JAVA:

una introduzione

Marco Ronchetti - ronchet@altavista.it

Java - Introduction

Applications are built

in the frame of the

OPERATING SYSTEM

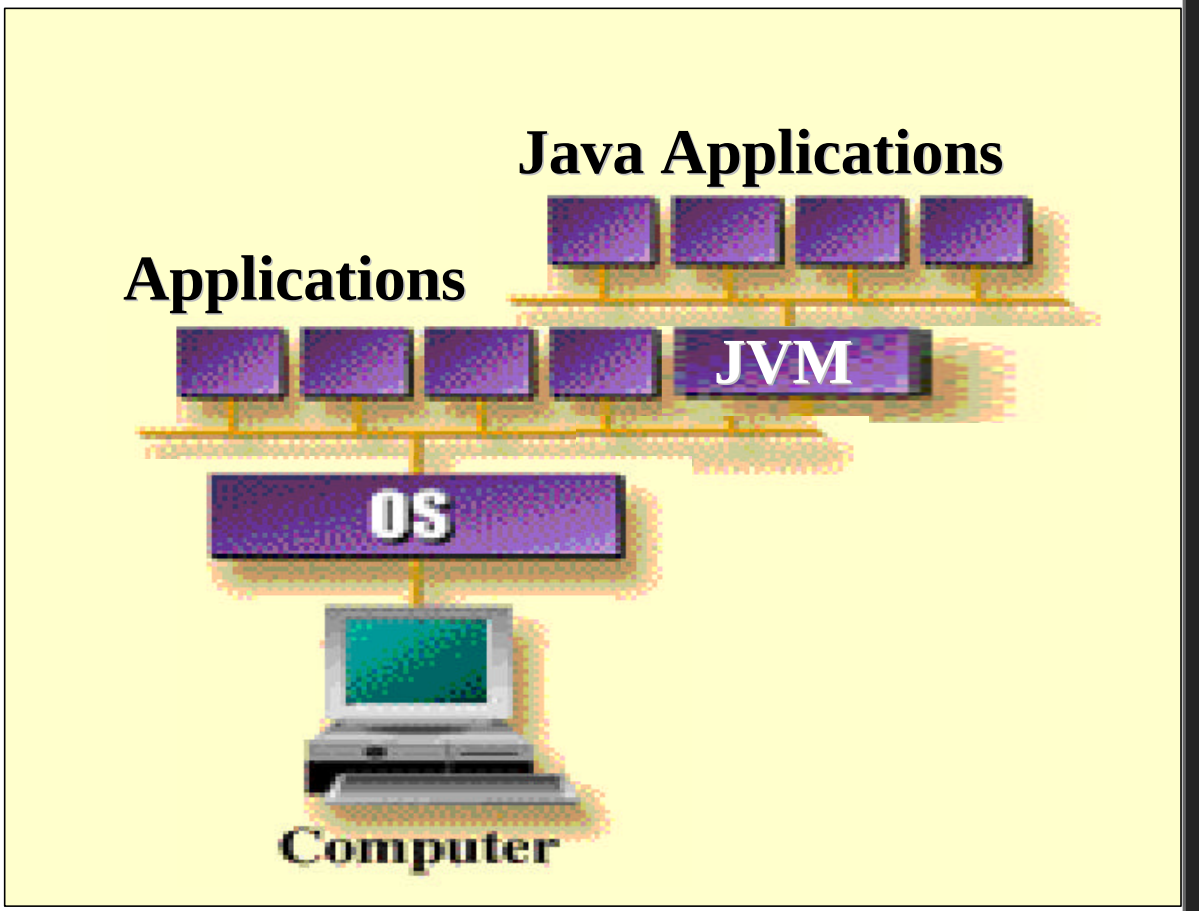
Which in turn is built over a particular

HARDWARE

Marco Ronchetti - ronchet@altavista.it

Java - Introduction

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



Marco Ronchetti - ronchet@altavista.it

The Java Virtual Machine

The Software Layer is called

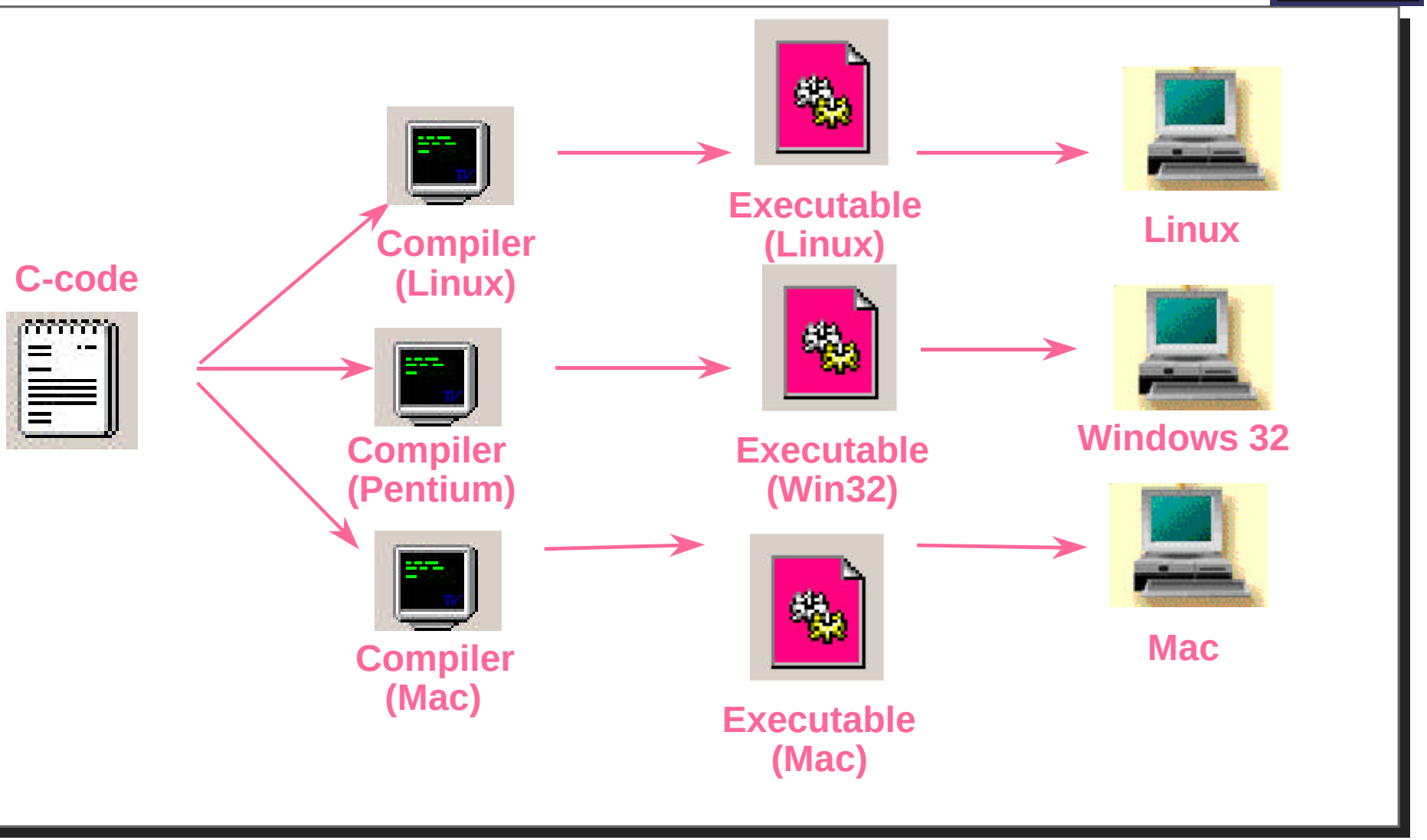
Java Virtual Machine

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

ByteCode

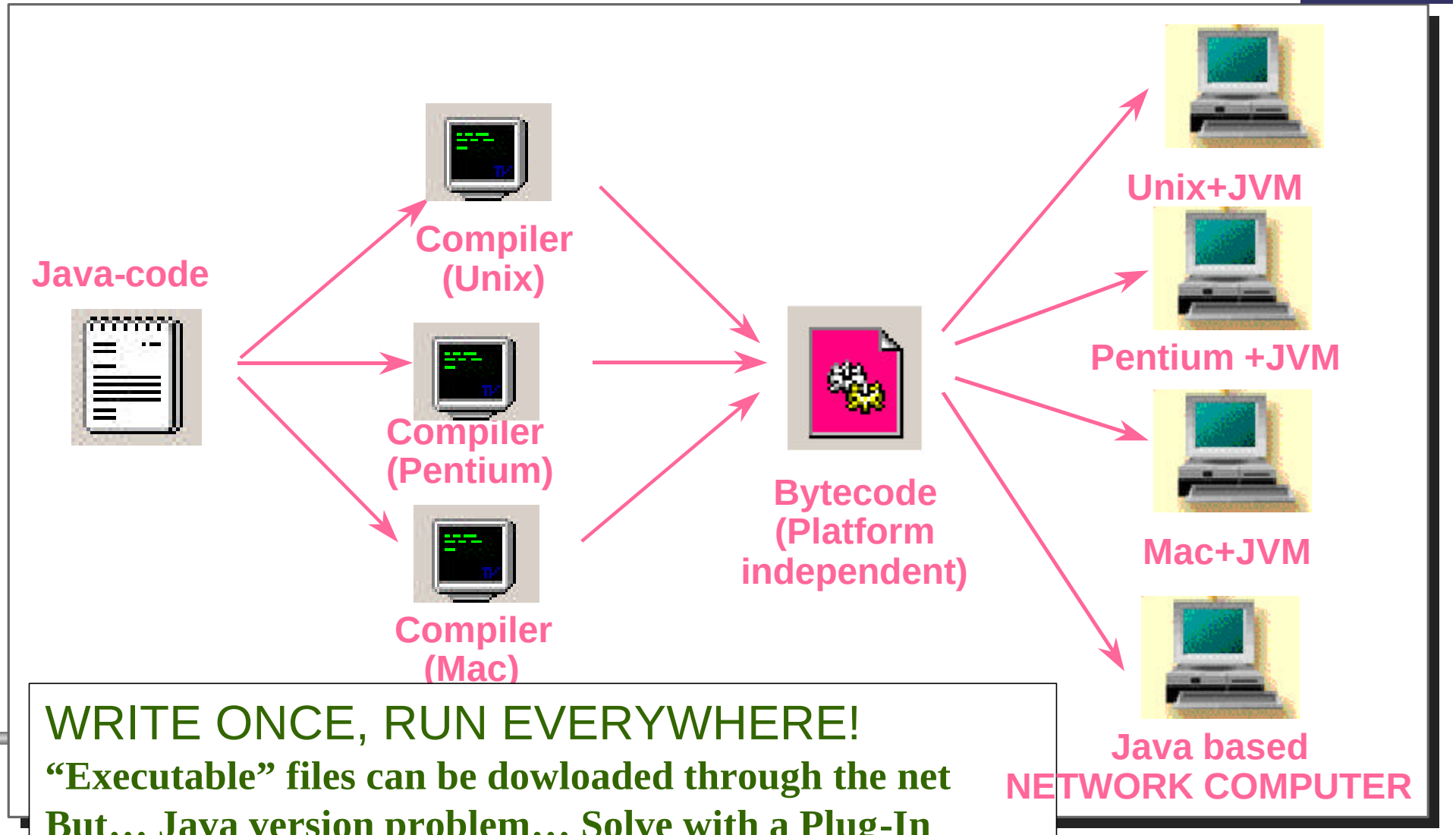
Marco Ronchetti - ronchet@altavista.it

Traditional “portability” (ideal)



Marco Ronchetti - ronchet@altavista.it

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

Marco Ronchetti - ronchet@altavista.it

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)

Marco Ronchetti - ronchet@altavista.it

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

Marco Ronchetti - ronchet@altavista.it

“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

Marco Ronchetti - ronchet@altavista.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**

Marco Ronchetti - ronchet@altavista.it

Java ha:

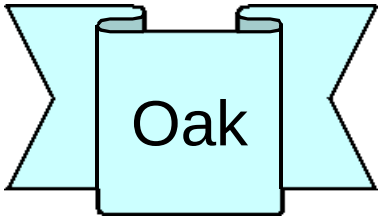
- eccezioni (Ada, C++)
- Garbage Collection (LISP, Smalltalk)
- libreria di classi (da Smalltalk, Objective-C)

Le specifiche del linguaggio e della Java Virtual Machine sono PUBBLICHE

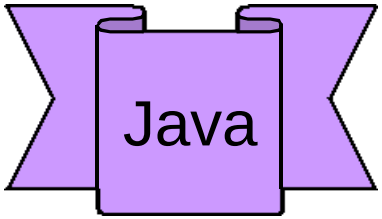
Marco Ronchetti - ronchet@altavista.it

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

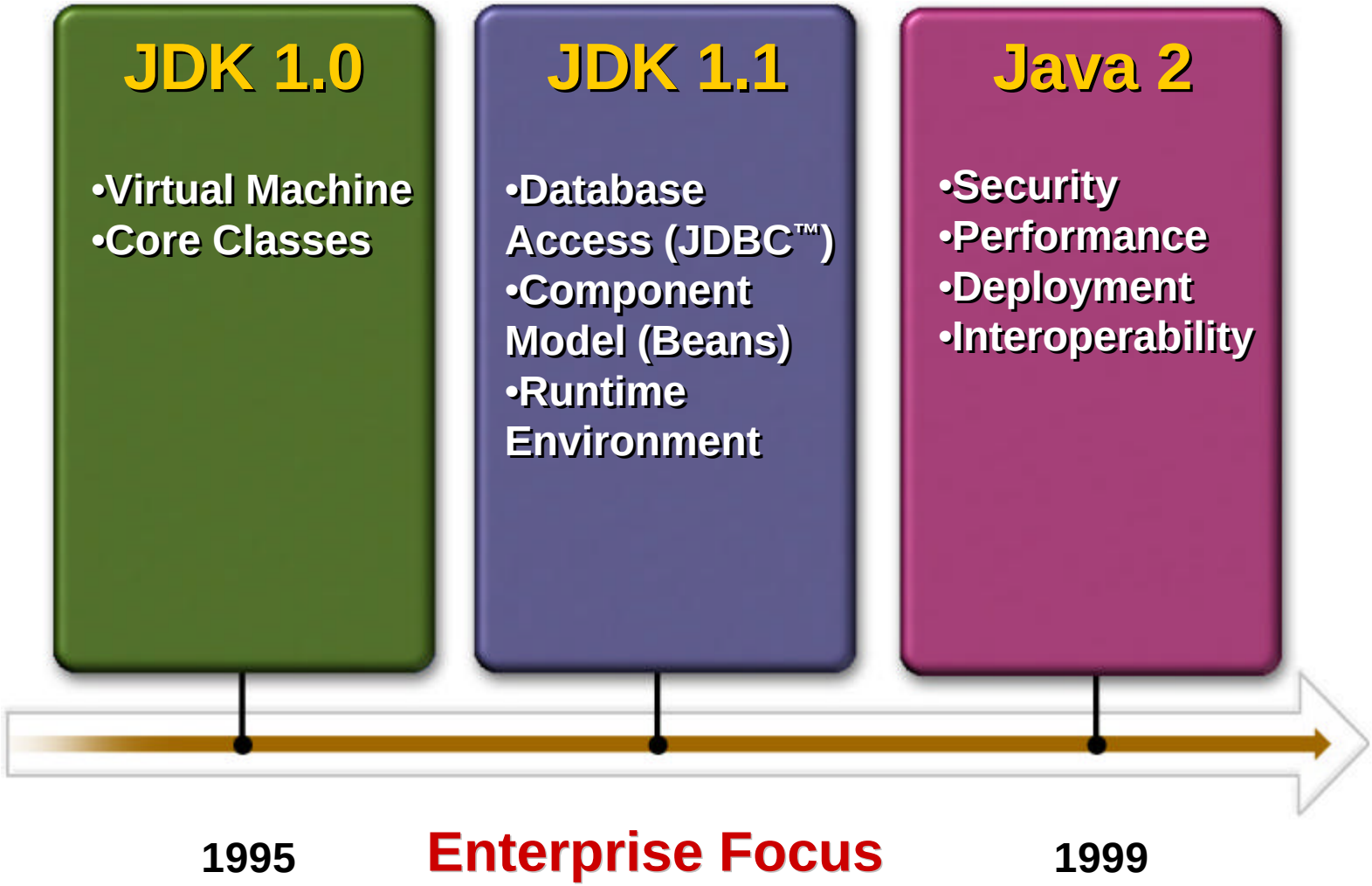
- **Device-independent GUI**
- **Web on the server side (Servlets, JSP, EJB, XML…)**

Prospettive

intelligent consumer electronics + smartcards

Marco Ronchetti - ronchet@altavista.it

Evolution of the Java Platform



Marco Ronchetti - ronchet@altavista.it

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. (Come il PPC emula il 68040 su Mac...).

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)

Marco Ronchetti - ronchet@altavista.it

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.

Marco Ronchetti - ronchet@altavista.it

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
        }
}
```

Marco Ronchetti - ronchet@altavista.it

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
   }
}
```

Marco Ronchetti - ronchet@altavista.it

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)

Marco Ronchetti - ronchet@altavista.it

Applets

Definizione:

Speciali programmi Java (senza “main”) richiamabili dal linguaggio HTML ed eseguiti in un contesto grafico.

Possono essere eseguiti da:

- Browser: un visualizzatore HTML dotato di interprete Java
- Programmi ad-hoc: visualizzatori di applets

  (Es. AppletViewer)

Marco Ronchetti - ronchet@altavista.it

Hello World (applet)

HelloWorldApplet.java

```
/* First Hello World Applet */
import java.awt.Graphics;
public class HelloWorldApplet extends java.applet.Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 5, 25);
    }
}
```

HelloWorldApplet.html

```
<HTML> <HEAD> <TITLE>Hello to everyone </TITLE> </HEAD> <BODY>
My Java Applet says:
<APPLET CODE="HelloWorldApplet.class" WIDTH=200 HEIGHT=75>
Il tuo browser non è abilitato per le applets </APPLET>
</BODY> </HTML>
```

Uso di JDK + Appletviewer

Compilazione e esecuzione:

$javac HelloWorldApplet.java

$appletviewer HelloWorldApplet.html

NOTA:
l'html NON viene interpretato, se non per la tag APPLET!

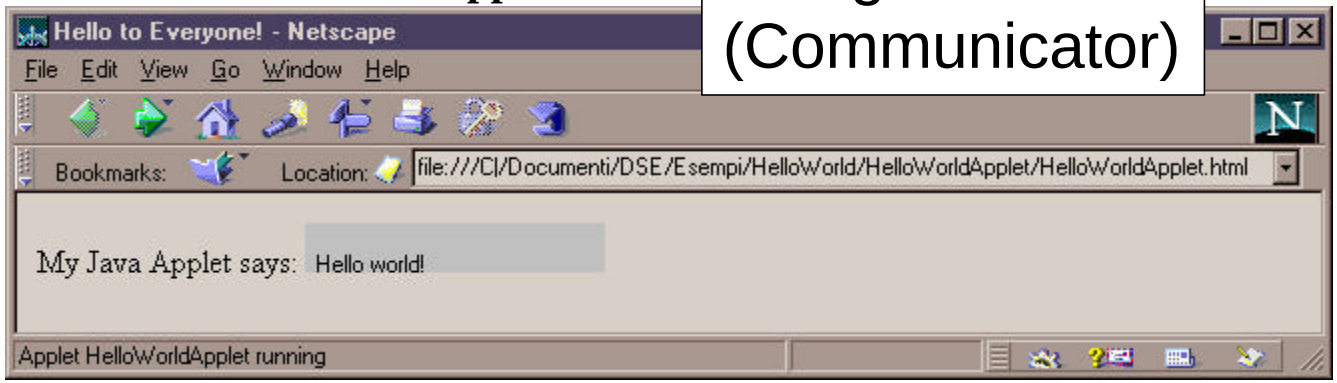
Marco Ronchetti - ronchet@altavista.it

Esecuzione tramite Browsers

Loading file:

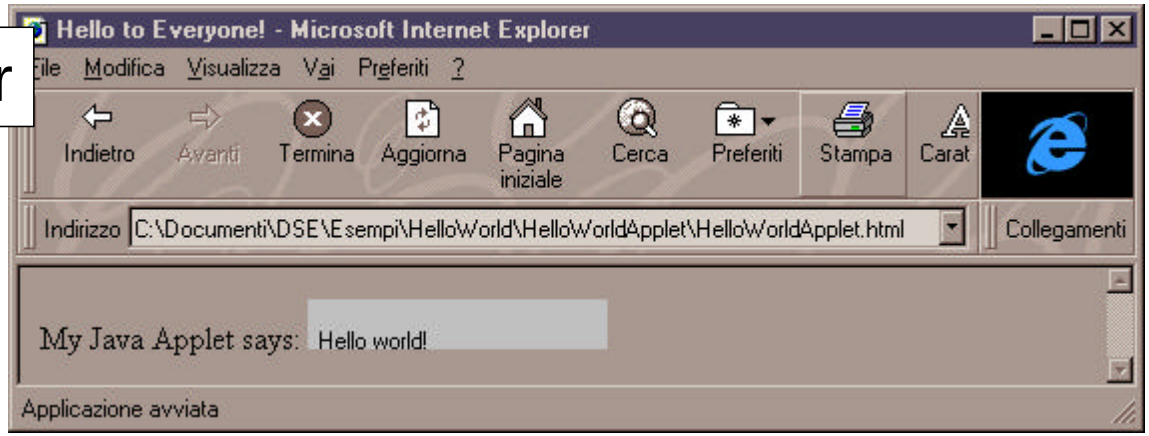
file:///.../HelloWorldApplet.html

Netscape Navigator (Communicator)



NOTA: l'html viene interpretato interamente

Ms Explorer



SunSoft Hotjava Mosaic...

HelloAgainApplet

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;
public class HelloAgainApplet extends java.applet.Applet {
    Font f = new java.awt.Font("TimesRoman",Font.BOLD,36);
    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString("Hello again!", 5, 50);
    }
}
```

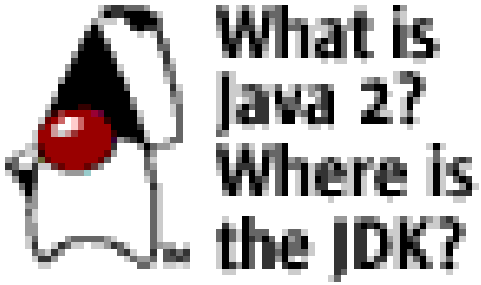
Source of: file:///C|/Documenti/DSE/Esempi/HelloWorld/HelloAgainA...

```
<HTML>
<HEAD>
<TITLE>Another Applet</TITLE>
</HEAD>
<BODY>
<P>My second Java applet says:<BR>
<APPLET CODE="HelloAgainApplet.class" WIDTH=200 HEIGHT=75>
</APPLET>
</BODY>
</HTML>
```

Marco Ronchetti - ronchet@altavista.it

Basic tools

http://www.java.sun.com/j2se



Java™ 2 Platform, Standard Edition (J2SE™)

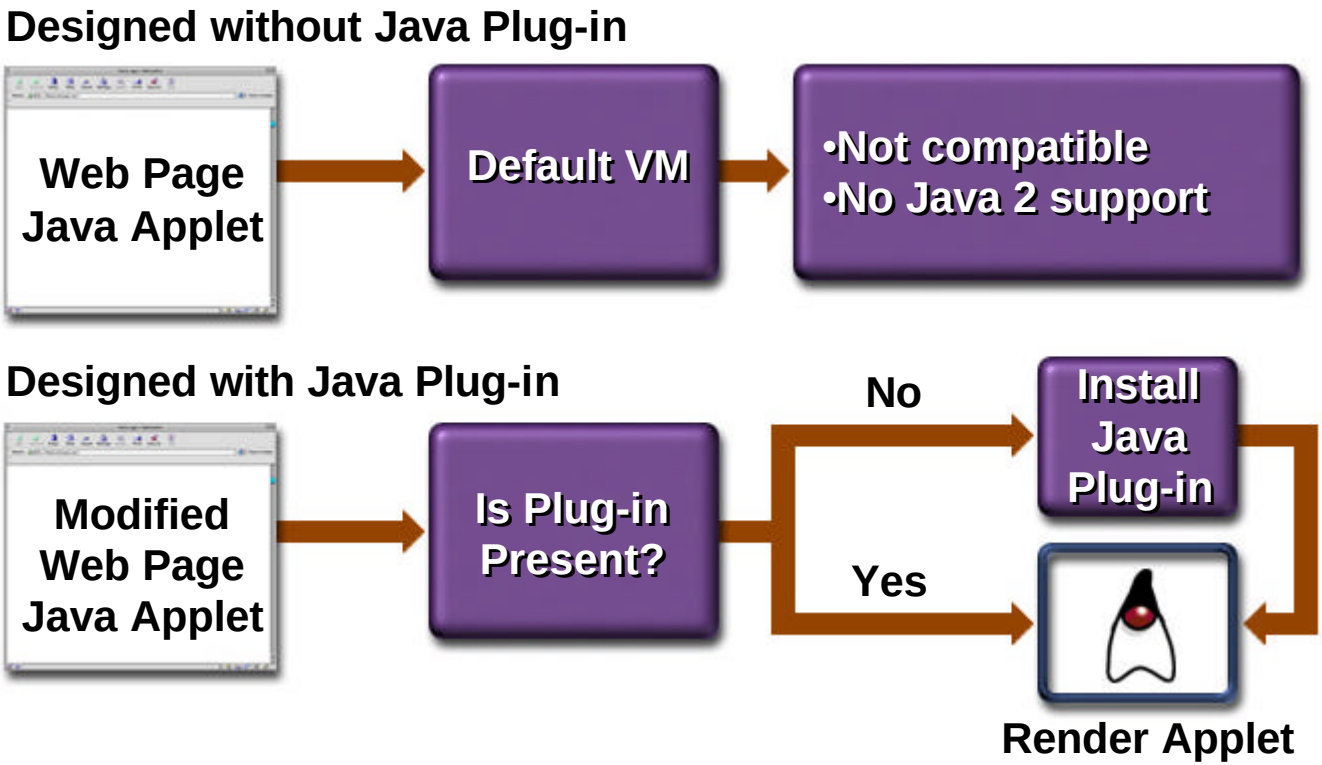
The essential Java 2 SDK, tools, runtimes, and APIs for developers writing, deploying, and running applets and applications in the Java programming language. Also includes earlier Java Development Kit versions JDK™ 1.1 and JRE 1.1

Marco Ronchetti - ronchet@altavista.it

Java 2 Standard Edition

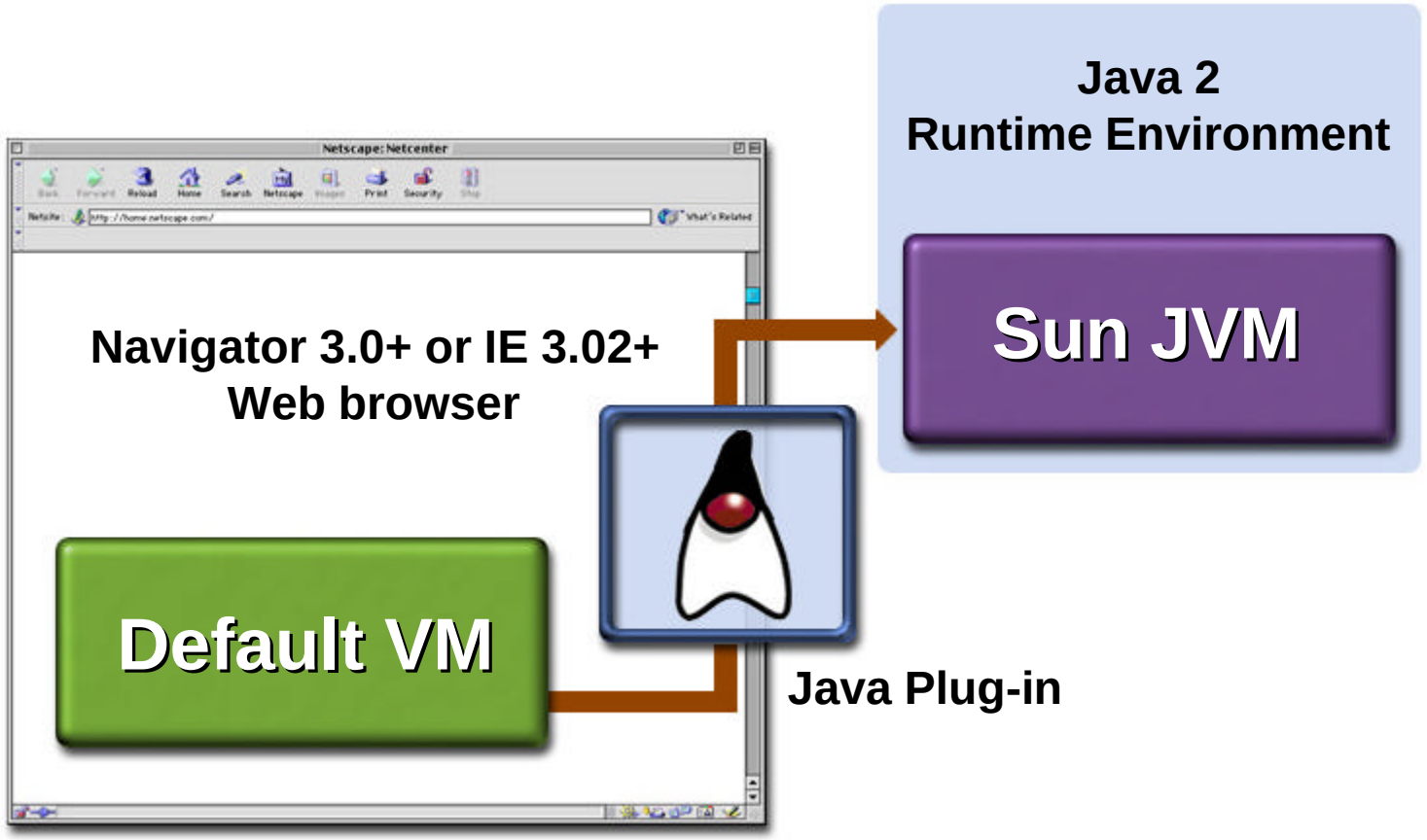
Marco Ronchetti - ronchet@altavista.it

Java 2 Standard Edition



Marco Ronchetti - ronchet@altavista.it

Java 2 Standard Edition



Marco Ronchetti - ronchet@altavista.it

Advanced development tool

Borland®
JBuilder™ 4

Pure Java. Visual Development. JAVA™2

http://www.borland.com/jbuilder/foundation/

Marco Ronchetti - ronchet@altavista.it

Advanced development tool

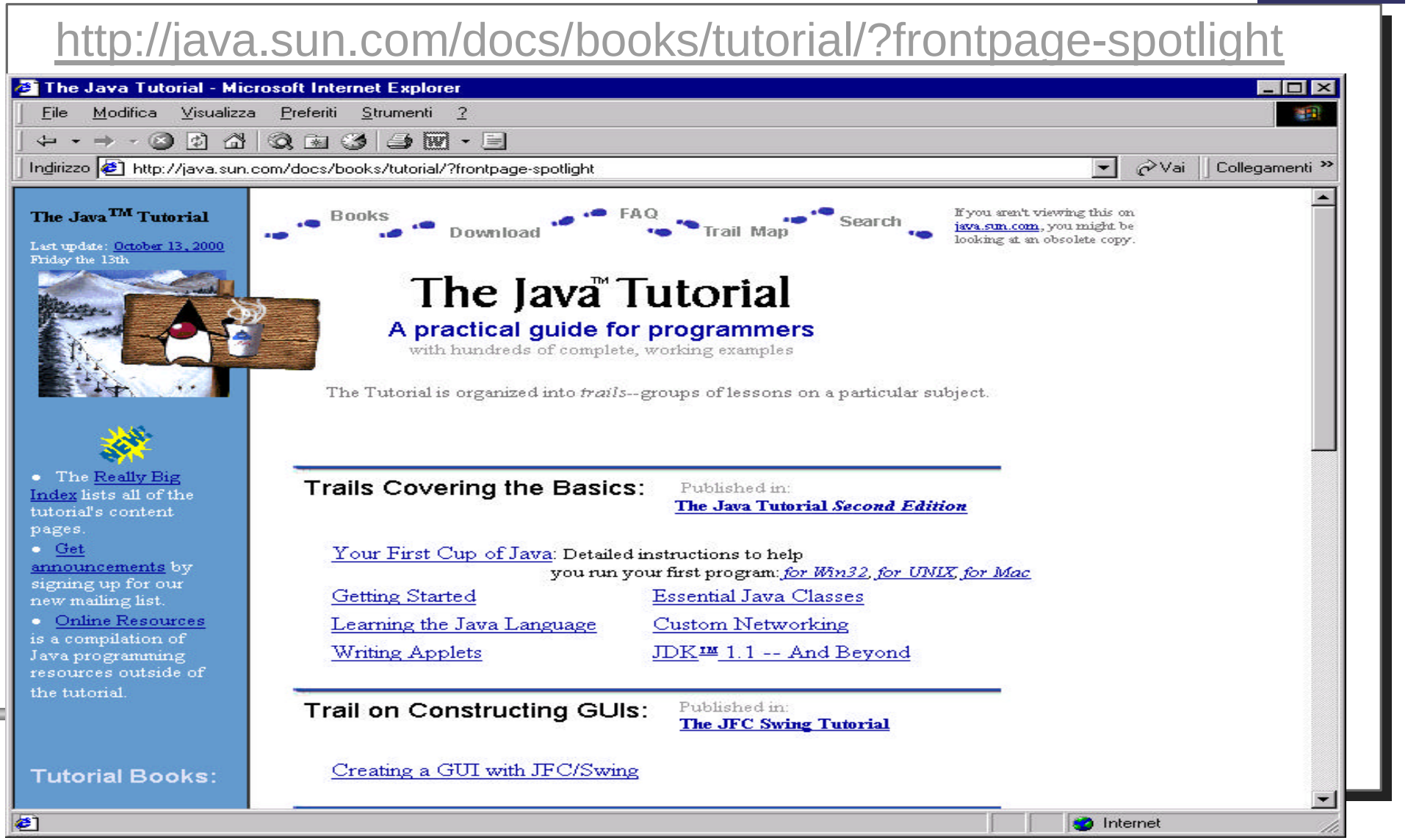
Receive a Free copy of the developerWorks' Developer Toolbox Sample CD, which includes:

- **IBM Websphere, Application Server**
- **IBM Developer Kits(JDKs) & Runtime Environment**
- **Advanced Edition IBM Websphere Studio,**
- **Entry Edition VisualAge for Java,**
- **Entry Edition DB2 Universal Database Enterprise Edition**
- **IBM HotMedia Software for the Java Environment**
- **IBM alphaWorks Java and XML Software Technologies**

http://www.alphaworks.ibm.com/aw.nsf/html/devtoolscd?open&l=edu,t=gr,p=toolsCD

Marco Ronchetti - ronchet@altavista.it

“The” Tutorials and examples



More Tutorials and examples

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

Free IBM developerWorks Online Java Tutorials Homepage

http://www2.software.ibm.com/developer/education.nsf/dw/java-onlinecourse-bytitle?open&l=edu,t=gr,p=javaeduhp

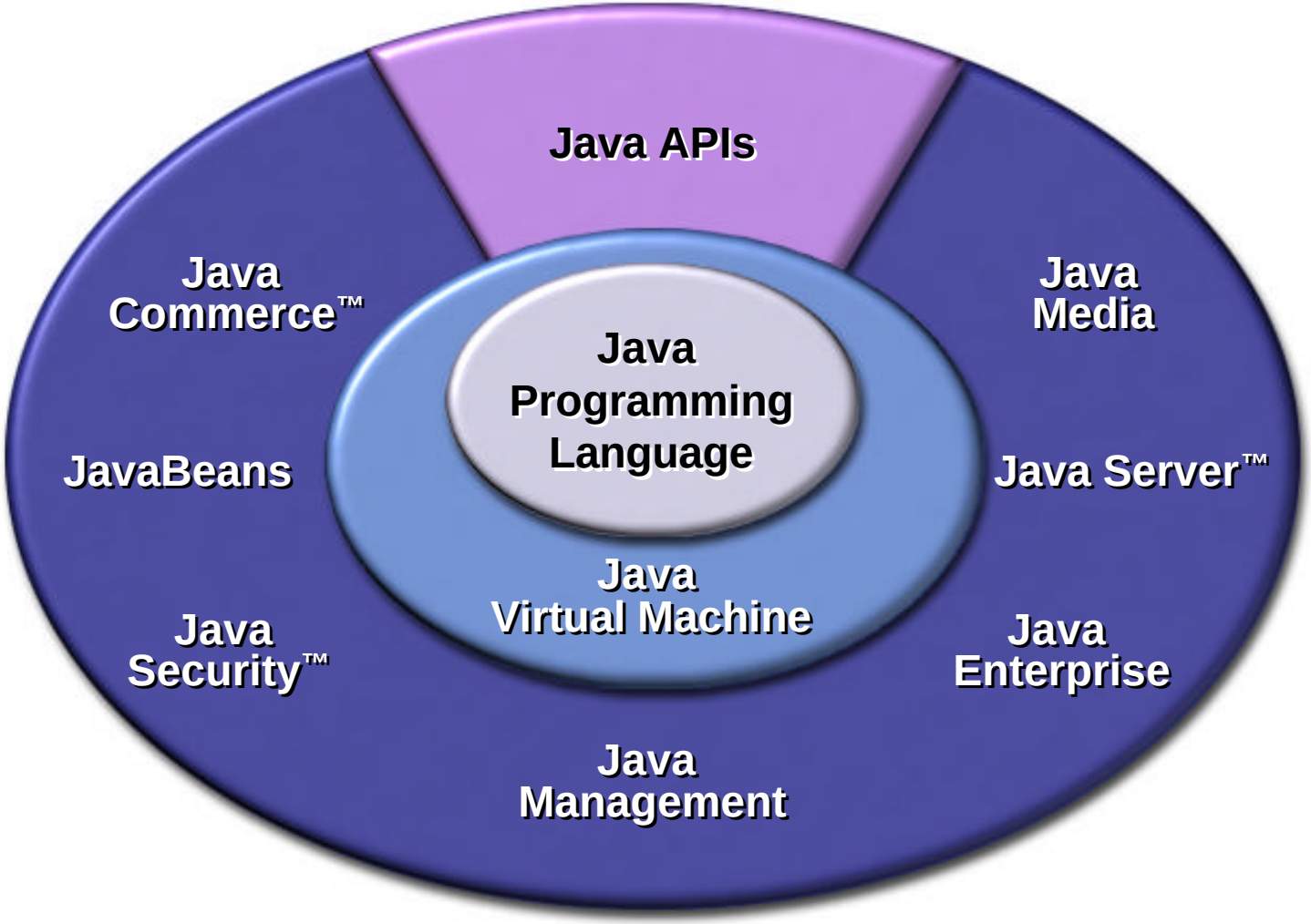
Marco Ronchetti - ronchet@altavista.it

Un buon libro…

Gratis in forma elettronica:
Thinking in Java
Bruce Eckel

http://www.eckelobjects.com/javabook.html

The Java Platform



Marco Ronchetti - ronchet@altavista.it

The Core Class libraries -1

The core API gives you the following features:

The Essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

Applets: The set of conventions used by Java applets.

Networking: URLs, TCP and UDP sockets, and IP addresses.

Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

Marco Ronchetti - ronchet@altavista.it

The Core Class libraries -2

Security: Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.

Software components: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.

The Core Class libraries -3

Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBC): Provides uniform access to a wide range of relational databases.

Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.

Marco Ronchetti - ronchet@altavista.it

JDK API


Package Index - Netscape
File Edit View Go Window Help
Java API Packages
package java.applet
package java.awt
package java.awt.datatransfer
package java.awt.event
package java.awt.image
package java.awt.peer
package java.beans
package java.io
package java.lang
package java.lang.reflect
package java.math
package java.net
package java.rmi
package java.rmi.dgc
package java.rmi.registry
package java.rmi.server
package java.security
package java.security.acl
package java.sql
package java.text
package java.util
package java.util.zip
File Edit View Go Window Help
Class Index
Boolean
Byte
Character
Class
ClassLoader
Compiler
Double
Float
Integer
Long
Math
Number
Object
Process
Runtime
SecurityManager
Short
String
StringBuffer
System
Thread
ThreadGroup
Throwable
Void
Class java.lang.String - Netscape
File Edit View Go Window Help
Class java.lang.String
java.lang.Object
|
+----java.lang.String
public final class String
extends Object
implements Serializable
A general class of objects to represent character Strings. Strings are constant, their values cannot be changed after creation. The compiler makes sure that each String constant actually results in a String object. 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);
Document: Done

JDK API

Class java.lang.String - Netscape

File Edit View Go Window Help

Constructor Index

- **String**()
  Constructs a new empty String.
- **String**(byte[])
  Construct a new String by converting the specified array of bytes using the platform's default character encoding.
- **String**(byte[], int)
  Constructs a new String whose value is the specified array of bytes. **Deprecated.**
- **String**(byte[], int, int)
  Construct a new String by converting the specified subarray of bytes using the platform's default character encoding.
- **String**(byte[], int, int, int)
  Constructs a new String whose initial value is the specified subarray of bytes. **Deprecated.**
- **String**(byte[], int, int, String)
  Construct a new String by converting the specified subarray of bytes using the specified character encoding.
- **String**(byte[], String)
  Construct a new String by converting the specified array of bytes using the specified character encoding.
- **String**(char[])
  Constructs a new String whose initial value is the specified array of

Document: Done

Class java.lang.String - Netscape

File Edit View Go Window Help

Method Index

- **charAt**(int)
  Returns the character at the specified index.
- **compareTo**(String)
  Compares this String to another specified String.
- **concat**(String)
  Concatenates the specified string to the end of this String and returns a new String object representing the concatenation.
- **copyValueOf**(char[])
  Returns a String that is equivalent to the specified character array.
- **copyValueOf**(char[], int, int)
  Returns a String that is equivalent to the specified character array.
- **endsWith**(String)
  Determines whether the String ends with some suffix.
- **equals**(Object)
  Compares this String to the specified object.
- **equalsIgnoreCase**(String)
  Compares this String to another object.
- **getBytes**()
  Convert this String into bytes according to the platform's default character encoding, storing the result into a new byte array.
- **getBytes**(int, int, byte[], int)

Document: Done

JDK API

Class java.lang.String - Netscape

File Edit View Go Window Help

Constructors

- **String**

```
public String()
```

Constructs a new empty String.

- **String**

```
public String(String value)
```

Constructs a new String that is a copy of the specified String.

Parameters:
value - the initial value of the String

- **String**

```
public String(char value[])
```

Constructs a new String whose initial value is the specified array of characters.

Parameters:
value - the initial value of the String

Document: Done

Class java.lang.String - Netscape

File Edit View Go Window Help

Methods

- **length**

```
public int length()
```

Returns the length of the String. The length of the String is equal to the number of 16 bit Unicode characters in the String.

- **charAt**

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

Returns the character at the specified index. An index ranges from `0` to `length() - 1`.

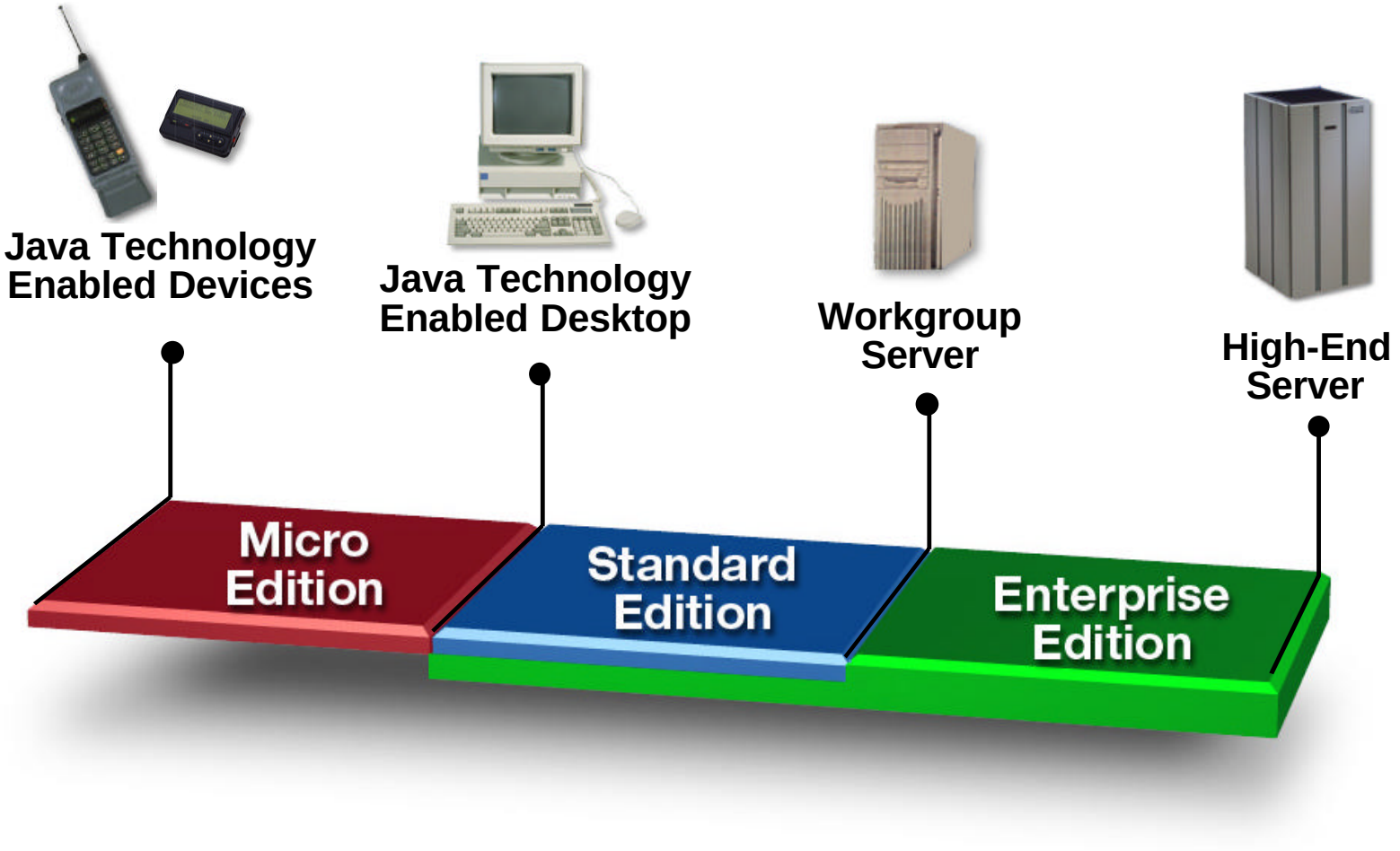
Parameters:
index - the index of the desired character
Throws: StringIndexOutOfBoundsException
If the index is not in the range `0` to `length()-1`.

- **getChars**

```
public void getChars(int srcBegin,
```

Document: Done

Java 2 Platform editions

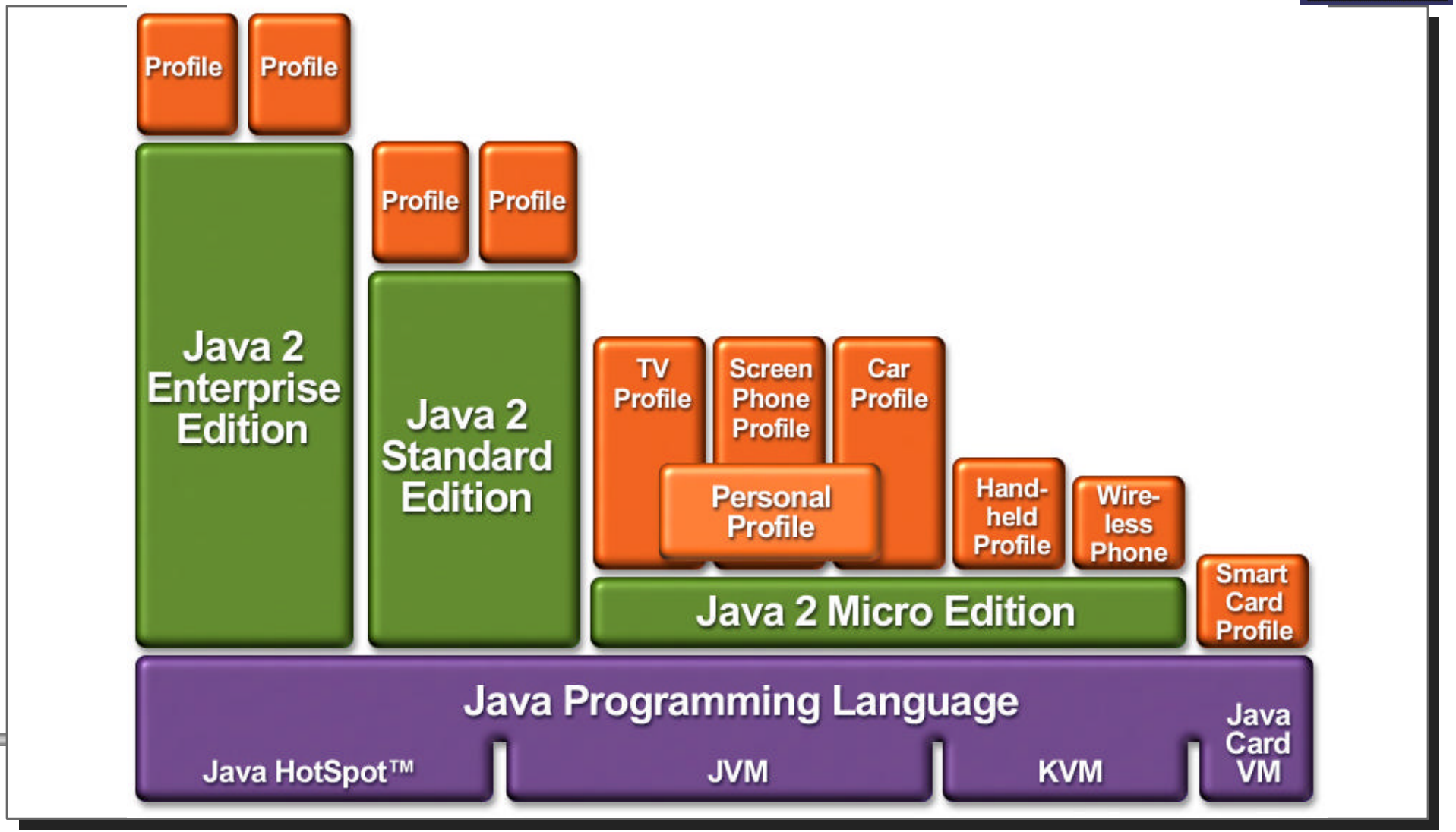


Marco Ronchetti - ronchet@altavista.it

Java 2 Platform

Marco Ronchetti - ronchet@altavista.it

Java 2 Platform



Marco Ronchetti - ronchet@altavista.it

Java Card Platform

Smart card sono piccoli computer Clock and power from the reader

Clock speed parte da 3.5 MHz

I/O parte da 9600 baud

processori a 8-, 16-, 32- bit

La piattaforma minima:

512 bytes RAM (I/O, stack)

24 KB ROM (VM, applets, native functions)

8 KB EEPROM (applets, object heap)

8-bit processor

Marco Ronchetti - ronchet@altavista.it

Sicurezza 1.1 per Applets

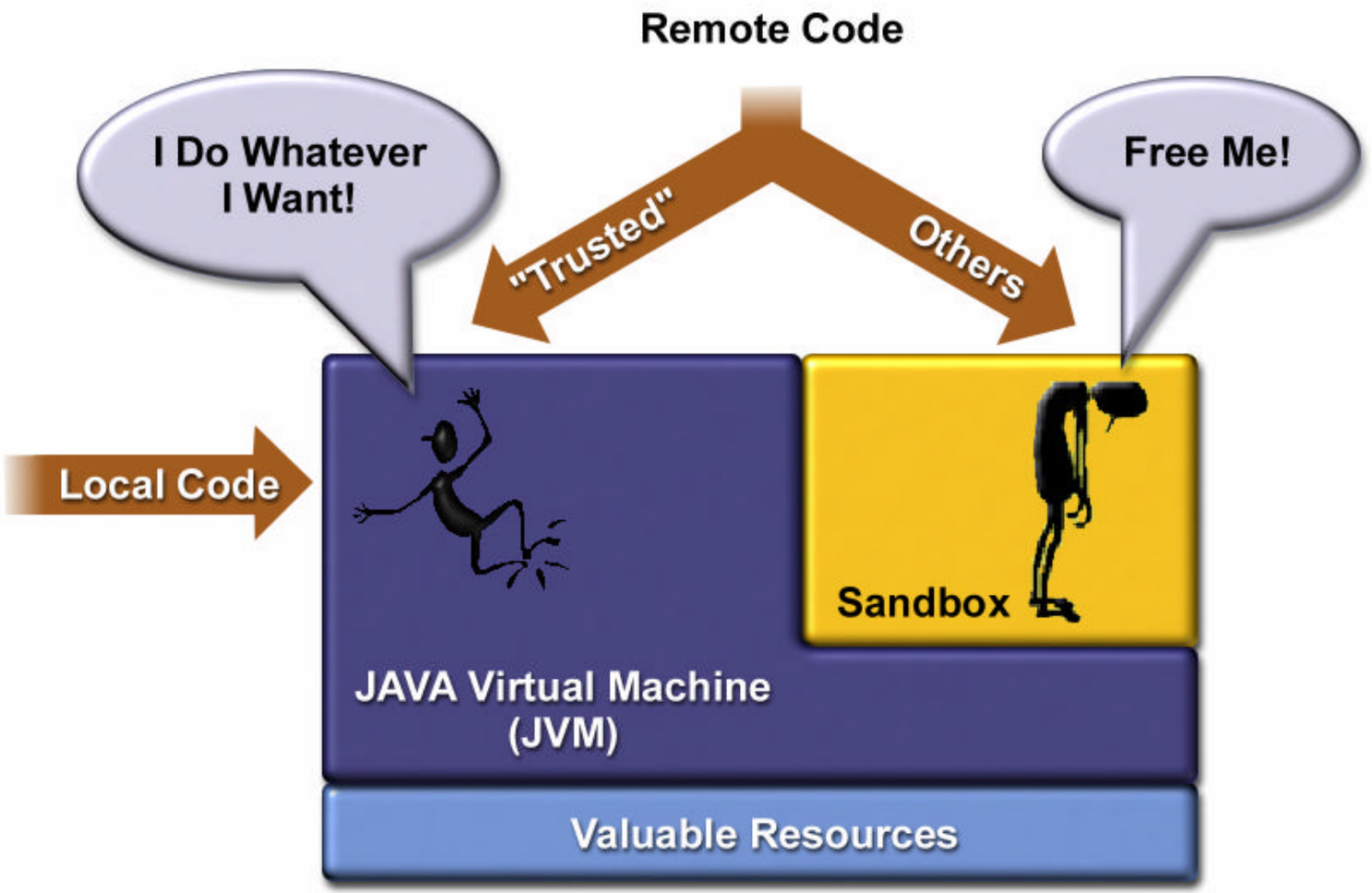
Il bytecode viene verificato per impedire usi impropri.

Le applet hanno possibilità limitate:

- accesso al file system: **NO**
- collegamento a siti remoti diversi da quello di provenienza: **NO**
- finestre con la scritta "**this is an insecure applet window**"

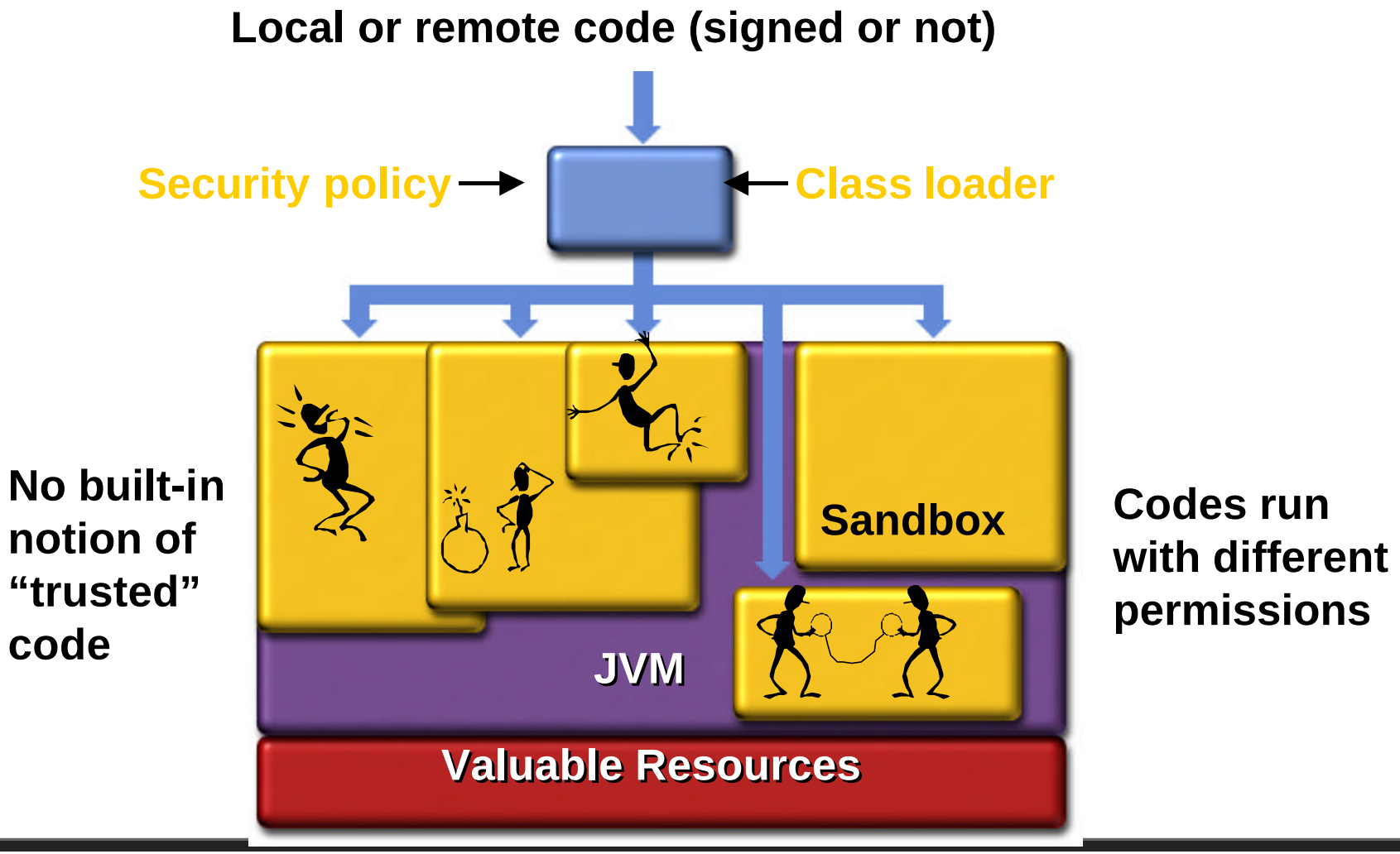
Marco Ronchetti - ronchet@altavista.it

Il modello di sicurezza 1.1



Marco Ronchetti - ronchet@altavista.it

Il modello di sicurezza 1.2



Differenze tra Java e C++

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

Marco Ronchetti - ronchet@altavista.it

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!

Marco Ronchetti - ronchet@altavista.it

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.

Marco Ronchetti - ronchet@altavista.it

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

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
*/
```

Marco Ronchetti - ronchet@altavista.it

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

Marco Ronchetti - ronchet@altavista.it

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”

Marco Ronchetti - ronchet@altavista.it

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.

Marco Ronchetti - ronchet@altavista.it

Operatori

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

Marco Ronchetti - ronchet@altavista.it

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. Further, Java doesn't consider [] (array access) and . (field access) to be operators.

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.

Marco Ronchetti - ronchet@altavista.it

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.

Marco Ronchetti - ronchet@altavista.it

Operatori

Because all integral types in Java are signed values, the Java

>>

operator is defined to do a right shift with sign extension.

The

>>>

operator treats the value to be shifted as an unsigned number and shifts the bits right with zero extension.

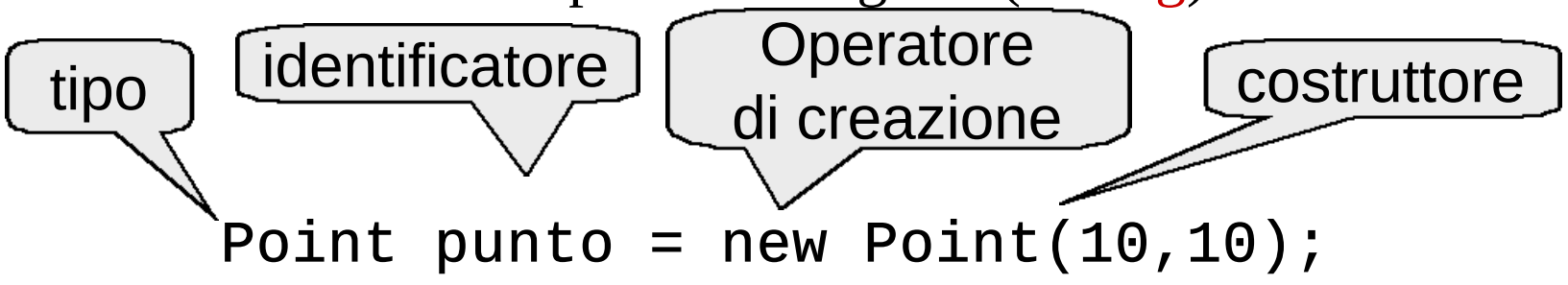
Marco Ronchetti - ronchet@altavista.it

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.

Marco Ronchetti - ronchet@altavista.it

“Java non ha i puntatori”

Ma è vero?

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

l’identificatore di un oggetto (“punto”)
é 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

Marco Ronchetti - ronchet@altavista.it

Variabili di istanza

```
/* creating objects then testing and modifying thier IVs. */

import java.awt.Point;
class TestPoint {
  public static void main (String args[]) {
    Point thePoint = new Point(10,10);
    System.out.println("x is " + thePoint.x);
    System.out.println("y is " + thePoint.y);

    System.out.println("Setting x to 5.");
    thePoint.x = 5;
    System.out.println("Setting y to 15.");
    thePoint.y = 15;
    System.out.println("x is " + thePoint.x);
    System.out.println("y is " + thePoint.y);
  }
}
```

Marco Ronchetti - ronchet@altavista.it

Una semplice classe

Una semplice classe con:

- due variabili d'istanza
- un costruttore
- un metodo

1

```
class Person {
  String name;
  int age;

  Person(String n, int a) {
    name = n;
    age = a;
  }

  void printPerson() {
    System.out.print("Hi, my name is " + name);
    System.out.println(". I am " + age + " years old.");
  }
```

2

```
  public static void main (String args[]) {
    Person p;

    p = new Person("Laura", 20);
    p.printPerson();
    System.out.println("--------");
    p = new Person("Tommy", 3);
    p.printPerson();
    System.out.println("--------");
  }
}
```

Method overloading 1

Un esempio di overloading (polimorfismo)

```
/* rectangle class
to demonstrate
polymorphic methods */

import java.awt.Point;

class MyRect {
  int x1 = 0;
  int y1 = 0;
  int x2 = 0;
  int y2 = 0;
```

1

```
MyRect buildRect(int x1, int y1, int x2, int y2) {
    this.x1 = x1;
    this.y1 = y1;
    this.x2 = x2;
    this.y2 = y2;
    return this;
  }
MyRect buildRect(Point topLeft, Point bottomRight) {
    x1 = topLeft.x;
    y1 = topLeft.y;
    x2 = bottomRight.x;
    y2 = bottomRight.y;
    return this;
}
MyRect buildRect(Point topLeft, int w, int h) {
    x1 = topLeft.x;
    y1 = topLeft.y;
    x2 = (x1 + w);
    y2 = (y1 + h);
    return this;
}
```

2

Marco Ronchetti - ronchet@altavista.it

Method overloading 2

```
  void printRect(){
    System.out.print("MyRect: <" + x1 + ", " + y1);
    System.out.println(", " + x2 + ", " + y2 + ">");
  }
```

3

```
public static void main (String args[]) {
    MyRect rect = new MyRect();
    System.out.println("Calling buildRect with coordinates 25,25 50,50:");
    rect.buildRect(25, 25, 50, 50);
    rect.printRect();
    System.out.println("----------");
    System.out.println("Calling buildRect w/points (10,10), (20,20):");
    rect.buildRect(new Point(10,10), new Point(20,20));
    rect.printRect();
    System.out.println("----------");
    System.out.print("Calling buildRect w/1 point (10,10),");
    System.out.println(" width (50) and height (50)");
    rect.buildRect(new Point(10,10), 50, 50);
    rect.printRect();
    System.out.println("----------");
  }
}
```

4

Overloading di costruttore


```
/* create news dates using various different ways */
import java.util.Date;

class CreateDates {
  public static void main (String args[]) {
    Date d1, d2, d3;

    d1 = new Date();
    System.out.println("Date 1: " + d1);

    d2 = new Date(71, 7, 1, 7, 30);
    System.out.println("Date 2: " + d2);

    d3 = new Date("April 3 1993 3:24 PM");
    System.out.println("Date 3: " + d3);
  }
}
```

Marco Ronchetti - ronchet@altavista.it

Riferimenti ad Oggetti


```
/* testing references; change one, the other changes.*/
import java.awt.Point;

class ReferencesTest {
  public static void main (String args[]) {
    Point pt1, pt2;

    pt1 = new Point(100,100);
    pt2 = pt1;

    pt1.x = 200;
    pt1.y = 200;

    System.out.println("Point1: " + pt1.x + "," + pt1.y);
    System.out.println("Point2: " + pt2.x + "," + pt2.y);
  }
}
```

Marco Ronchetti - ronchet@altavista.it

Confronto tra oggetti


```
/* compare two different strings using equals */
class EqualsTest {
  public static void main (String args[]) {
    String str1, str2;

    str1 = "she sells sea shells by the sea shore.";
    str2 = str1;

    System.out.println("String1: " + str1);
    System.out.println("String2: " + str2);
    System.out.println("Same object? " + (str1 == str2));

    str2 = new String(str1);

    System.out.println("String1: " + str1);
    System.out.println("String2: " + str2);
    System.out.println("Same object? " + (str1 == str2));
    System.out.println("Same value? " + str1.equals(str2));
  }
}
```

Marco Ronchetti - ronchet@altavista.it

subclassing & overriding

super indica la superclasse

this indica l'oggetto corrente

```
import java.awt.Point;
class NamedPoint extends Point {
  String name;
  NamedPoint(int x, int y, String name) {
    super(x,y);
    this.name = name;
  }

  public void printName() {
     System.out.println("Name is " + this.name);
  }

  public static void main (String arg[]) {
    NamedPoint np = new NamedPoint(5, 5, "SmallPoint");

    System.out.println("X is " + np.x);
    System.out.println("Y is " + np.y);
    np.printName();
  }
}
```

Classi e sotto

```
/* superclass for testing
   method overrides */
class PrintClass {
  int x = 0;
  int y = 1;

  void printMe() {
    System.out.println("X is " + x + ", Y is " + y);
    System.out.println("I am an instance of the class" +
           this.getClass().getName());
  }
}
```

1

```
class PrintSubClass extends PrintClass {
  int z = 3;
  public static void main (String args[]) {
    PrintSubClass obj = new PrintSubClass();
    obj.printMe();
  }
}
```

3

```
/* subclass for testing method overrides */
class PrintSubClass2 extends PrintClass {
  int z = 3;
  void printMe() {
    System.out.println("x is " + x + ", y is " + y + ", z is " + z);
    System.out.println("I am an instance of the class " +
        this.getClass().getName());
  }
  public static void main (String args[]) {
    PrintSubClass2 obj = new PrintSubClass2();
    obj.printMe();
  }
}
```

2

Marco Ronchetti - ronchet@altavista.it

Reference Type Summary

- All objects and arrays are handled by reference in Java. (Those object references are passed-by-value to methods, however.)
- The = and == operators assign and test references to objects. Use clone() and equals() to actually copy or test the objects themselves.
- The necessary referencing and dereferencing of objects and arrays is handled automatically by Java.
- A reference type can never be cast to a primitive type.
- A primitive type can never be cast to a reference type.
- There is no pointer arithmetic in Java.
- There is no sizeof operator in Java.
- null is a special value that means "no object" or indicates an absence of reference.

Marco Ronchetti - ronchet@altavista.it

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()

Marco Ronchetti - ronchet@altavista.it

memory management

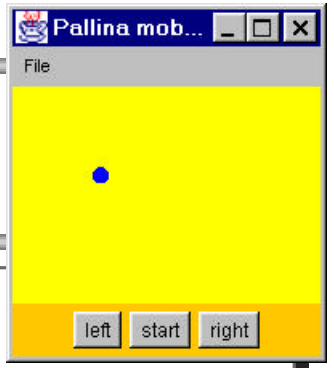
Operazioni da eseguirsi alla nascita di un oggetto vanno definite nel metodo “costruttore”.

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)

Marco Ronchetti - ronchet@altavista.it

Learning by example 1



Requisiti del progetto:

Una finestra dotata di menu e contenente una pallina mobile in orizzontale e quattro bottoni: sinistra, destra, start, stop.

Sinistra e destra muovono la pallina di una quantità random compresa tra 1 e 5 pixel a sx e dx.

La pallina puo' (in una versione del programma) lasciare traccia del proprio cammino.

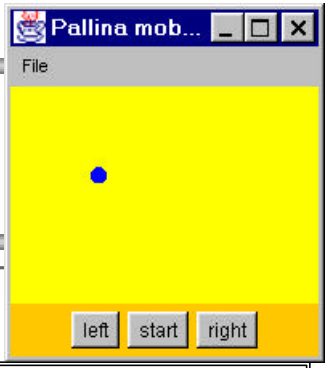
Start avvia un moto browniano della pallina, stop lo ferma.

Il menu contiene la voce di "Exit" che termina l'esecuzione del programma.

Il programma può anche terminare chiudendo la finestra tramite il solito bottone con una x in alto a destra.

Marco Ronchetti - ronchet@altavista.it

Learning by example 2



class DemoUnoAWT

```
package milano;
import java.awt.*;

public class DemoUnoAWT {

  DemoUnoAWT(){
    FinestraAWT finestra=new FinestraAWT();
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    finestra.setLocation((screenSize.width - finestra.larghezza) / 2, (screenSize.height - finestra.altezza) / 2);
    finestra.setVisible(true);
  }

  public static void main(String a[ ]){
    new DemoUnoAWT();
  }
}
```

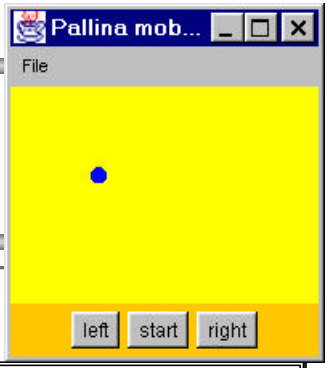
Classe (definizione)

Costruttore

main program

Marco Ronchetti – ronchet@altavista.it

Learning by example 3



class DemoUnoAWT

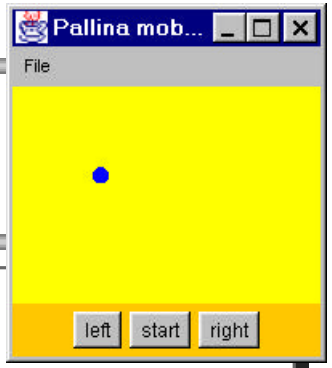
```
package milano;
import java.awt.*

public class Dem    noAWT {

  DemoUno WT(){
    FinestraAWT finestra=new FinestraAWT();
    Dimension screenSize = Toolkit.getDefau Toolkit().getScreenSize();
    finestra.setLocation((screenSize.width - finestra.  bezza) / 2, (screenSize.height - finestra.altezza) / 2);
    finestra.setVisible(true);
  }

  public static void main(String a[]){
    new DemoUnoAWT();
  }
}
```

Classe (tipo)

Oggetto (Variabile)

Costruttore (invocazione)

ample 4




```
package milano;
import java.awt.*;
public class Schermo extends Panel{

  int centroCerchioY=50;
  int centroCerchioX=50;

  public Schermo() {
    this.setForeground(Color.blue);
    this.setBackground(Color.yellow);
  }
  public void  moveCentroCerchioX(int dx){
    centroCerchioX += dx;
  }
  public void paint(Graphics g) {
      g.fillOval(centroCerchioX,centroCerchioY,10,10);
}
```

ereditarietà

class Schermo extends Panel

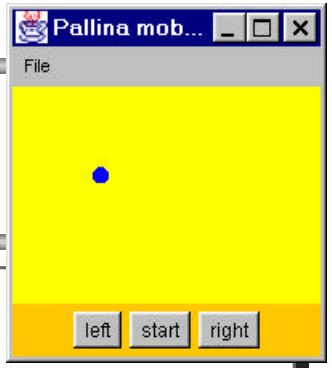
Variabili di istanza
(comuni a tutti i metodi)

Costruttore

```
/*public void update(Graphics g) {
    //attivando questa funzione la pallina
    // lascia la traccia
    paint(g);
  }  */
}
```

Marco Ronchetti - ronchet@altavista.it

Learning by example 5



class FinestraAWT extends Frame

```
package milano;
import java.awt.*;
import java.awt.event.*;
public class FinestraAWT extends Frame{

  public int larghezza=200;
  public int altezza=200;
  Schermo panel2;

  FinestraAWT() {
    super("Pallina mobile"); // setta il titolo della finestra
    inizializza();
  }
 …
```

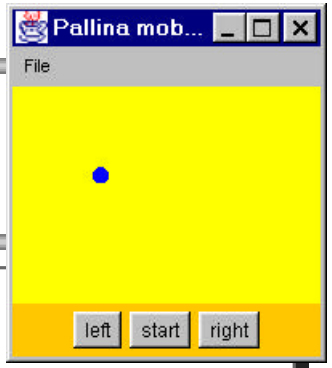
ereditarietà

Variabili di istanza
(comuni a tutti i metodi)

Costruttore

Marco Ronchetti - ronchet@altavista.it

Learning by example 6



Registra l'ascoltatore per gli eventi di finestra

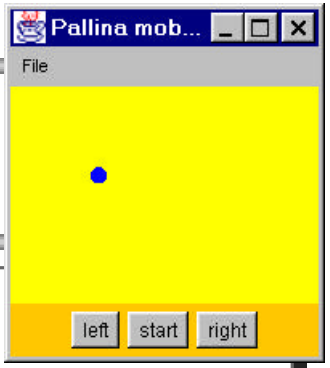
Inner class

Metodo per reagire all'evento "chiusura della finestra)

Schermo è la classe (panel) che disegna la pallina

```
class FinestraAWT e
  void inizializza(){
    setSize(larghezza,altezza);
    this.addWindowListener(new WindowAdapter() {
     public void windowClosing(WindowEvent e){
        System.exit(1);
     }
    }) ;
    // Crea il pannello inferiore ----------------------
    creaPannelloConBottoni();
    // Crea il pannello centrale ----------------------
    panel2 = new Schermo();
    this.add(panel2, BorderLayout.CENTER);
    // Aggiungi i menu ---------------------------------
    creaMenu();
  }
```

Learning by example 7



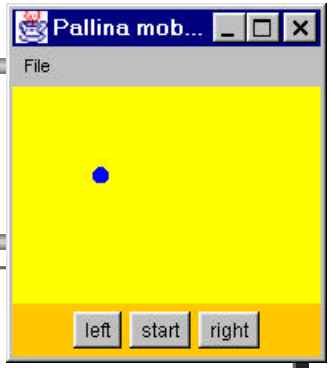

```
void creaPannelloConBottoni() {
  // Crea il bottone di left ---------------------------
  Button left = new Button();
  left.setLabel("left");
  left.addActionListener(
    new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      bottoneLeftPremuto();
    }
  });
  // Crea il bottone di right ---------------------------
  Button right = new Button();
  right.setLabel("right");
  right.addActionListener(
    new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      bottoneRightPremuto();
    }
  });
```

class FinestraAWT extends Frame

```
// Crea il bottone di start ---------------------------
    Button start = new Button();
    start.setLabel("start");
    start.addActionListener(
      new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        bottoneStartPremuto2();
      }
    });

    Panel panel1 = new Panel();
    panel1.setBackground(Color.orange);
    panel1.add(left);
    panel1.add(start, null);
    panel1.add(right, null);
    this.add(panel1, BorderLayout.SOUTH);
  }
```

Marco Ronchetti - ronchet@altavista.it

Learning by example 8



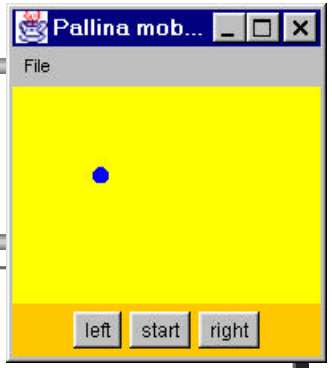
class FinestraAWT extends Frame

```
void creaMenu() {
   MenuBar menuBar1 = new MenuBar();
   Menu file = new Menu("File");
   MenuItem exit = new MenuItem("Exit");
   exit.addActionListener(new ActionListener()  {
    public void actionPerformed(ActionEvent e) {
       System.exit(1);
    }
   });
   file.add(exit);
   menuBar1.add(file);
   this.setMenuBar(menuBar1);
 }
```

Definizione dei menu

Marco Ronchetti - ronchet@altavista.it

Learning by example 9



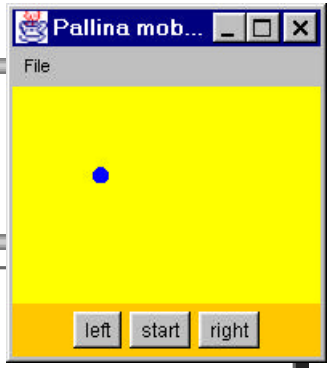
class FinestraAWT extends Frame

```
void creaMenu() {
   MenuBar menuBar1 = new MenuBar();
   Menu file = new Menu("File");
   MenuItem exit = new MenuItem("Exit");
   exit.addActionListener(new ActionListener()  {
    public void actionPerformed(ActionEvent e) {
       System.exit(1);
    }
   });
   file.add(exit);
   menuBar1.add(file);
   this.setMenuBar(menuBar1);
 }
```

Definizione dei menu

Marco Ronchetti - ronchet@altavista.it

Learning by example 10



class FinestraAWT extends Frame

```
int centroCerchioX=larghezza/2;
 int centroCerchioY=altezza/2;
 // ---------------- Gestione Delle Azioni
 void bottoneLeftPremuto() {
   int dx=(int)(1+5*Math.random());
   panel2.moveCentroCerchioX(-dx);
   panel2.repaint();
 }
 void bottoneRightPremuto() {
   int dx=(int)(1+5*Math.random());
   panel2.moveCentroCerchioX(dx);
   panel2.repaint();
 }
```

random:
double compreso tra 0 e 1

Marco Ronchetti - ronchet@altavista.it

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
```

Marco Ronchetti - ronchet@altavista.it

java.util

Marco Ronchetti - ronchet@altavista.it

Metodi della classe Stringa


```
class TestString {
  public static void main (String args[]) {
    String str = “Nel mezzo del cammin di nostra vita";

    System.out.println("The string is: " + str);
    System.out.println("Length of this string: " + str.length());
    System.out.println("The character at position 5: " + str.charAt(5));
    System.out.println("The substring from positions 11 to 17: " +
                       str.substring(11,17));
    System.out.println("The index of the character d: " +str.indexOf('d'));
    System.out.println("The index of the beginning of the substring \”cammin\": "
                       + str.indexOf(”cammin"));
    System.out.println("The string in upper case: " + str.toUpperCase());
  }
```

Vedere le API!

Marco Ronchetti - ronchet@altavista.it

Parametri di ingresso

I parametri del main sono inclusi in un vettore di String

```
/* 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);
  }
}
```

Marco Ronchetti - ronchet@altavista.it

Valore in uscita

Note that main() must be declared to return void. Thus you cannot return a value from your Java program with a return statement in main(). If you need to return a value, call System.exit() with the desired integer value, as we've done in the example.

Note that the handling and interpretation of this exit value are, of course, operating-system dependent. System.exit() causes the Java interpreter to exit immediately, whether or not other threads are running.

An Echo Program in Java

```
public class echo {
    public static void main(String argv[]) {
        for(int i=0; i < argv.length; i++)
            System.out.print(argv[i] + " ");
        System.out.print("\n");
        System.exit(0);
    }
}
```

Environment

The Java API does not allow a Java program to read operating system environment variables because they are platform-dependent. However, Java defines a similar, platform-independent mechanism, known as the system properties list, for associating textual values with names.

A Java program can look up the value of a named property with the System.getProperty() method:

```
String homedir=System.getProperty("user.home");
String debug=System.getProperty("myapp.debug");
```

The Java interpreter automatically defines a number of standard system properties when it starts up. You can insert additional property definitions into the list by specifying the -D option to the interpreter:

```
% java -Dmyapp.debug=true myapp
```

Marco Ronchetti - ronchet@altavista.it

Modificatori: visibilita'

public	visibile da tutti
(non def.)	visibile da tutti nello stesso package
protected	visibile dalle sottoclassi
private	nascosta da tutti

Uso di metodi "di accesso":

```
public class  ACorrectClass {
    private String  aUsefulString;

    public String  aUsefulString() { // "get" the value
        return aUsefulString;
    }

    private protected void aUsefulString(String s) {// "set" the value
        aUsefulString = s;
        /* performSomeImportantBookkeepingOn(s); */
    }
}
```

Modificatori: final

Variabili dichiarate final sono costanti.

Metodi dichiarati final non possono essere sovrascritti

Classi dichiarate final non possono essere subclassate.

Marco Ronchetti - ronchet@altavista.it

Modificatori: abstract

Metodi dichiarati abstract devono essere sovrascritti

Classi dichiarate abstract devono essere subclassate.

“interface” e’ una classe astratta costituita solo di metodi astratti

Marco Ronchetti - ronchet@altavista.it

Modificatori: static

Variabili e metodi associati ad una Classe anziche' ad un Oggetto sono definiti "static".

I metodi possono essere richiamati senza creare una istanza.

Es: System.out.println(...)

Le variabili servono come singola variabile condivisa tra le varie istanze

```
public class  InstanceCounterTester
              extends InstanceCounter {
    public static void  main(String argv[]) {
        for (int  i = 0;  i < 10;  ++i)
            new InstanceCounter();
            System.out.println("made " +
               InstanceCounter.instanceCount());
    }
}
```

1

```
public class  InstanceCounter {
    private static int
        instanceCount = 0; // a class variable
    private protected static int
        instanceCount() {    // a class method
        return instanceCount;
    }
    private static void  incrementCount() {
        ++instanceCount;
    }
    InstanceCounter() {
        InstanceCounter.incrementCount();
    }
}
```

2

Exceptions

Le eccezioni interrompono il normale flusso di un programma Java per riportare (ad un modulo di livello elevato) notizia di condizioni eccezionali che possono essere avvenute ad un piu' basso livello.

Le eccezioni permettono al programmatore di trattare condizioni inusuali o inattese senza farcire il codice di infiniti controlli di errore.

Ne risulta un codice piu' facile da leggere e piu' robusto. Il trattamento di condizioni eccezionali puo' essere delegato ad un singolo pezzo di codice.

NOTA: anche le eccezioni sono "classi"!

Marco Ronchetti - ronchet@altavista.it

Exceptions

Scheletro di un codice che gestisce eccezioni

```
class myClass {
  void someMethod (String args[]) throws
TypeOfException {
      ...
      try {
              codiceChePuoGenerareEccezioni();
          }
      catch (KindOfException e) {
              codiceCheTrattaLEccezione(e);
          }
      finally {
              codiceEseguitoInOgniCaso();
          }
      ...
  }
}
```

Marco Ronchetti - ronchet@altavista.it

IO di base


```
import java.io.*;
class SimpleReader {
 public static void main (String args[])
    throws IOException {
  boolean error;
  float f=0;
  int i=0;
  DataInputStream s =
      new DataInputStream(System.in);
  System.out.println("Dammi una stringa");
  String str=s.readLine();
  System.out.println("Hai scritto "+str);
  do {
      System.out.println("Dammi un float");
      try{
      error=false;
        f=Float.valueOf(s.readLine()).floatValue();
      } catch (NumberFormatException e) {
        error=true;
        System.out.println("Input non valido");
      }
  } while (error);
  System.out.println("Hai scritto "+f);
  System.out.println("Dammi un intero");
 i=Integer.valueOf(s.readLine()).intValue();
  System.out.println("Hai scritto "+i);
  System.out.println("Dammi un byte per finire");
  byte b=s.readByte();
  System.out.println("Hai scritto il
carattere"+b);
 }
}
```

Miscellanea

No Enumerated Types

Java does not support the C enum keyword for defining types that consist of one of a specified number of named values. This is somewhat surprising for a strongly-typed language like Java. Enumerated types can be partially simulated with the use of static final constant values.

No typedef

Java does not support the C typedef keyword to define aliases for type names. Java has a much simpler type naming scheme than C does, however, and so there is no need for something like typedef.

Marco Ronchetti - ronchet@altavista.it

Miscellanea

No Method Types

C allows you to store the address of a function in a variable and to pass function addresses to other functions. You cannot do this in Java: methods are not data, and cannot be manipulated by Java programs. Note, however, that objects are data, and that objects can define methods. [9] So, when you need to pass a method to another method, you declare a class that defines the desired method and pass an instance of that class.

No Variable-Length Argument Lists

Java does not allow you to define methods that take a variable number of arguments, as C does. This is because Java is a strongly typed language and there is no way to do appropriate type checking for a method with variable arguments. Method overloading allows you to simulate C "varargs" functions for simple cases, but there is no general replacement for this C feature.

Miscellanea

No Bitfields

Java does not support the C ability to define variables that occupy particular bits within struct and union types. This feature of C is usually only used to interface directly to hardware devices, which is never necessary with Java's platform-independent programming model.

No Variable-Length Argument Lists

Java does not allow you to define methods that take a variable number of arguments, as C does. This is because Java is a strongly typed language and there is no way to do appropriate type checking for a method with variable arguments. Method overloading allows you to simulate C "varargs" functions for simple cases, but there is no general replacement for this C feature.

Marco Ronchetti - ronchet@altavista.it