



Programmazione II

Linguaggi di Programmazione Mod. 1

Marco Ronchetti





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)*



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**.*



Orario

Teoria: martedì, venerdì

Esercitazioni:

Lunedì SOLO quando indicato!

La prossima settimana (27/2): matricole dispari

La settimana successiva (6/3): matricole pari

Esercitatore: Valeria Viannei



Impegno

1 credito = 25 ore di studio

6 crediti = 150 ore. In aula: $12 \times 4 = 48$ ore

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



Esame

Esame articolato in due fasi:

- **Primo scritto (40 min,**
 - 8 esercizi di lettura di codice,
 - 10 domande a risposta multipla,
 - correzione immediata)
- **Prova pratica (sviluppo di codice, 3/4 ore).**

**Sul sito web del corso troverete il materiale degli anni scorsi,
compresi alcuni testi di esame.**



Risorse del corso

Risorse del corso



Sito web

latemar.science.unitn.it/segue/index.php?&login=logout&action=site&site=ronchet

home | directory | tracking


Marco Ronchetti's Web Site > Marco Ronchetti > About me (Home page)

Marco Ronchetti | Software | Corsi per le scuole | Software libero | My courses (Didattica) | Dedicato al futuro

About me (Home page) | How to reach me | Present duties | Past duties | Curriculum Vitae | Publications | Positions | My students | My Web 2.0

ABOUT ME (HOME PAGE)

Marco Ronchetti
马可·罗克迪



Marco Ronchetti
Computer Science Associate Professor

Dipartimento di Ingegneria e Scienza dell'Informazione - Facoltà di Scienze
Università di Trento - Via Sommarive 38 - 38050 Povo di Trento - Italy
Tel.: +39 0461 282033 e-mail: marco.ronchetti at unitn.it

350 visitors

This is my official (i.e. related to my job) Web

Linguaggi di programmazione 1
Laboratorio di programmazione di sistemi
Web Architectures - 2015/16
eSchooling

competenze
didattica per competenze

Hot Stuff

- The LODE system
- Io e Wikipedia
- Carta servizi per studenti con disabilità e BES
- WillDOS
- Software libero

latemar.science.unitn.it

oppure cercare
con google
marco ronchetti

home | directory | tracking

Marco Ronchetti (professor) | logout

Linguaggi di programmazione - Modulo 1 (a.a. 2015/16)

Linguaggi di programmazione 1 > Calendario e materiale didattico > Calendario e materiale didattico

Informazioni generali | Esame | Calendario e materiale didattico [reorder] [move] [edit] [del] [+ add section]

This section has only one page. If you want to hide this sidebar when viewing this section see: [display options](#)

Calendario e materiale didattico
[reorder] | [move] | [edit] | [del] | [+ add item]

1	Mar. 16 Feb. 14:00	Introduzione al corso - Richiami di C++	Slides
E1A	Gio 18 Feb. 16:00 GRUPPO A-L	Laboratorio - Introduzione a Netbeans	
2	Ven 19 Feb 9:00	Richiami di C++	
3	Mar. 23 Feb. 15:00	Dalle struct alle classi	
E1B	Gio 25 Feb 16:00 GRUPPO M-Z	Laboratorio - Introduzione a Netbeans	



Un buon libro...

Gratis in forma elettronica, in inglese:

Thinking in Java

Bruce Eckel

<http://www.mindview.net/Books>

In Italiano:

Thinking in Java

Bruce Eckel

Ed. Apogeo

(in libreria)



Basic tools: JRE, JDK

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



[Sign In/Register](#) [Help](#) [Country](#) [Communities](#) [I am a...](#) [I want to...](#)

[Products](#) [Solutions](#) [Downloads](#) [Store](#) [Support](#) [Training](#) [Partners](#) [About](#) [OTN](#)

[Oracle Technology Network](#) > [Java](#) > [Java SE](#) > [Downloads](#)

[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Technologies](#) [Training](#)

Java SE Downloads

[DOWNLOAD](#)

Java Platform (JDK) 8u73 / 8u74

[DOWNLOAD](#)

NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 8u73 / 8u74
 Java SE 8u73 includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Java SE 8u74 is a patch-set update, including all of 8u73 plus additional features (described in the release notes).
[Learn more](#)

- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
- [Java SE Products](#)
- [Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme Files](#)
 - [JDK ReadMe](#)
 - [JRE ReadMe](#)

JDK
[DOWNLOAD](#)

Server JRE
[DOWNLOAD](#)

JRE
[DOWNLOAD](#)

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources

- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Java.net](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine



“The” Tutorials and examples

<https://docs.oracle.com/javase/tutorial>



The Java™ Tutorials

[Download Ebooks](#)
[Download JDK](#)
[Search Java Tutorials](#)

The Java Tutorials are practical guides for programmers who want to use the Java programming language to create applications. They include hundreds of complete, working examples, and dozens of lessons. Groups of related lessons are organized into “trails”.

The Java Tutorials primarily describe features in Java SE 8. For best results, [download JDK 8](#).

What's New

The Java Tutorials are continuously updated to keep up with changes to the Java Platform and to incorporate feedback from our readers.

This release of the tutorial corresponds to the JDK 8u40 release.

This release includes a new lesson in the Deployment trail that describes how to use the Java packaging tools to generate self-contained applications. Self-contained applications are Java applications that are bundled with the JRE that is needed to run. These applications are installed on a user's local drive and launched in the same way as native applications. See [Deploying Self-Contained Applications](#) for more information.

Trails Covering the Basics

These trails are available in book form as *The Java Tutorial, Sixth Edition*. To buy this book, refer to the box to the right.

- » [Getting Started](#) — An introduction to Java technology and lessons on installing Java development software and using it to create a simple program.
- » [Learning the Java Language](#) — Lessons describing the essential concepts and features of the Java Programming Language.
- » [Essential Java Classes](#) — Lessons on exceptions, basic input/output, concurrency, regular expressions, and the platform environment.
- » [Collections](#) — Lessons on using and extending the Java Collections Framework.
- » [Date-Time APIs](#) — How to use the `java.time` packages to write date and time code.
- » [Deployment](#) — How to package applications and applets using JAR files, and deploy them using Java Web Start and Java Plug-in.
- » [Preparation for Java Programming Language Certification](#) — List of available training and tutorial resources.



Not sure where to start?
See [Learning Paths](#)

Tutorial Contents

[Really Big Index](#)

Tutorial Resources

- » View the [Java Tutorials Online](#) (Last Updated 3/3/2015).
- » The [Java Tutorials' Blog](#) has news and updates about the Java SE tutorials.
- » [Download the latest Java Tutorials bundle](#).



Java

JAVA:

hands on



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  
    }  
}
```



Sintassi del linguaggio

Dichiarazioni: come in C

Strutture di controllo (for, while, if, switch...) :
come in C

I/O:

```
System.out.println("..." + var + "...");
```




Piccolo esempio

```
class Applicazione{  
public static void main (String args[]) {  
    Applicazione p= new Applicazione();  
}  
Applicazione() {  
    int i;  
    for (i=1;i<10;i++)  
        if (i%2== 0) System.out.println(i);  
    System.out.println("finito!");  
}  
}
```



Uso di JDK

Compilazione:

\$javac HelloWorld.java

produce HelloWorld.class

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

Obbligatorio
specificare
l'estensione!

Esecuzione...

\$java HelloWorld

(la classe indicata deve contenere il main)

Obbligatorio
omettere
l'estensione!



Differenze sintattiche tra Java e C++

?(Java == ((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...)



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



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.**



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:

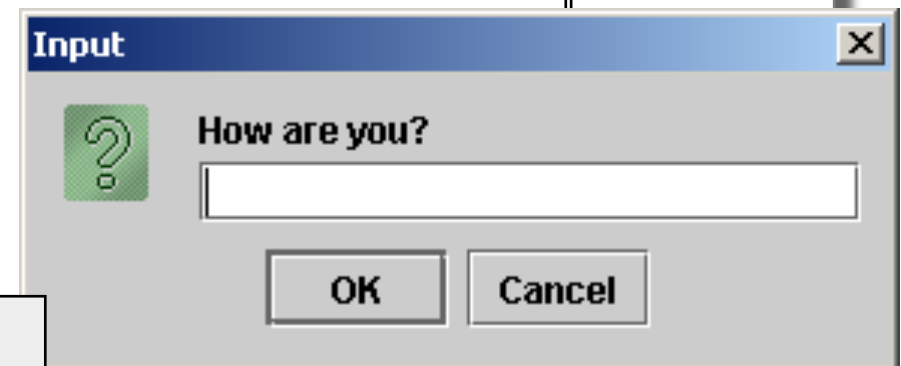
```
char data[] = {'a', 'b', 'c'};  
String str = new String(data);
```




Lettura di stringhe con GUI

```
import javax.swing.JOptionPane;  
class Applicazione {  
  
...  
  
String input =  
    JOptionPane.showInputDialog(  
        "How are you?");  
    System.out.println(input);  
  
...  
  
    System.exit(1);  
  
...  
}
```

Essenziale!
Altrimenti la thread che gestisce la GUI
rimane viva, e il processo non termina





Lettura di un intero

```
int type;  
String s;  
...  
do {  
    try {  
        type =Integer.parseInt(  
            JOptionPane.showInputDialog(  
                "Pila (1) o Coda (2)?"));  
    } catch (Exception e) {type=0;}  
} while (type<1 || type>2);  
switch (type) {  
    case 2: s="Pari"; break;  
    case 1: s="Dispari"; break;  
}  
System.out.println(s);  
...  
System.exit(1);
```



Altre differenze

- Gestione degli arrays `i[10]`
- `import`
- argomento del test booleano: `if(...)`
- parametri di ingresso



Java

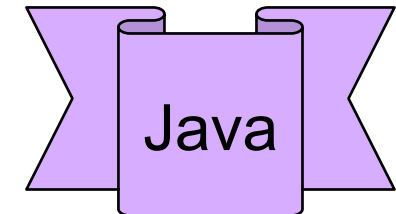
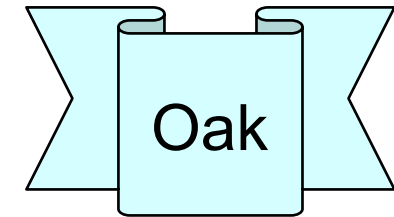
JAVA:

storia e modello



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!**



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**



Prestazioni...

Inferiori al C++...

**Tempo di sviluppo:
Inferiore al C++ ...**



Java - Introduction

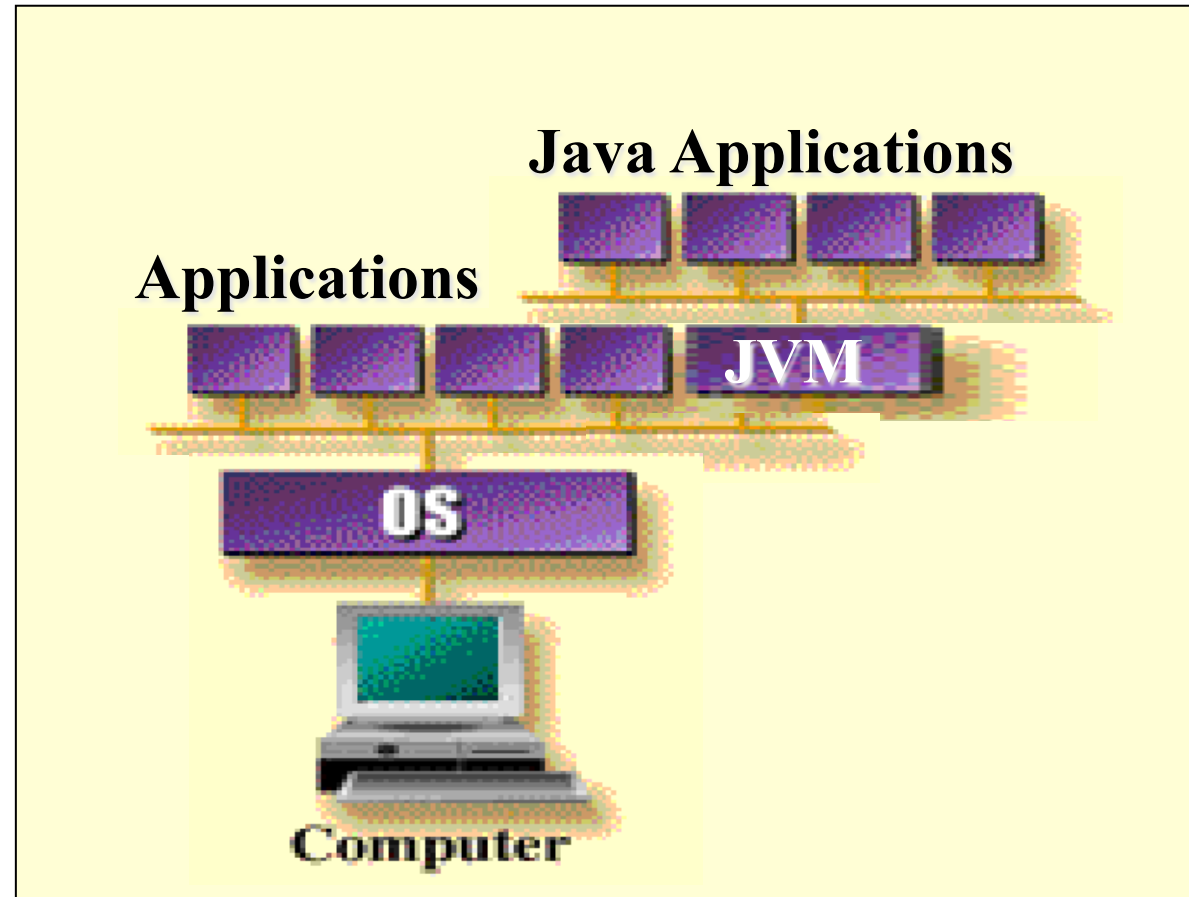
Applications are built
in the frame of the
OPERATING SYSTEM
Which in turn is built over a
particular
HARDWARE





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



Applicazioni

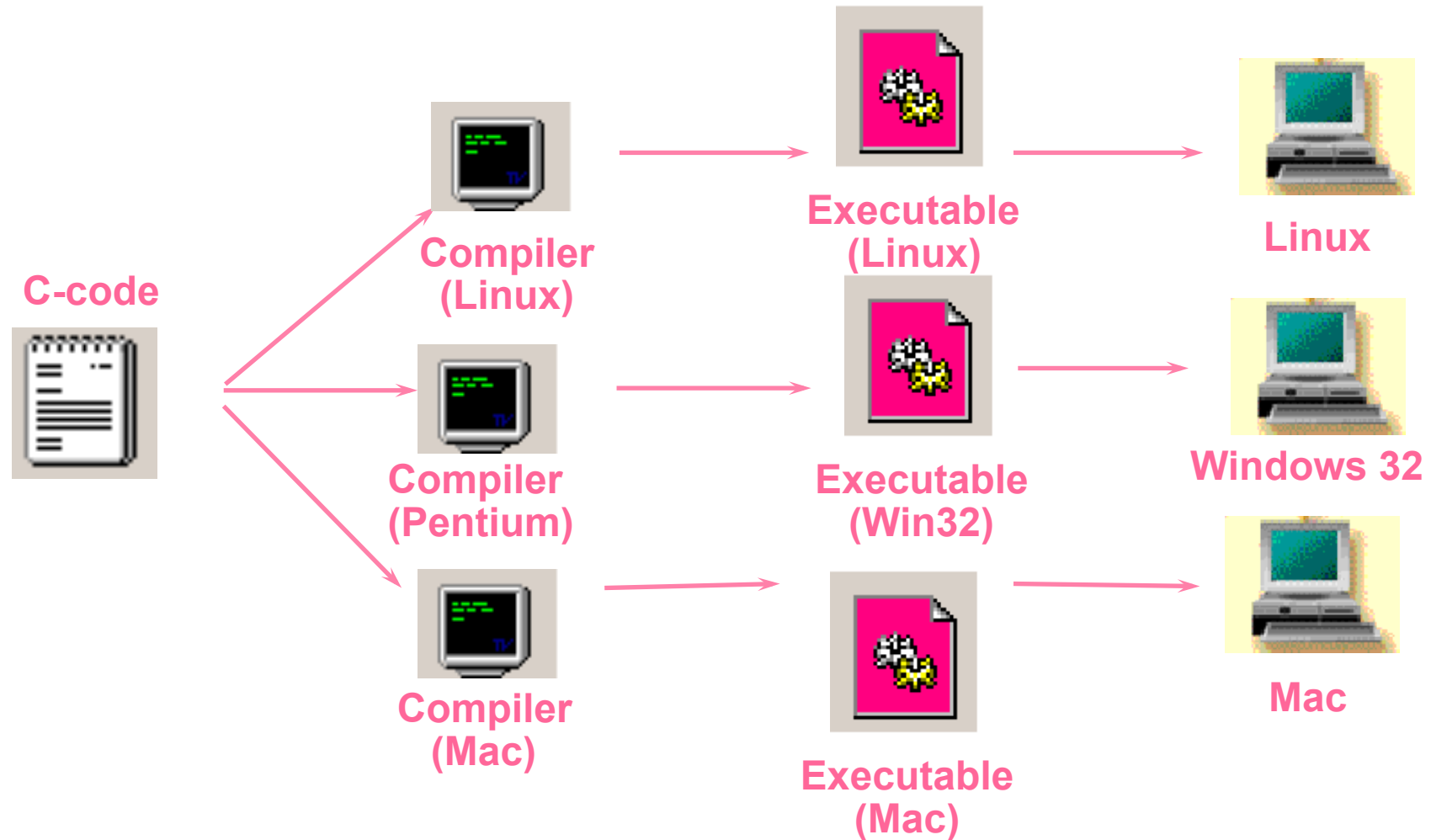
Definizione:

Programmi stand-alone scritti in linguaggio Java.

Possono essere eseguiti da una Java Virtual Machine:

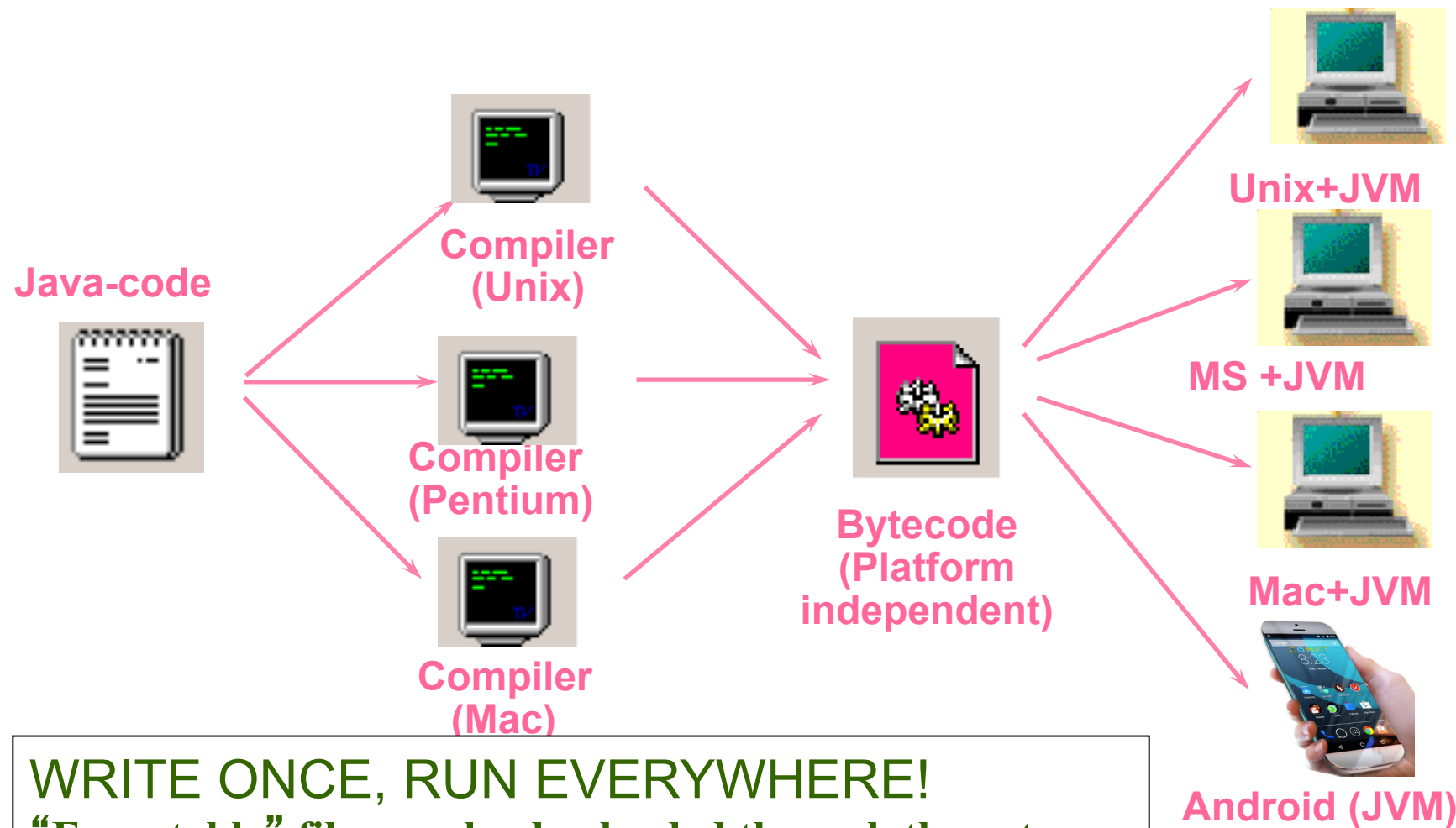
- Fisica: un processore il cui assembler e' il bytecode
- Virtuale: un interprete o Just In Time Compiler Java.

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



“the first universal software platform”

Consists of:

The language *Easy!*

The Virtual Machine

(Many) class libraries and API

You don't care!

*That's the
difficult part!*

Java: the platform for “Internet Computing”

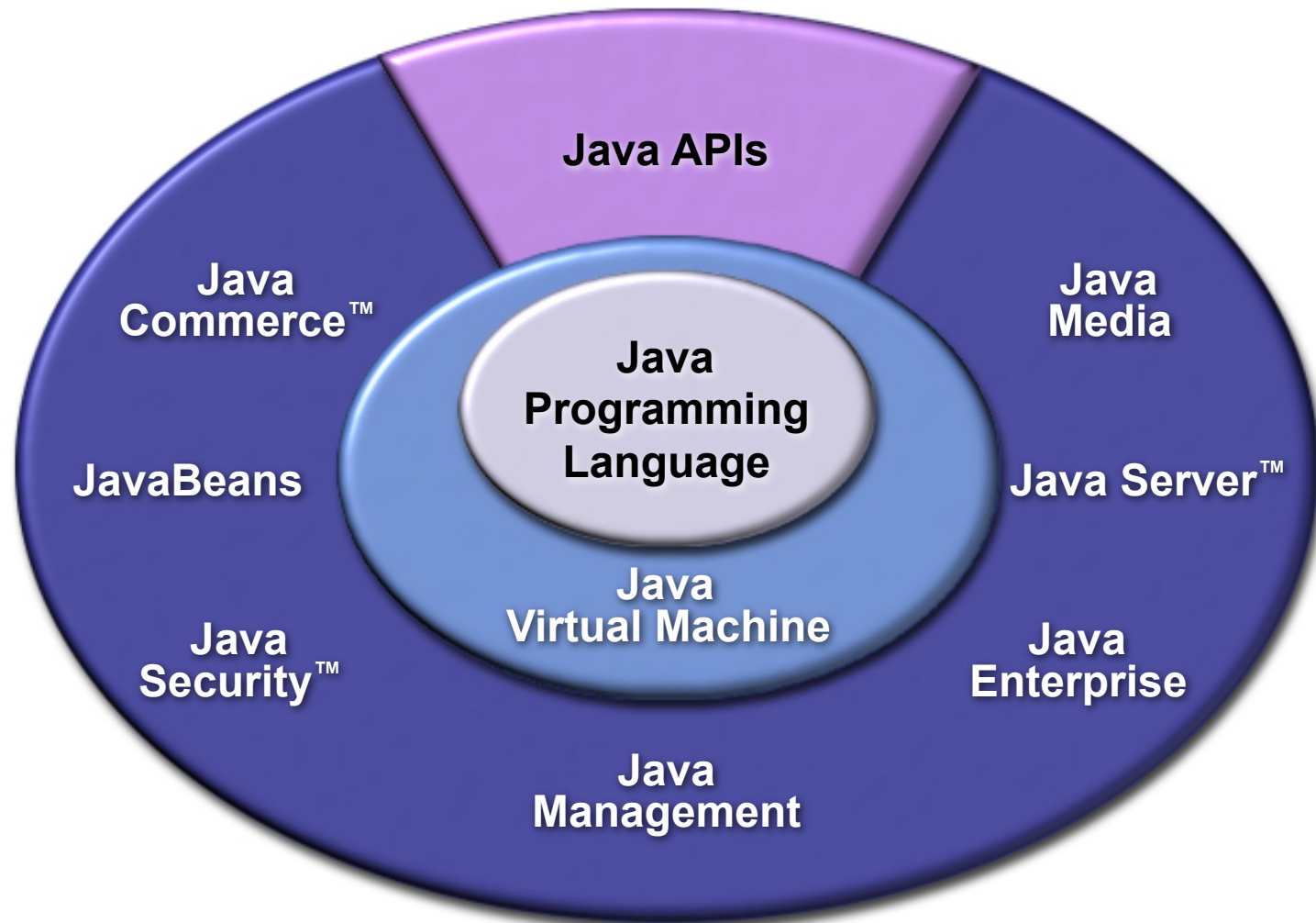
Hardware independent

• Scalable

• Open



The Java Platform





In salsa microsoft...

Visual-J

C#

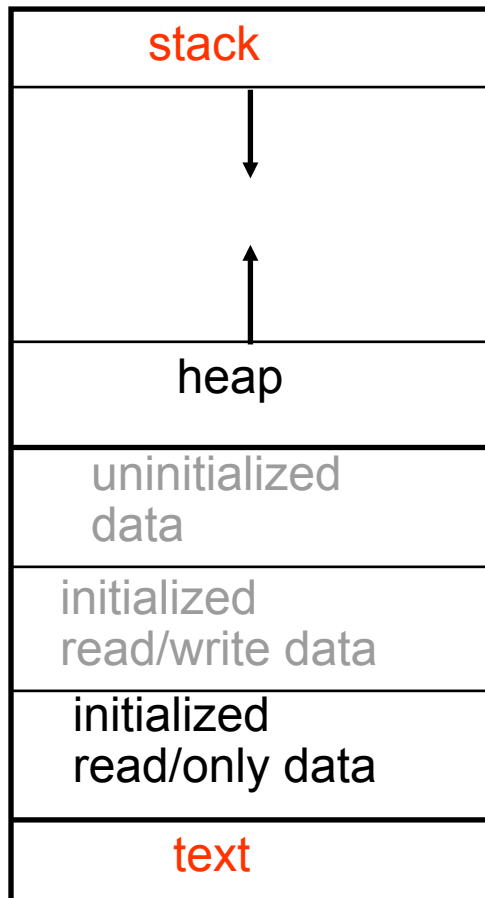


Richiami di C++ di base

Richiami di C++ di base Parte 1



Il modello di memoria



*memoria allocata dalle funzioni
(Variabili automatiche)*

*memoria allocata dinamicamente
dal programmatore*

Variabili globali e statiche

<- questo é supportato solo da alcuni hardware

Codice eseguibile



Modularizzazione: Funzioni

Funzioni come "procedure parametrizzate"

```
tipo funzione(tipo argom1, ..., tipo argomN) {  
    corpo della funzione  
    return var;  
}
```



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) {
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " =
" << res << "\n";
}
```

stack

a	2
b	3
res	?

0
4
8
12
16
20
24
28
32
36
40
44
...

main

heap



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}

int prodotto(int b, int a) {
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}

main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " = "
    << res << "\n";
}
```

stack

a	2	0
b	3	4
res	?	8
a	3	12
b	2	16
res	0	20
k	0	24
		28
		32
		36
		40
		44
		...
heap		

main

prodotto



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) { stack
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " =
    " << res << "\n";
}
```

a	2	0	main
b	3	4	
res	?	8	
a	3	12	prodotto
b	2	16	
res	0	20	
k	0	24	
a	0	28	somma
b	3	32	
res	?	36	
		40	
		44	
heap		...	



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) { stack
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " =
" << res << "\n";
}
```

a	2	0	main
b	3	4	
res	?	8	
a	3	12	prodotto
b	2	16	
res	0	20	
k	0	24	somma
a	0	28	
b	3	32	
res	3	36	
		40	
		44	
heap		...	



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) {
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " = "
    << res << "\n";
}
```

stack

a	2	0	main
b	3	4	
res	?	8	
a	3	12	prodotto
b	2	16	
res	3	20	
k	1	24	
a	0	28	
b	3	32	
res	3	36	
		40	
		44	
heap		...	



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) { stack
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " =
" << res << "\n";
}
```

a	2	0	main
b	3	4	
res	?	8	
a	3	12	prodotto
b	2	16	
res	3	20	
k	1	24	somma
a	3	28	
b	3	32	
res	6	36	heap
		40	
		44	
heap		...	



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) {
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " = "
    << res << "\n";
}
```

stack

a	2	0
b	3	4
res	?	8
a	3	12
b	2	16
res	6	20
k	1	24
a	3	28
b	3	32
res	6	36
		40
		44
		...
heap		

main

prodotto



Modularizzazione: Funzioni

Esempio

```
int somma(int a, int b) {
    int res;
    res=a+b;
    return res;
}
int prodotto(int b, int a) {
    int res=0;
    for (int k=0; k<b; k++)
        res=somma(res,a);
    return res;
}
main() {
    int a,b,res;
    cout << "dammi due numeri \n";
    cin >> a >> b;
    res=prodotto(a,b);
    cout << a << " * " << b << " = "
    << res << "\n";
}
```

stack

a	2	0
b	3	4
res	6	8
a	3	12
b	2	16
res	6	20
k	1	24
a	3	28
b	3	32
res	6	36
		40
		44
		...
heap		

main



Funzioni ricorsive

Una funzione può richiamare se stessa.

```
int fact(int n) {  
    if (n==0) return 1;  
    else return n*fact(n-1);  
}  
main(void) {  
    int n;  
    cout<<"dammi un numero\n";  
    cin >> n;  
    cout << "Il suo fattoriale vale "<<fact(n)<<"\n";  
}
```

Cosa avviene nello stack?