

STRUMENTI DI BASE DEL JDK

PROGRAMMAZIONE AD OGGETTI

C.D.L. INGEGNERIA E SCIENZE INFORMATICHE

Danilo Pianini — danilo.pianini@unibo.it

Slide compilate il: 2025-09-28

 [versione stampabile](#)

 [menu principale](#)

Tipi di piattaforme Java

Domini applicativi

Esistono diverse piattaforme Java per diversi **domini applicativi**. Fra diverse piattaforme cambia il set di librerie disponibili e l'ambiente runtime (JVM).

- **Java ME** (Java Platform, Micro Edition) – per dispositivi resource-constrained
- **Java SE** (Java Platform, Standard Edition) – per applicazioni general-purpose su computer e server
- **Jakarta EE** (ex Java Platform, Enterprise Edition) – Java SE + librerie estese per applicazioni distribuite
- **Java Card** – per applicazioni su smart card e dispositivi con memoria limitata

Sviluppo o semplice esecuzione

Una distribuzione Java può o meno includere gli strumenti per lo sviluppo (SDK). Nel primo caso si parla di **Java Development Kit** (JDK), nel secondo di **Java Runtime Environment** (JRE).

Implementazioni della JVM

Data una piattaforma, possono esserci più **implementazioni** ad esempio, per Java SE, con caratteristiche diverse (ad esempio, prestazioni e utilizzo della memoria): OpenJDK (HotSpot), Eclipse OpenJ9, GraalVM, Codename One

Distribuzioni

Data una implementazione, possono esserci più **distribuzioni** (binari pre-built) della JVM che possono contenere anche delle modifiche minori fra loro. Ad esempio, per OpenJDK: Temurin (Adoptium), DragonWell (Alibaba), Corretto (Amazon), Zulu (Azul), Liberica (BellSoft), Oracle, RedHat, Microsoft, JetBrains...

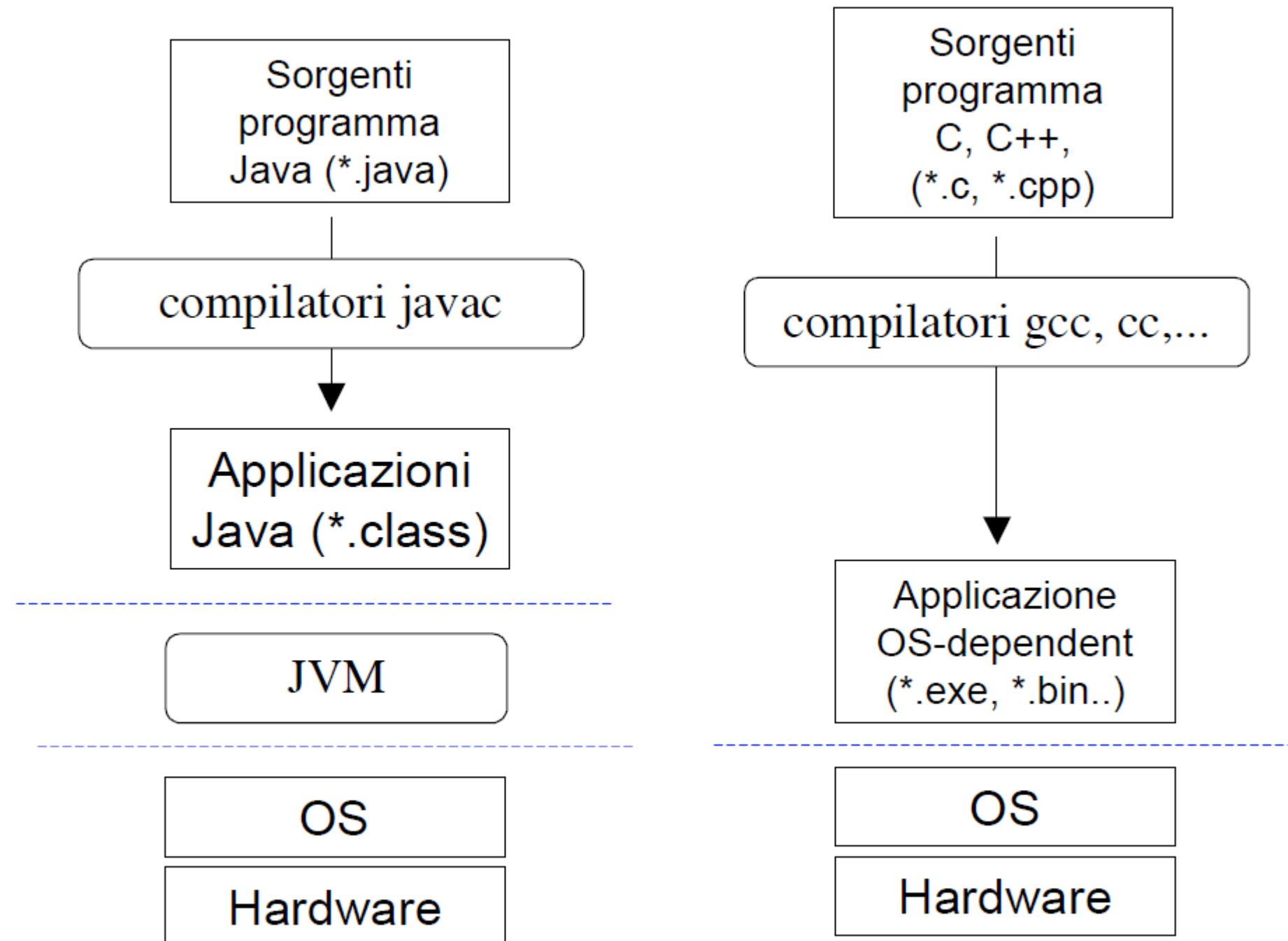
Ambiente di riferimento per l'A.A. corrente

Utilizzeremo Java SE Development Kit 21 (JDK 21)

Include il necessario per eseguire applicazioni Java, ossia il JRE con virtual machine (**java**) e relative librerie, più gli strumenti di sviluppo, fra cui:

- compilatore (**javac**),
- documentatore (**javadoc**),
- impacchettatore (**jar**),
- e disassemblatore (**javap**).
- Java SE 21 – penultima versione Long-Term Support (LTS) a disposizione prima dell’inizio del corso
 - ▶ Java 25, ultimissima LTS, è uscita il 16 settembre 2025
 - ▶ Chi lo desidera può sperimentare con le versioni più recenti
 - ▶ Nota: la compatibilità è solo “all’indietro” (**backwards compatibility**): nuove JVM possono in generale eseguire applicazioni compilate per un vecchio bytecode, ma non viceversa
- Faremo riferimento a OpenJDK standard (che fa da **reference**)
 - ▶ Ossia, per capire se una JVM si comporta correttamente, si confronta il suo comportamento con OpenJDK
- Come distributore di riferimento, raccomandiamo Adoptium (<https://adoptium.net/>)

Java Virtual Machine: Architettura a Runtime



IL FILE SYSTEM E L'INTERPRETE DEI COMANDI (RICHIAMO)

Elementi base del file system

I sistemi operativi (S.O.) odierni consentono di memorizzare permanentemente le informazioni su supporti di memorizzazione di massa (dischi magnetici, dispositivi a stato solido...)



Il **file system** è l'insieme di metodi e strutture dati impiegati dal S.O. per controllare l'accesso e la memorizzazione dei dati.

- **Livello logico**: riguarda come l'informazione è presentata all'utente
- **Livello fisico**: riguarda come il livello logico è mappato nelle memorie

Le informazioni sono organizzate in file e cartelle (livello logico):

- i **file** contengono le informazioni
- le **cartelle** sono contenitori, all'interno contengono i file ed altre cartelle

La cartella più esterna (contenente tutte le altre) è la **radice** (*root*) che rappresenta il livello gerarchico più alto del file system

-   Sistemi *nix (Linux, MacOS, BSD, Solaris...): vi è una unica radice, ossia /
-  Sistemi Windows: c'è una root per FS, indicata da una "lettera di unità" (ad esempio: **C:**, **D:**)

File e cartelle possono essere individuati da un percorso o **path**

Il percorso può essere *assoluto*, ossia partire dalla root:

- **/home/user/frameworkFS.jar** (percorso Unix assoluto)
- **C:\Windows\System\win64.dll** (percorso Windows assoluto)

Oppure relativo *relativo* ad un certo punto del filesystem:

- si identifica con **.** la cartella corrente e con **..** la cartella di livello superiore
- **./src/main/java/HelloWorld.java** (percorso Unix relativo, equivalente a **src/main/java/HelloWorld.java**)
- **../Downloads/myapp.jar** (percorso Unix relativo, che risale, come primo passo, alla cartella superiore)

Manipolare il file system

L'utente può osservare e manipolare il file system:

- sapere quali file e cartelle contiene una cartella
- creare nuovi file e cartelle
- copiare file e cartelle
- eliminare file e cartelle
- spostare file e cartelle dentro altre cartelle (equivalente a copiare e rinominare)
- rinominare file e cartelle (equivalente a spostare dentro la stessa cartella)

Il software grafico che consente di osservare e manipolare il file system prende il nome di *file manager*.

-  “Esplora risorse” (**explorer.exe**)
-  “Finder”
-  ne esistono diversi (Nautilus, Dolphin, Thunar, Astro...)

ATTENZIONE: alcuni file manager (specialmente **explorer.exe**) sono preimpostati per nascondere informazioni utili e possono trarre in inganno!

Soluzione: meglio fidarsi dell'interprete comandi...

Interprete Comandi

Programma che permette di interagire con il S.O. mediante comandi impartiti in modalità testuale


- Nell'antichità (in termini informatici) le interfacce grafiche erano sostanzialmente inesistenti, e l'interazione con i calcolatori avveniva di norma tramite interfaccia testuale
- Tutt'oggi, le interfacce testuali sono utilizzate:
 - ▶ per **automatizzare** le operazioni
 - ▶ per **velocizzare** le operazioni (scrivere un comando è spesso molto più veloce di andare a fare click col mouse in giro per lo schermo)
 - ▶ per fare operazioni complesse con pochi semplici comandi
 - ▶ non tutti i software sono dotati di interfaccia grafica
 - ▶ alcune opzioni di configurazione del sistema operativo restano accessibili solo via linea di comando
 - ▶ (anche su Windows: ad esempio i comandi per associare le estensioni ad un eseguibile)

Lo vedrete in maniera esaustiva nel corso di Sistemi Operativi...

Sistemi *nix (Linux, MacOS X, FreeBSD, Minix...)

Nei sistemi UNIX-like esistono vari tipi di interpreti, chiamati shell

Alcuni esempi

- Bourne shell (sh)
 - ▶ Prima shell sviluppata per UNIX (1977)
- C-Shell (csh)
 - ▶ Sviluppata da Bill Joy per BSD
- Bourne Again Shell (bash)
 - ▶ Parte del progetto GNU, è un super set di Bourne shell
- ZSH, Fish, e altri terminali di ultima generazione
 - ▶ Altamente personalizzabili
 - ▶ Molto flessibili
 - ▶ Autocompletamento avanzato e contestualità
 - ▶ ZSH è default su , seppur con configurazione minimale
 - ▶ La shell che vedrete sul sistema del docente è ZSH con configurazione di default di Manjaro Linux
 - ▶ Simili configurazioni possono essere ottenute applicando [oh-my-zsh](#) e [powerlevel10k](#)

Panoramica delle differenze: <http://www.faqs.org/faqs/unix-faq/shell/shell-differences/>

Sistemi Windows

L'interprete comandi storico è rappresentato dal programma `cmd.exe` (`C:\Windows\System32\cmd.exe`)

- Eredita sintassi e funzionalità della maggior parte dei comandi del vecchio MSDOS

Versioni recenti hanno introdotto **PowerShell**, basato su .NET e C#

Da Windows 10 è possibile installare Linux dentro Windows usando **Windows Subsystem for Linux (WSL2)**

- Può essere un modo ragionevole di avere shell Unix in ambiente Windows
 - ▶ ...le macchine di laboratorio hanno una WSL2 con zsh ♥

Più avanti vedremo che lo strumento che useremo per fare controllo di versione installa un proprio emulatore bash.

Aprire un terminale in laboratorio

Terminale DOS/cmd

- Start ⇒ Programmi ⇒ Accessori ⇒ Prompt dei comandi
- Oppure: Start ⇒ Nella barra di ricerca, digitare **cmd** ⇒ cliccare su **cmd.exe**

Terminale Bash emulato (git bash)

- Icona sul desktop
- Oppure: Start ⇒ Nella barra di ricerca, digitare **git bash**
- Consente di lavorare su Windows con un terminale meglio equipaggiato, vicino ad un nativo Linux

Manjaro con Terminale ZSH via WSL2 (con configurazione simile a quella di Pianini)

- Tasto destro sul file “Lancia WSL” presente sul Desktop → *Esegui con PowerShell*
 - ▶ Attendere l’importazione della distro Linux (massimo 2 minuti)
 - ▶ Non aprire altre istanze finché la prima non è avviata!
 - ▶ Se dopo due minuti non è avviata, chiudere e ritentare
- Questa è una shell nativa Linux preconfigurata dal docente

File system e terminale: cheat sheet


Operazione	Comando Unix	Comando Win
Ottenere aiuto / elenco comandi	help	help
Visualizzare la directory corrente	pwd	echo %cd%
Contenuto della directory corrente	ls -alh	dir
Passare alla directory di nome baz	cd baz	cd baz
Passare alla directory superiore	cd ..	cd..
Cambiare unità disco (passare a D:)	-	D:
Eliminare il file foo (non va con le cartelle!)	rm foo	del foo
Eliminare la directory di nome bar	rm -r bar	rd bar
Creare una directory di nome baz	mkdir baz	md baz
Copiare il file foo in bar	cp foo bar	copy foo bar
Spostare (rinominare) il file foo in bar	mv foo bar	move foo bar
Creare un file vuoto di nome foo	touch foo	echo. > foo
Visualizzare il contenuto del file foo	cat foo	type foo
Interrompere il processo corrente (segnale SIGINT)	CTRL + C	CTRL + C

Uso intelligente del terminale

I terminali moderni possono essere utilizzati in modo piuttosto efficace (in particolare se moderni e opportunamente personalizzati).

Autocompletamento

Quasi tutti i terminali offrono la possibilità di effettuare autocompletamento, ossia chiedere al sistema di provare a completare un comando.

- Per farlo si utilizza il tasto  Tab.
- L'autocompletamento cambia (anche molto) fra terminali diversi ed è soggetto a personalizzazione (specialmente su terminali moderni)

Memoria dei comandi precedenti

Sia Bash/zsh che cmd offrono la possibilità di richiamare il comando precedente premendo .

- Su bash/zsh i comandi sono *persistenti* (disponibili anche se il terminale viene riavviato).

Interruzione di un programma

È possibile interrompere forzatamente un programma, qualora non risponda per vari motivi o non si voglia attenderne la normale terminazione (Esempio tipico: programma in loop infinito).

- Per farlo si preme  +  (invio segnale **SIGINT**).

Visualizzazione della storia dei comandi

È possibile mostrare i comandi usati in precedenza. Le shell Unix memorizzano i comandi usati anche in sessioni precedenti.

- Su *nix, usare il comando **history**, su cmd, usare 

Ricerca nella storia dei comandi precedenti (solo bash/zsh/fish)

- Premendo  +  seguito da un testo, questo viene cercato in comandi lanciati in precedenza, anche in sessioni precedenti.

PREPARAZIONE DELL'AMBIENTE DI LAVORO (PER LAB01)

Preparazione Ambiente di Lavoro

1. Accendere il PC ed eseguire l'accesso
2. Accedere al sito del corso
3. Scaricare dalla sezione dedicata a questa settimana il materiale dell'esercitazione odierna
4. Spostare il file scaricato sul Desktop
5. Decomprimere il file sul Desktop
6. Aprire un terminale
 - ▶ Si scelga quello che si preferisce fra quelli proposti
7. Si comprenda in quale cartella ci si trova usando l'apposito comando
 - ▶ Generalmente il prompt si apre nella directory principale dell'utente
 - ▶ Nel caso in cui si usasse zsh dentro WSL, o Git Bash, il file system di windows è agganciato emulando un file system Unix.
8. Posizionarsi all'interno della cartella scompattata con l'ausilio del comando **cd** (change directory)
9. Verificare che il contenuto della cartella sia quello atteso
10. Scegliere un editor di testo per modificare i file sorgente
 - ▶ Il laboratorio è equipaggiato con diversi editor di testo
 - ▶ **NON** sono adatti per programmare i word processors (Libreoffice Writer, ■■ Word, ■■ WordPad...), né l'editor di testo incluso in Windows (Notepad).
 - ▶ Per questo lab, suggerisce l'utilizzo di **Notepad++**
 - ▶ consigliato prima di passare ad editor/IDE più sofisticati (come *VS Code*)
 - ▶ dobbiamo abituarci a scrivere codice corretto senza ausili
 - ▶ dobbiamo abituarci a compilare ed eseguire manualmente, da linea di comando

COMPILAZIONE ED ESECUZIONE DA RIGA DI COMANDO

Compilazione ed Esecuzione, comandi di base

Compilazione

Compilazione di un file (comando **javac**)

- **javac** NomeFileSorgente.java

È possibile usare la shell expansion per passare più file alla volta al compilatore:

- **javac** *.java

Esecuzione

Esecuzione di un programma Java (comando **java**)

- **java** NomeDellaClasse

Si noti che il compilatore *traduce* **file** sorgente in **file** binari, mentre l'interprete Java (la Java Virtual Machine) esegue una ed una sola **classe**.

javac (IL COMPILATORE) LAVORA SU *FILE*, **java** (L'INTERPRETE) SU *CLASSI*

jshell: Java REPL

A partire da Java 9, è stata introdotto in java l'interprete *REPL jshell*

- Sta per **R**ead-**E**val-**P**rint **L**oop *jshell* consente di effettuare al volo compilazione ed esecuzione
- Lo useremo ogni tanto per mostrarvi il risultato di alcune espressioni
- Non utile per lo sviluppo di intere applicazioni

Dietro le quinte

jshell non è magico: dietro le quinte compila in memoria, quindi lancia l'interprete sul bytecode risultante

Errori comuni

I seguenti errori sono comuni se non è chiaro cosa facciano interprete e compilatore:

- ❌ *java NomeClasse.java* (l'interprete non lavora su file)
 - ▶ Anche se da Java 9 potrebbe funzionare, ma solo perché l'interprete dietro le quinte compila il file e lo esegue
 - ▶ Noi dovremo imparare a costruire applicazioni vere, quindi: **prima *si compila*, poi *si esegue*!**
- ❌ *java path/to/NomeClasse.class* (l'interprete non lavora su file)

APPENDICE

Richiamo per gli esercizi del Lab01

Numeri complessi – breve ripasso

Siano $z, w \in \mathbb{C} : z = a + ib, w = c + id$, allora:

- Confronto: $z = w \iff a = c \wedge b = d$
- Somma algebrica: $z \pm w = a \pm c + i(b \pm d)$
- Prodotto: $z \cdot w = (a + ib)(c + id) = ac - bd + i(bc + ad)$
- Rapporto: $\frac{z}{w} = \frac{(a+ib)(c-id)}{(c+id)(c-id)} = \frac{ac+bd}{c^2+d^2} + i \frac{bc-ad}{c^2+d^2}$

STRUMENTI DI BASE DEL JDK

PROGRAMMAZIONE AD OGGETTI
C.D.L. INGEGNERIA E SCIENZE INFORMATICHE

Danilo Pianini — danilo.pianini@unibo.it

Slide compilate il: 2025-09-28

 [versione stampabile](#)

 [menu principale](#)