Strumenti di Base

Programmazione ad Oggetti – Lab01

Docenti: Roberto **Casadei**, Danilo **Pianini**
Tutor: Luca **Deluigi**

C.D.S. Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Campus di Cesena

4 aprile 2023

Outline

Outline

Tipi di piattaforme Java

Java platform

- **Java SE** (Java Platform, Standard Edition) – per applicazioni general-purpose su computer e server
- **Jakarta EE** (ex Java Platform, Enterprise Edition) – per applicazioni "enterprise" (mission-critical)
- **Java SE Embedded** – per applicazioni embedded
- **Java ME Embedded** (Java Platform, Micro Edition) – per dispositivi resource-constrained

Specifiche vs. implementazioni

- Una piattaforma Java include **specifiche** e **implementazioni (di riferimento)** per
  - ambiente d'esecuzione (macchina virtuale)
  - linguaggio (Java) e librerie (API)
  - strumenti di sviluppo (SDK)

Java SE Development Kit (JDK)

Java Development Kit (JDK)

JRE – Java Runtime Environment: solo interprete

JDK – JRE + tool per lo sviluppo di programmi (e.g. compilatore)

Noi faremo riferimento al JDK

Include il necessario per eseguire applicazioni Java (JRE), i tool (e molte librerie) per sviluppare applicazioni, e la relativa documentazione

I principali (non tutti) tool del JDK

javac – Compilatore Java

java – Java virtual machine, per eseguire applicazioni Java

javadoc – Per creare automaticamente documentazione in HTML

jar – Creazione di archivi compressi (anche eseguibili)

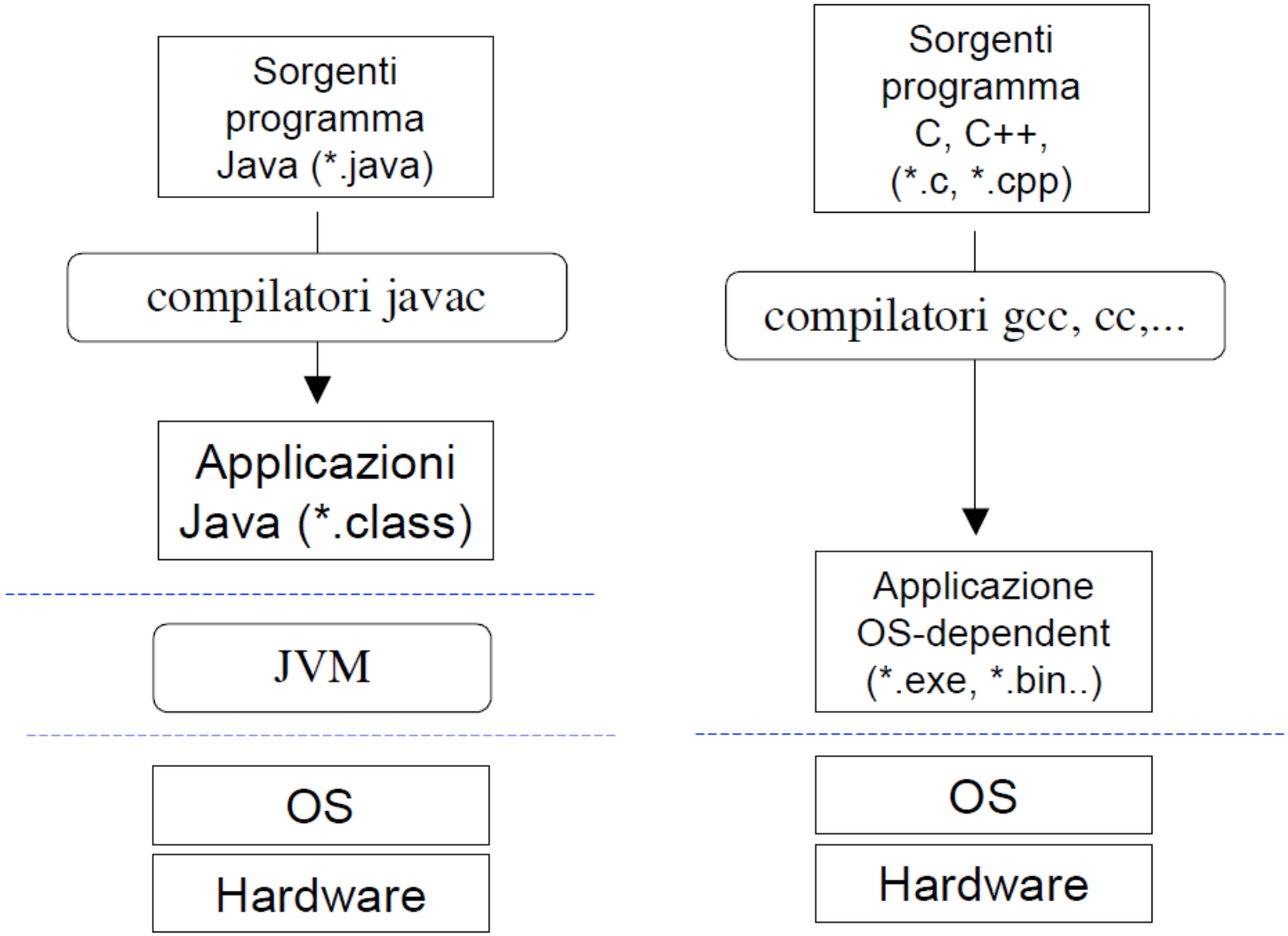
Java Development Kit (JDK)

- Java 17 – ultima versione LTS del JDK
  - Esistono varie versioni e implementazioni (OpenJDK, Oracle JDK, Eclipse OpenJ9, Amazon, Graal, Zulu...)
  - **OpenJDK** – implementazione *open-source*, **reference version**
    - Per capire se una JVM si comporta correttamente, si confronta il suo comportamento con OpenJDK.
  - Esistono vari distributori (e.g. Adoptium, Oracle) che offrono *binari pre-built* dell'implementazione OpenJDK (sorgenti)

Riferimenti per il corso

- Java 17, distribuita da Adoptium
- Adoptium: riferimento per il download del JDK
  - `https://adoptium.net/`
- Chi lo desidera può sperimentare con le versioni più recenti
  - NB: la compatibilità è solo all'indietro (*backwards compatibility*): è ok eseguire, su nuove JVM, applicazioni compilate per vecchie JVM
  - ...e spesso non completa...

Java Virtual Machine: Architettura a Runtime



Outline

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.
  1. **Livello logico**: riguarda come l'informazione è presentata all'utente
  2. **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)**
  - *nix (Linux, MacOS, BSD, Solaris...): vi è una unica radice, ossia `/`
  - Windows: c'è una root per FS, indicata da una lettera di unità (es. `C:`)
- File e cartelle possono essere individuati da un percorso o **path** (*assoluto*, dalla root; o *relativo*, da un certo punto del filesystem):
  - `/home/user/frameworkFS.jar` (percorso Unix)
  - `C:\Windows\win32.dll` (percorso Windows)
  - Altra sintassi: `.` (cartella corrente), `..` (cartella padre)

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
- spostare file e cartelle dentro altre cartelle
- rinominare file e cartelle
- eliminare file e cartelle

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

- Su Windows, esso è "Esplora risorse" (`explorer.exe`)
- Su MacOS, il principale è "Finder"
- Su Linux ne esistono diversi (Nautilus, Dolphin, Thunar, Astro...)

Interprete Comandi

Programma che permette di interagire con il S.O. mediante comandi impartiti in modalità testuale (non grafica), via linea di comando

- 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

Per una panoramica completa delle differenze

`http://www.faqs.org/faqs/unix-faq/shell/shell-differences/`

Sistemi Windows

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

- Eredita in realtà 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 ♡

Aprire un terminale in laboratorio I

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

Aprire un terminale in laboratorio II

Manjaro con Terminale ZSH via WSL2 (con configurazioni di Pianini)

- Avviare WSL tramite collegamento "Lancia WSL" presente nella cartella "WSL" del Desktop
  - Tasto destro sul file $\rightarrow$ *Esegui con PowerShell* (rispondere "T" alla domanda sui criteri d'esecuzione, e attendere)
- Attendere l'importazione della distro Linux (qualche minuto)
- Non aprire altre istanze finché la prima non è avviata!
- Questa è una shell nativa Linux, preconfigurata dal docente
  - (e quindi potenzialmente con personalizzazioni che dipendono dal gusto personale)

File system e terminale: cheat sheet

Operazione	Comando Unix	Comando Win
Ottenere aiuto / elenco comandi	`help`	`help`
Visualizzare la directory corrente	`pwd`	`echo %cd%`
Eliminare il file `foo` (non va con le cartelle!)	`rm foo`	`del foo`
Eliminare la directory di nome `bar`	`rm -r bar`	`rd bar`
Contenuto della directory corrente	`ls -alh`	`dir`
Cambiare unità disco (passare a D:)	–	`D:`
Passare alla directory di nome `baz`	`cd baz`	`cd baz`
Passare alla directory di livello superiore	`cd ..`	`cd..`
Spostare (rinominare) un file `foo` in `bar`	`mv foo bar`	`move foo bar`
Copiare il file `foo` in `baz`	`cp foo bar`	`copy foo bar`
Creare la directory di nome `bar`	`mkdir bar`	`md bar`

Uso intelligente del terminale I

I terminali moderni possono essere utilizzati in modo piuttosto efficace. Il numero ed il tipo di utilizzi "intelligenti" cambia da terminale a terminale e da sistema a sistema.

Autocompletamento

Sia *nix che Windows 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
  - Zsh e Fish in particolare posso "capire" come autocompletare i sottocomandi in modo intelligente.

Memoria dei comandi precendenti

Sia Bash/zsh che cmd offrono la possibilità di richiamare rapidamente i comandi inviati precedentemente premendo il tasto **FRECCIA SU**.

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

Uso intelligente del terminale II

Interruzione di un programma

È possibile interrompere forzatamente un programma, qualora non risponda per vari motivi o non si voglia attenderne la normale terminazione.

- Per farlo, sia su Windows che in *nix, si prema **CTRL+C** (invio segnale `SIGINT`).
- Esempio tipico: programma in loop infinito

Visualizzazione della storia dei comandi

- Su bash / zsh / fish eccetera, usare il comando `history`
- Su Windows cmd, usare il tasto F7

Ricerca nella storia dei comandi precedenti

- Premendo **CTRL+R** seguito da un testo da cercare, i sistemi *nix supportano la ricerca all'interno dei comandi lanciati recentemente, anche in sessioni utente precedenti. Non disponibile su Windows.

Outline

Preparazione Ambiente di Lavoro

- Accendere il PC ed eseguire l'accesso
- Accedere al sito del corso
- Scaricare dalla sezione dedicata a questa settimana il materiale dell'esercitazione odierna
- Spostare il file scaricato sul Desktop
- Decomprimere il file sul Desktop

Preparazione Ambiente di Lavoro

- Aprire un terminale
- Si scelga l'opzione che si preferisce fra quelli proposti
- 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.
- Posizionarsi all'interno della cartella scompattata con l'ausilio del comando `cd` (change directory)
- Verificare che il contenuto della cartella sia quello atteso

Preparazione Ambiente di Lavoro

- Scegliere un editor di testo per modificare i file sorgente
  - Il laboratorio è equipaggiato con diversi editor di testo: JEdit, Notepad++, Visual Studio Code...
  - **NON** sono adatti per l'apertura e la modifica di file Java (né di alcun altro linguaggio di programmazione) i word processors (Libreoffice Writer, Microsoft Word, Microsoft 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

Outline

Compilazione ed Esecuzione, comandi di base

Compilazione

Compilazione di una classe (comando javac)

- `javac NomeFileSorgente.java`
- `javac *.java` compila tutti i sorgenti nella directory corrente

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**.
Il compilatore Java lavora su file, l'interprete Java su classi.

Errori comuni

I seguenti errori sono da evitare:

- NO: `java NomeClasse.java` (l'interprete non lavora su file)
- NO: `java path/to/NomeClasse.class` (l'interprete non lavora su file)

Da Java 9: interpretazione "al volo"

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

In sistemi con `jshell`, lanciare l'interprete `java` su un file non compilato

- Ad esempio, `java HelloWorld.java`

può causare l'esecuzione!

Dietro le quinte

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

Noi costruiamo applicazioni: **prima** *si compila*, **poi** *si esegue*!

Outline

A1 – Operazioni con i Numeri Complessi

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: $\frac{z}{w} = \frac{(a+ib)(c-id)}{(c+id)(c-id)} = \frac{ac+bd}{c^2+d^2} + i\frac{bc-ad}{c^2+d^2}$