensione del vettore.

Solo al momento della ALLOCAZIONE viene richiesto lo spazio desiderato.

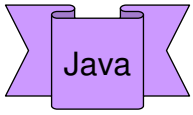
```
String[ ] strings; // this variable can refer to any String array
strings = new String[10];   // one that contains 10 Strings
strings = new String[20];   // or one that contains 20.

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

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

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:

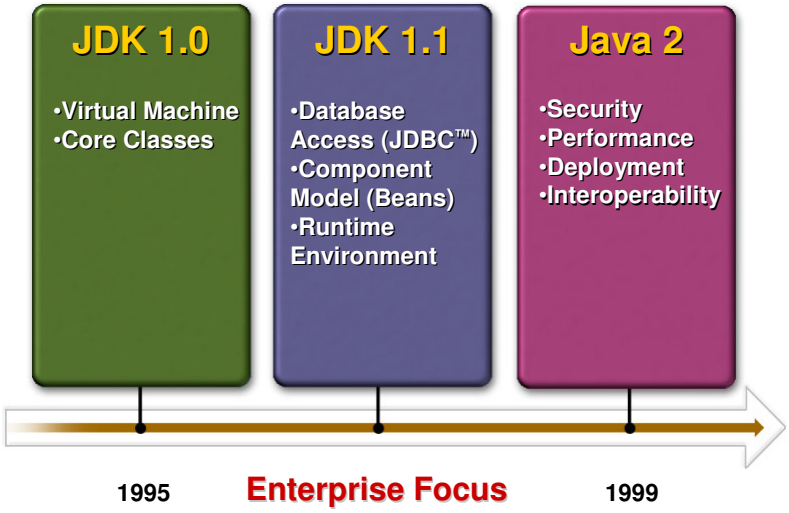
Successi

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

Prospettive

intelligent consumer electronics + smartcards

Evolution of the Java Platform



"The" Tutorials and examples

http://java.sun.com/docs/books/tutorial/?frontpage-spotlight

More Tutorials and examples

http://www.phrantic.com/scoop/onjava.html

http://www.mokabyte.it/

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 Java Platform