UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Programmazione 3
e
Laboratorio di Programmazione 3

Introduzione
alla
programmazione OO

Angelo Ciaramella

Linguaggi di Programmazione

- I linguaggi di programmazione ad alto livello sono indipendenti dall'architettura del computer

- Compilatore legge le istruzioni in un linguaggio di programmazione
  - le tradurrà in codice macchina solo se si attengono fedelmente alle convenzioni del linguaggio

Paradigmi di programmazione

- In parallelo con i linguaggi di programmazione si sono sviluppati vari paradigmi di programmazione
  - Programmazione imperativa
  - Programmazione dichiarativa
    - Funzionale, logica, algebrica
  - Programmazione orientata ad oggetti
    - Object Oriented (OO) Programming

Paradigmi di programmazione

- Un linguaggio fornisce costrutti e meccanismi per programmare in uno o più paradigmi

Pascal, Fortran, Cobol, C	Imperativo
ML, Lisp, Haskell, Scheme	Dichiarativo (funzionale)
Prolog	Dichiarativo (logico)
Smalltalk	OO
C++, Java, Delphi	Imperativo + OO
Clos, Zeta-Lisp	Funzionale + OO

Paradigmi di programmazione

- Programmazione imperativa
  - Enfasi posta nella progettazione e nella scrittura del procedimento con il quale, a partire dai dati in ingresso, si producono i risultati
- Programmazione OO
  - Enfasi posta nella progettazione di un modello del dominio di applicazione
  - Un oggetto è una rappresentazione di un oggetto del mondo reale o di un concetto
  - Una computazione si sviluppa come una sequenza di attivazioni di funzionalità di oggetti

Programmazione OO in Java

oggetti	**instanze** di classe (*instances*)
classi	**classi** (*classes*)
struttura	**variabili di istanza** (*instance variables*)
funzionalità	**metodi** (*methods*)
attivazione di funzionalità	**chiamata di metodo** (*method call*)
creazione di oggetti	primitiva **new**
ereditarietà	relazione **extends**

Caratteristiche generali di Java

Breve Storia di Java

- Java fu originariamente progettato per programmare apparecchi di consumo
- Il suo primo successo fu nella scrittura di Applet per internet
- Nel 1995 il Browser HotJava, poteva scaricare programmi, chiamati Applet, dal Web ed eseguirli
- Dal 1996, molti browser supportano Java
- Oltre al linguaggio di programmazione, Java ha una ricca libreria

Caratteristiche di Java

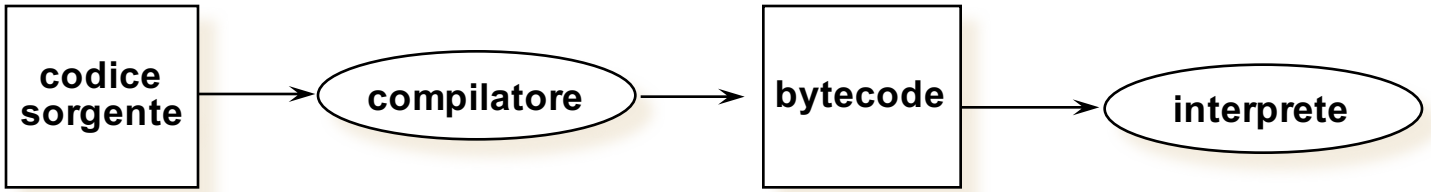
- Caratteristiche principali
  - Semplice e OO
  - Interpretato
  - Architetturalmente neutro e portabile
  - Robusto
  - Distribuito
  - Sicuro
  - Dinamico
  - Ad elevati prestazioni
  - Concorrente (multi-thread)

Semplice e OO

- Sintassi simile a C e C++
- Elimina i costrutti più "pericolosi" di C e C++
  - aritmetica dei puntatori
  - (de)allocazione esplicita della memoria
  - strutture (struct)
  - definizioni di tipi (typedef)
  - preprocessore (#define)
- Aggiunge *garbage collection* automatica
- Conserva la tecnologia OO di base di C++
- Rivisita C++ in alcuni aspetti

Interpretato

- Il compilatore produce un codice intermedio per una "Java Virtual Machine" (bytecode)
  - Successivamente interpretato



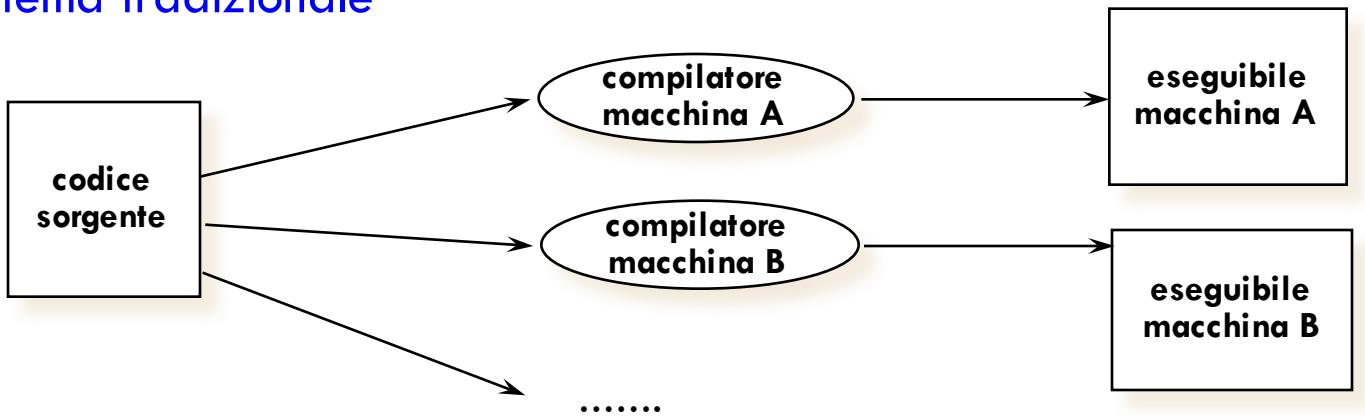
Generazione di bytecode

Architetturalmente indipendente

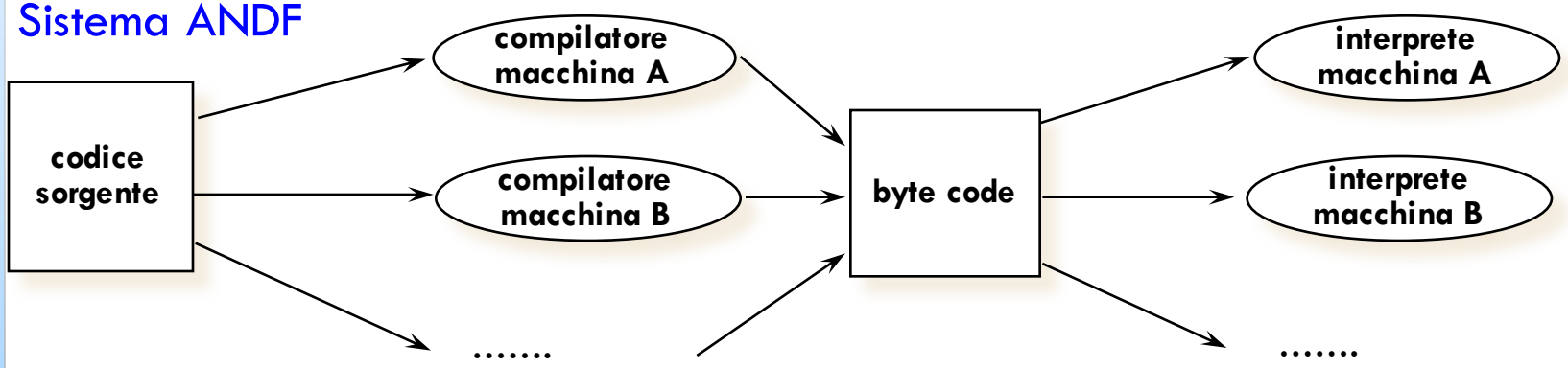
- Il *bytecode* è indipendente dall'architettura hardware
  - ANDF - Architecture Neutral Distribution Format
- Pertanto, un programma bytecode può essere eseguito su qualsiasi sistema su cui giri un ambiente run-time Java

Architetturalmente indipendente

Sistema tradizionale

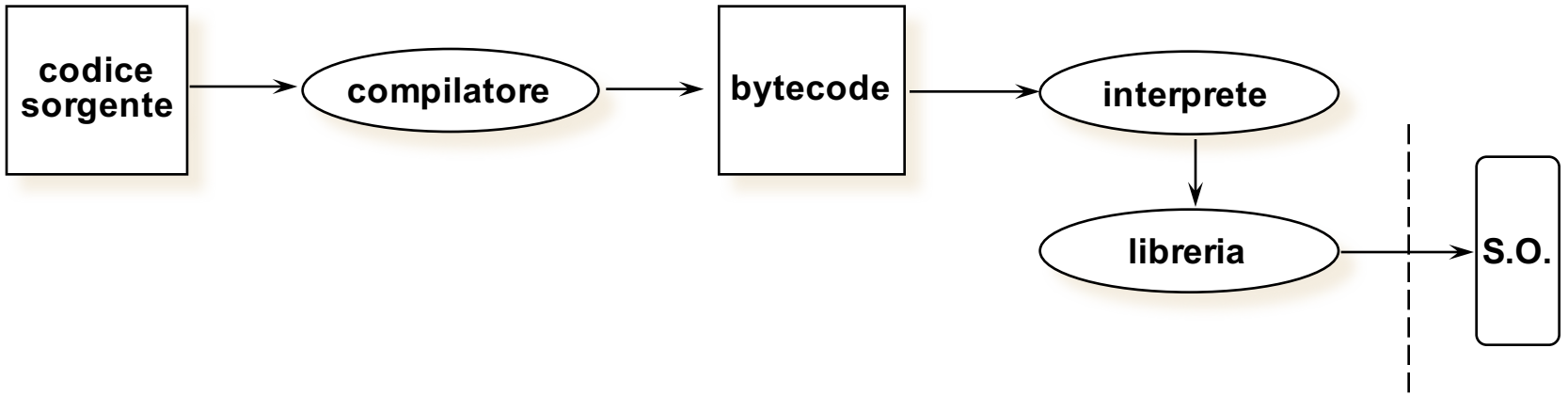


Sistema ANDF



Portabile

- Il sistema Java (compilatore + interprete + librerie run-time) è facilmente portabile su piattaforme diverse
  - il compilatore Java è scritto in Java
  - l'ambiente run-time è scritto in ANSI C con interfacce standard (POSIX) verso il sistema operativo
  - nessuna "*implementation dependency*"



Robusto

- Controlli estensivi
  - a compile-time e a run-time, per rilevare gli errori quanto prima possibile (es.: type checking)
- Le caratteristiche insicure di C e C++ sono rimosse
  - Nessuna gestione esplicita dei puntatori (no aritmetica dei puntatori, no malloc e free esplicite, …)
  - Gestione della memoria con *garbage collection*
  - Array e stringhe "veri"
- Verifica del bytecode a *load-time*

Distribuito

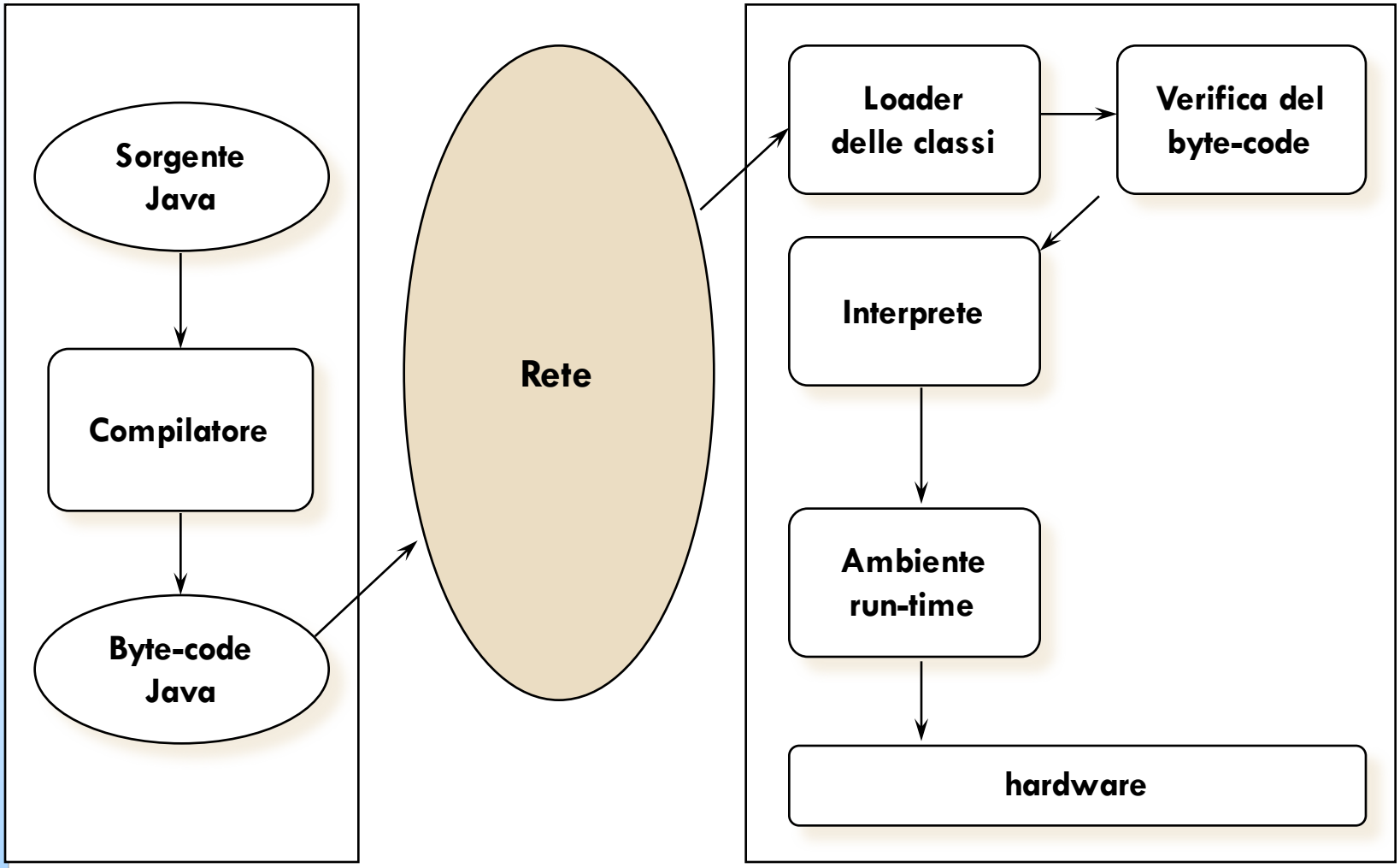
- Pensato per essere eseguito in rete
- L'ambiente run-time incorpora funzioni di rete
  - basso livello
    - TCP/IP
  - alto livello
    - HTTP, …
- La rete è facilmente accessibile (come i file locali)

Sicuro

- L'ambiente di esecuzione si protegge da bytecode potenzialmente "ostile"

- Esempi
  - il bytecode viene verificato prima dell'interpretazione ("*theorem prover*"), in modo da essere certi di alcune sue caratteristiche
  - gli indirizzamenti alla memoria nel bytecode sono risolti sotto il controllo dell'interprete

Sicuro

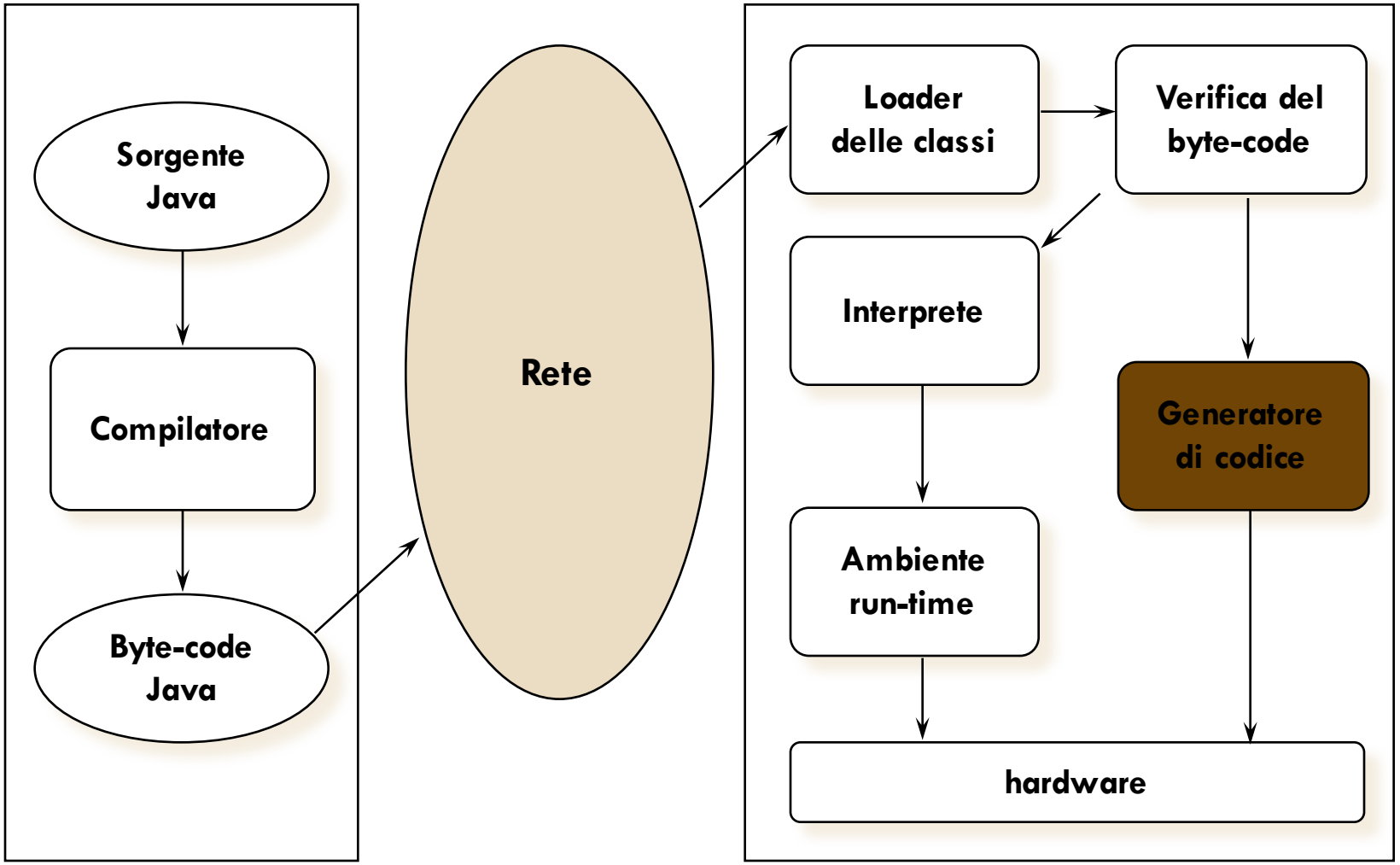

Sorgente Java
Compilatore
Byte-code Java
Rete
Loader delle classi
Verifica del byte-code
Interprete
Ambiente run-time
hardware

Elevate prestazioni

- La verifica del bytecode permette di saltare molti controlli a run-time
  - l'interprete è pertanto efficiente

- Per maggiore efficienza, c'è possibilità di compilazione on-the-fly del bytecode in codice macchina

Elevate prestazioni



Dinamico

- Il codice è eseguibile anche in assenza di alcuni moduli
  - … le classi necessarie per la esecuzione di un programma Java possono essere caricate e collegate dinamicamente quando servono

- Esempio
  - nuove release di moduli caricabili automaticamente dalla rete quando servono

Concorrente

- Multi-threading parte integrante del linguaggio
  - Applicazioni interattive più facili a scriversi
  - Migliore "reattività" (anche se non real-time)
- Consente ad applicazioni Java di sfruttare il meccanismo di concorrenza logica
- Parti separate di un programma possono essere eseguite come se fossero (dal punto di vista del programmatore) processate parallelamente
- L'uso di *thread* rappresenta un modo semplice di gestire la concorrenza tra processi

Ricco

- La Standard Library Java contiene una ricca collezione di classi e di metodi preconfezionati
  - Language support
  - Utilities
  - Input/output
  - Networking
  - JavaFX, Swing e Abstract Window Toolkit (AWT)
  - …

Tecnologia e piattaforma

- L'interprete Java esiste per molti Sistemi Operativi (SO)
  - Windows, Linux, Solaris (il suo sistema operativo di origine), Mac, ecc.

- Esistono numerosissime librerie scritte in Java che completano la già ampia libreria standard
  - Java è uno dei linguaggi più usati attualmente, specialmente per scrivere applicazioni Web

Compilatori

- Alcuni compilatori possono essere eseguiti dalla linea di comando
  - Usando il Java Development Kit (JDK)
- Altri dispongono di un ambiente di sviluppo integrato
  - IDE - Integrate Development Environment
    - NetBeans, Eclipse, XCode, …

Linea di comando

- Per scrivere un programma Java, abbiamo bisogno di
  - Semplice editor di testo
    - Editor Java open source EJE (http://sourceforge.net/projects/eje/)
    - TextMate
  - Il JDK versione Standard Edition
    - http://www.oracle.com/technetwork/java/index.html

Linea di comando

- Tramite un editor (e.g., *gedit*) di testo scriviamo il file `file.java`
- Compiliamo
  - `javac file.java`
- Se è andato a buon fine otteniamo
  - `file.class`
- Eseguiamo con il comando
  - `java file (.class)`

Directory del compilatore

- Dopo l'istallazione abbiamo le seguenti directory
  - Programmi per il JDK (compilatore, interprete, debugger, ..)
    - `\java\bin`
  - Classi standard per Java
    - `\java\lib`
- File Header per programmi che considerano C/C++ e Java
  - `\java\include`
- Programmi dimostrativi
  - `\java\demo`

javac

- Il compilatore converte un file `filename.java` in un bytecode
- Dopo la compilazione il bytecode è memorizzato nel file `filename.class`
- Sintassi

```
javac [options] filename.java
```


javac

- options
  - -nowite
    - compilazione senza creare la classe
  - -nowarm
    - compilazione disattivando i warning
  - -verbose
    - compilazione con visualizzazione delle informazioni delle sorgenti
  - -d dir
    - la directory dir è la prima directory per i package
  - -*debug*
    - il compilatore viene eseguito in modalità debug
  - -g
    - prepara il bytecode per il debug

java

- L'interprete è usato per eseguire l'applicazione compilata

- Sintassi

```
java [options] filename
```

  - `filename`
    - nome del file (classe)
  - `options`
    - digitare java dalla linea di comando

javap

- Il diassembler è usato per diassemblare il bytecode compilato

- Sintassi

```
javap [options] filenames
```

  - `filenames`
    - nome dei file (classi)
  - `options`
    - digitare java dalla linea di comando

javah

- Crea heder file C per estendere il codice Java per il linguaggio C

- Sintassi

```
javah [options] filename
```

  - `filename`
    - nome del file (classe)
  - `options`
    - digitare javah dalla linea di comando

javadoc

- Crea dei file HTML per gestire la documentazione delle classi e/o metodi

- Sintassi

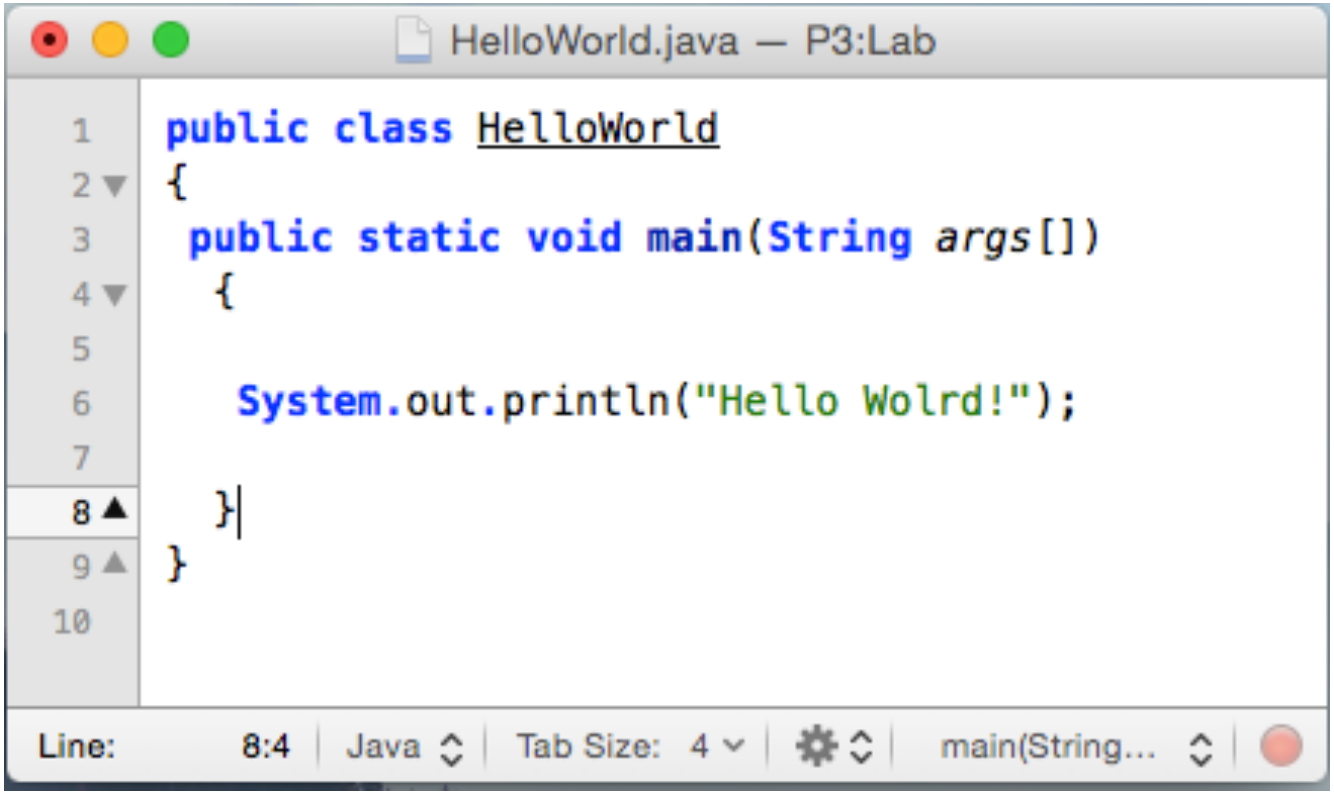
```
javadoc filename.class
```

  - `filename.class`
    - File della classe da processare

javad e appletviewer

- javad
  - Tool di debugging
  - Usato su file locali e remoti
- Appletviewer
  - Permette di testare gli applet java

"Hello World"


HelloWorld.java — P3:Lab
1 public class HelloWorld
2 {
3 public static void main(String args[])
4 {
5
6 System.out.println("Hello Wolrd!");
7
8 }
9 }
10
Line: 8:4
Java
Tab Size: 4
main(String...