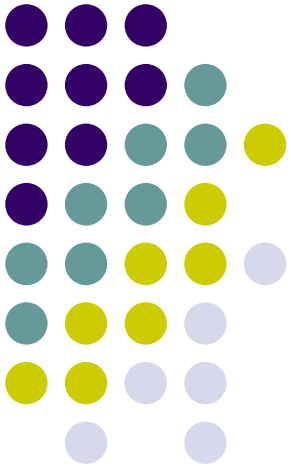
Università di Napoli Federico II - Dip. di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica

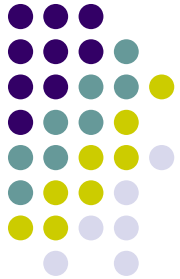
DIE TI

Corso di Programmazione

Linguaggio JAVA



Caratteristiche principali (1/2)



- Originario di un progetto di Sun Microsystem, “Green”, è stato ufficializzato nel 1995.

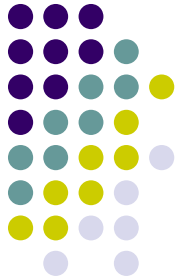
JAVA è un linguaggio di programmazione..

Object Oriented	⇒	*Utilizzabilità e riusabilità*
Indip dall’architettura	⇒	*Portabilità e longevità*
Distribuito e Multithread.	⇒	*Applicazioni Web-based*

Versioni più aggiornate di Java SUN (http://java.sun.com)

Java SE 11, Java SE 17, Java SE 21

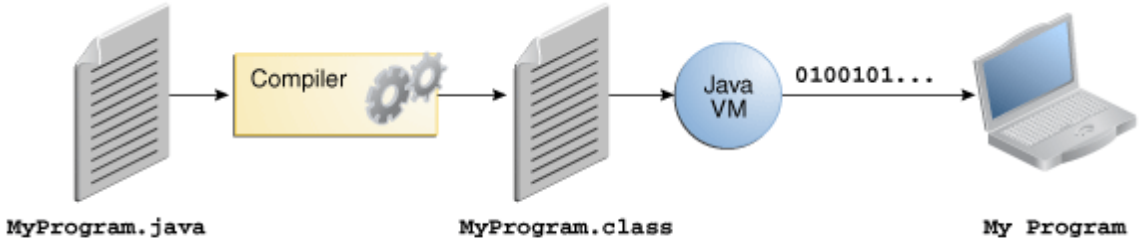
La filosofia di Java.



- Il Computer è la rete
  - Le applicazioni Java dovranno poter essere eseguite su nodi di elaborazioni distribuiti
- Scrivi una volta...esegui ovunque
  - Le applicazioni dovranno poter essere realizzate “una tantum” per poi essere eseguite su un qualsiasi calcolatore (portabilità del codice)

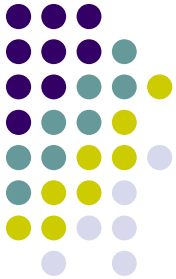
Processo di sviluppo in JAVA

- In Java, il codice sorgente viene scritto in file di testo con estensione ***.java***
- Viene compilato ma la compilazione NON produce un file oggetto per la macchina su cui è stata fatta la compilazione bensì un oggetto in bytecode in un file con estensione ***.class***
- In questo modo si ottiene la portabilità, ***il bytecode generato è lo stesso per tutte le architetture***
- Per ottenere un codice eseguibile è necessario che sulla macchina sia presente la ***Virtual Machine Java (Java VM)*** che è *disponibile per diversi sistemi operativi* e ha il compito di interpretare il bytecode per la macchina su cui si trova

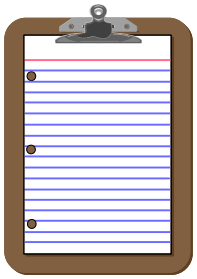


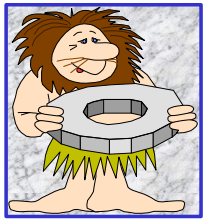
https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Compilazione e traduzione



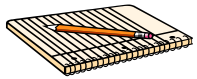
Codice JAVA





Compilatore
JAVA

Bytecode

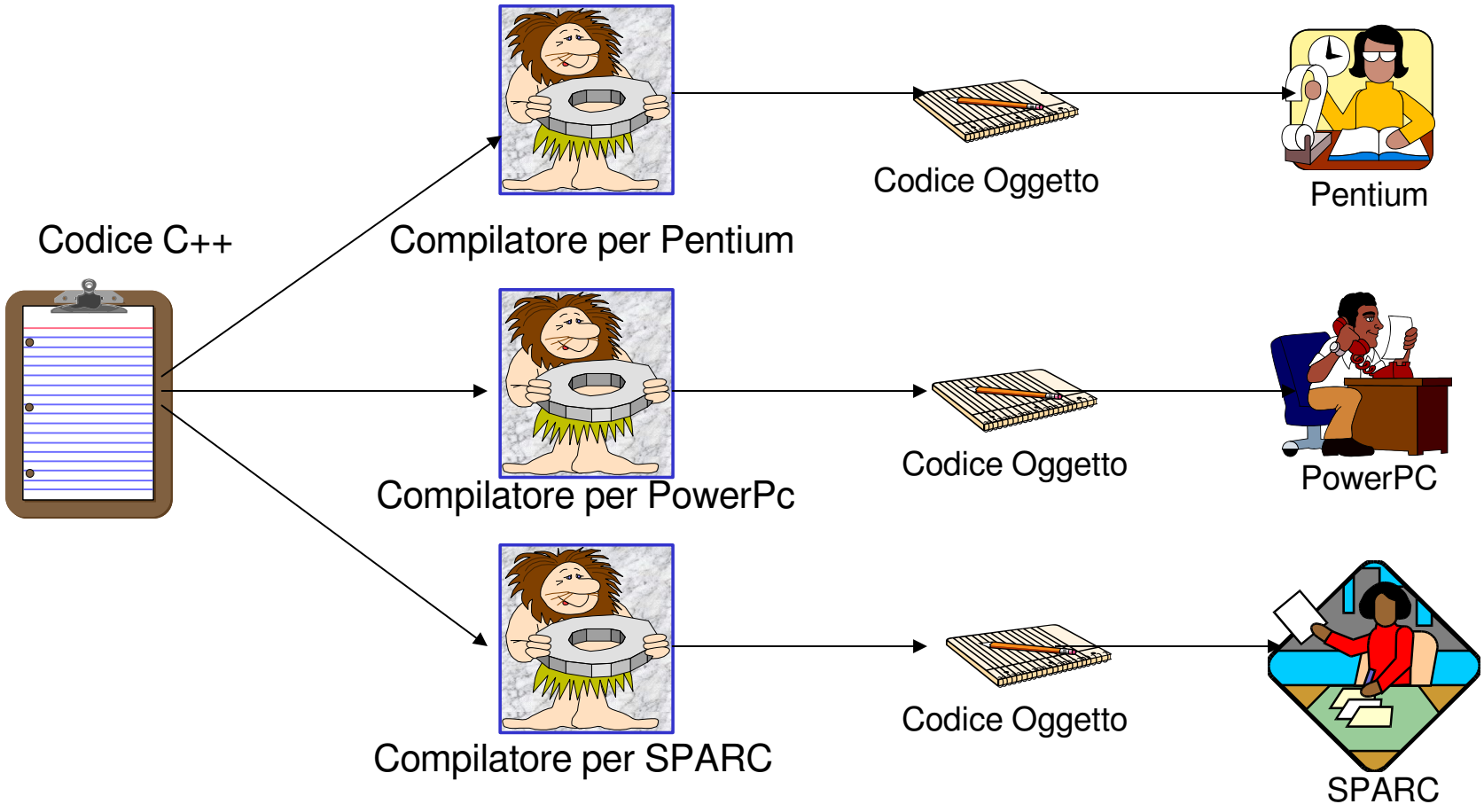


Interprete
Pentium

Interprete
PowerPC

Interprete
SPARC

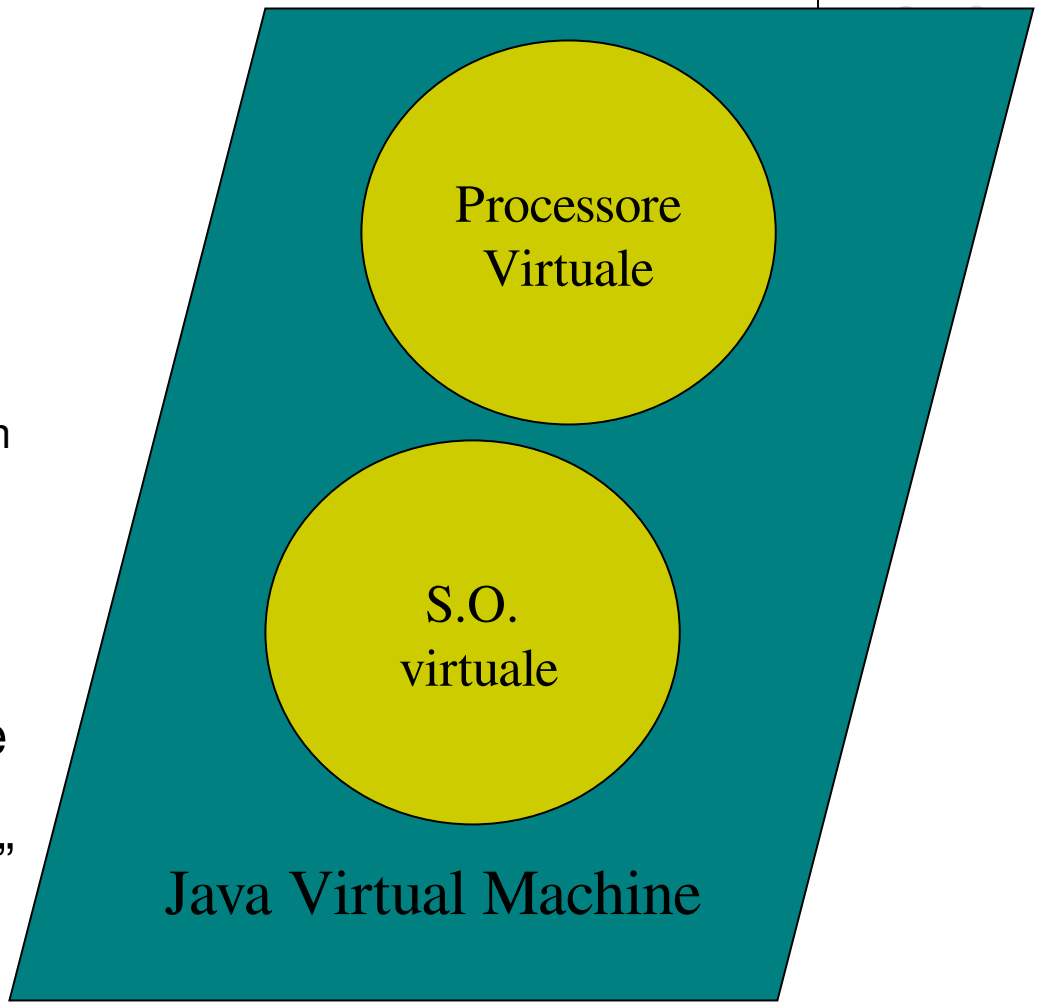
Confronto con il linguaggio C++



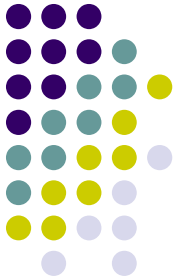
La macchina virtuale....

- In realtà la Java Virtual Machine è più di un semplice interprete...è – appunto - una «macchina virtuale» costituita da:
  - Un processore virtuale
  - un sistema operativo virtuale

  che costituiscono, nel loro insieme, un ambiente virtuale in cui eseguire il codice Java
- In altre parole il compilatore Java genera il codice per un processore "virtuale". Il codice Java è compilato riferendosi a questo processore "campione" ("tipo") che non è quello reale.

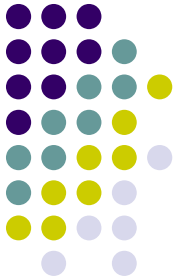


Sviluppo di una applicazione Java



- La preparazione e l'esecuzione di un programma Java consiste di 5 fasi
- Editazione (va bene un qualsiasi editor)
- Compilazione e Collegamento
- Caricamento
- Verifica
- Esecuzione

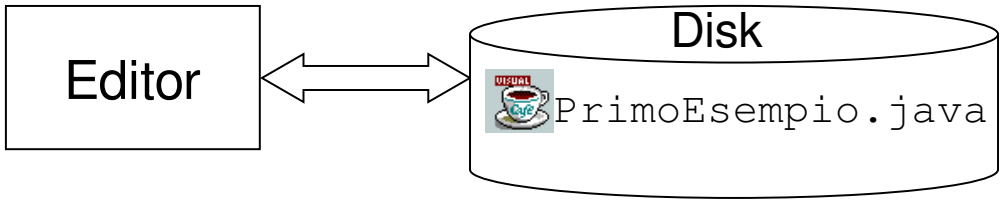
Sviluppo di una applicazione Java



La preparazione e l'esecuzione di un programma Java consiste di 5 fasi

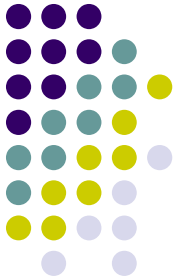
1) La fase di EDIT

```
C:\PRG>edit PrimoEsempio.java
```



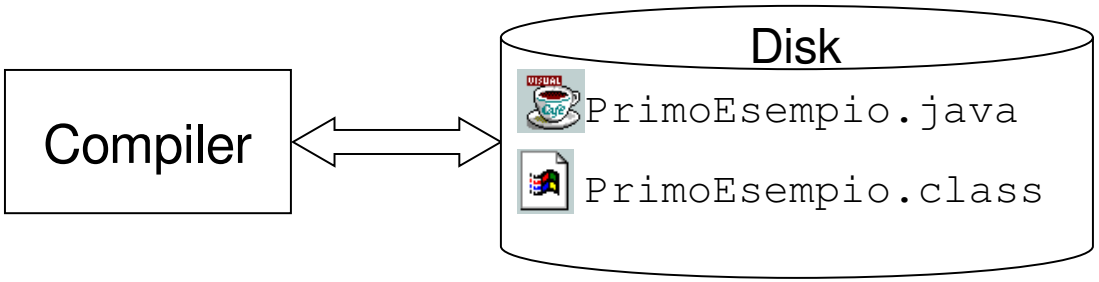
Viene creato il file contenente il programma e, successivamente memorizzato nel disco.
ATTENZIONE: ***Il nome del file deve coincidere con quello della classe (case sensitive !)***

Sviluppo di una applicazione Java



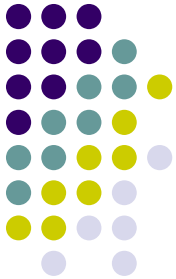
2) La fase di COMPILAZIONE

```
C:\PRG>javac PrimoEsempio.java
```



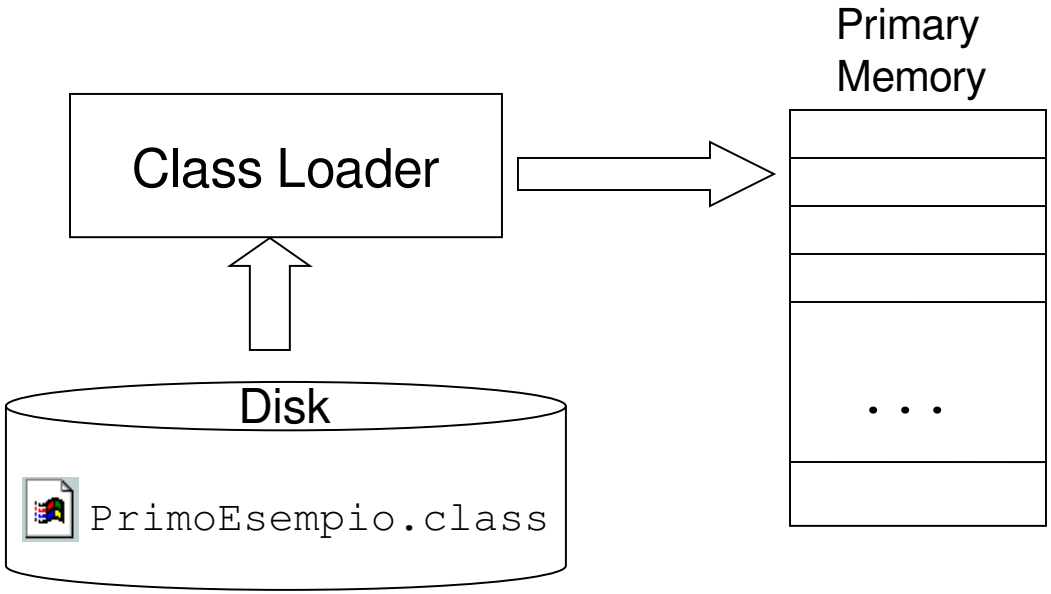
Il compilatore JAVA traduce il programma nel corrispondente bytecode, il linguaggio compreso dall'interprete java, e lo salva su disco

Sviluppo di una applicazione Java



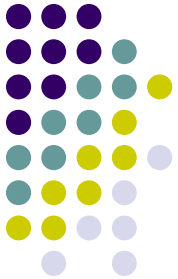
3) La fase di CARICAMENTO

```
C:\PRG>java PrimoEsempio
```

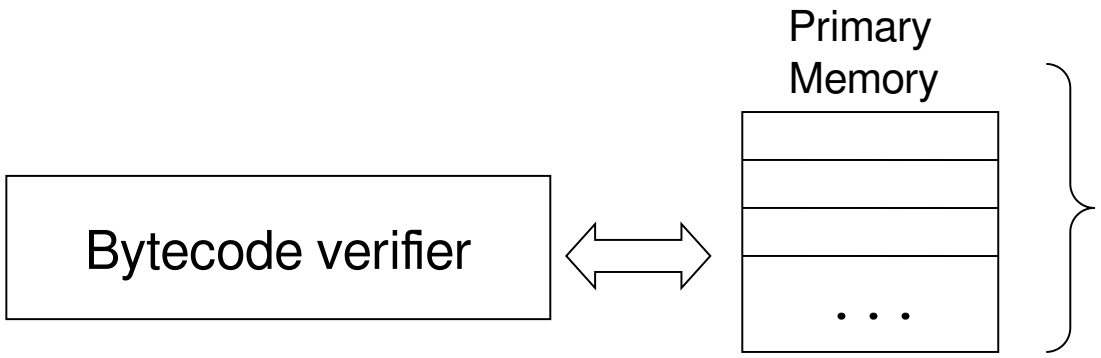


Il Class loader carica il bytecode nella memoria principale

Sviluppo di una applicazione Java

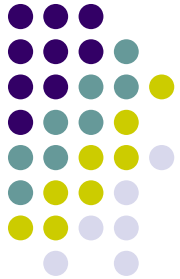


4) La fase di VERIFICA del bytecode

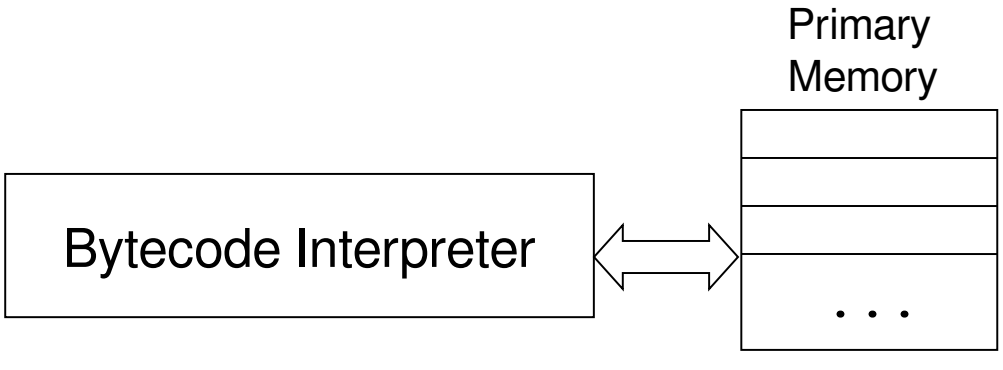


Il "bytecode verifier" ha il compito di validare il bytecode e assicurarsi che non si violino i vincoli di sicurezza dell'ambiente JAVA

Sviluppo di una applicazione Java

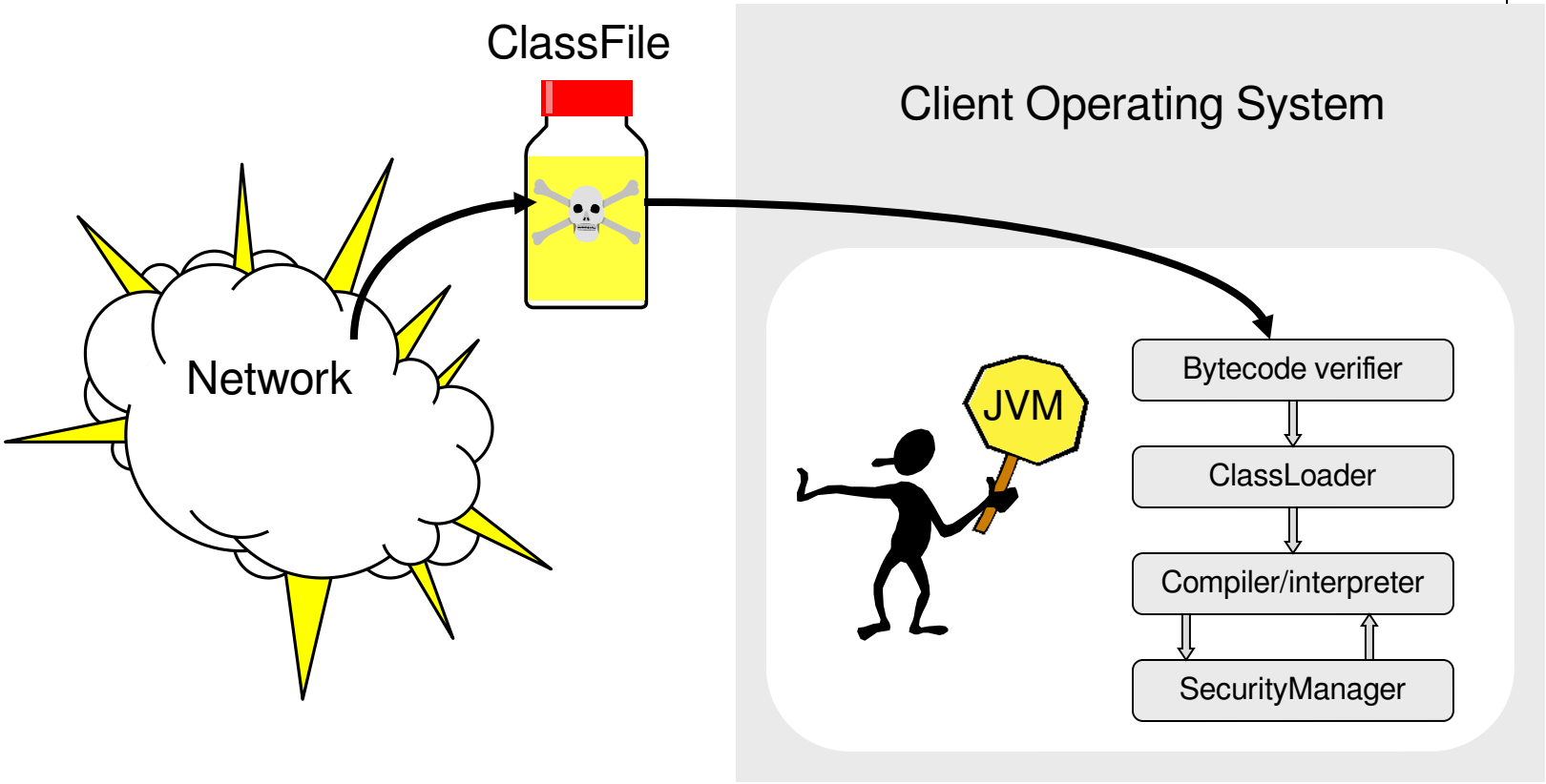


5) La fase di ESECUZIONE



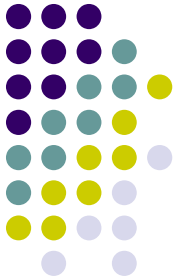
L'interprete del bytecode traduce le istruzioni del bytecode nelle istruzioni compatibili con il processore.

Il modello di sicurezza



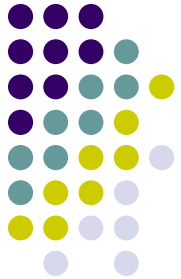
Il Security Manager implementa le politiche di sicurezza per l'accesso alle risorse (molto importante per le Applet)

Loader e Verifier



- Il Class Loader: carica solo le classi necessarie per l'esecuzione del programma;
- Il Bytecode Verifier controlla che:
  - il codice rispetti le specifiche della JVM;
  - il codice non violi l'integrità del sistema;
  - il codice non generi stack overflow o underflow;
  - non vi siano conversioni di tipo illegali (e.g. la conversione di interi in puntatori).

Applicazioni JAVA



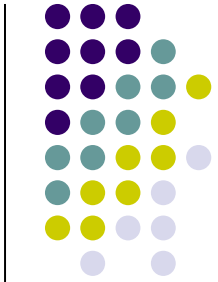
- Un applicazione è un programma eseguibile autonomamente

Per rendere eseguibile un'applicazione Java occorre che una classe faccia da "punto di avvio".
Questa classe ha solo bisogno di un metodo ***main()***, il primo ad essere invocato durante l'esecuzione dell'applicazione.

```
public static void main (String args[]){
        ……..
    }
```

- **Public** indica che il metodo è a disposizione di altre classi e oggetti
- **static** indica che *main()* è un metodo invocabile anche se non esistono oggetti (vedremo meglio poi)
- **void** indica che il metodo *main()* non restituisce un valore
- **main()** accetta come parametro un array di stringhe, utilizzato per passare argomenti dalla riga di comando

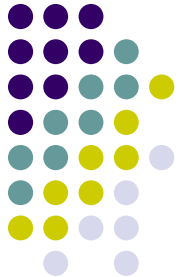
Un po' di storia



- Da Wikipedia.it:
  - Java è stato creato a partire da ricerche effettuate alla Stanford University agli inizi degli anni novanta.
  - Nel 1992 nasce il linguaggio Oak (in italiano "quercia"), prodotto da Sun Microsystems e realizzato da un gruppo di esperti sviluppatori capitanati da James Gosling.
  - Questo nome fu successivamente cambiato in Java (una varietà di caffè indonesiana; il logo adottato è una tazzina per tale bevanda) per problemi di copyright: il linguaggio di programmazione Oak esisteva già.

https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)

Ancora un po' di storia



- Da Wikipedia.it:
  - Java fu annunciato ufficialmente il 23 maggio 1995 a SunWorld.
  - Il 13 novembre 2006 la Sun Microsystems ha distribuito la sua implementazione del compilatore Java e della macchina virtuale sotto licenza GPL.
  - L'8 maggio 2007 Sun ha pubblicato anche le librerie, tranne alcuni componenti non di sua proprietà, sotto licenza GPL, rendendo Java un linguaggio di programmazione la cui implementazione di riferimento è libera.
  - Il linguaggio è definito da un documento chiamato The Java Language Specification, spesso abbreviato JLS. La prima edizione del documento è stata pubblicata nel 1996.
    - Da allora il linguaggio ha subito numerose modifiche e integrazioni, aggiunte di volta in volta nelle edizioni successive.
    - Al Gennaio 2023 la versione più recente delle specifiche è la Java SE 21 Edition.

https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)

Un po' di numeri (e storia...)

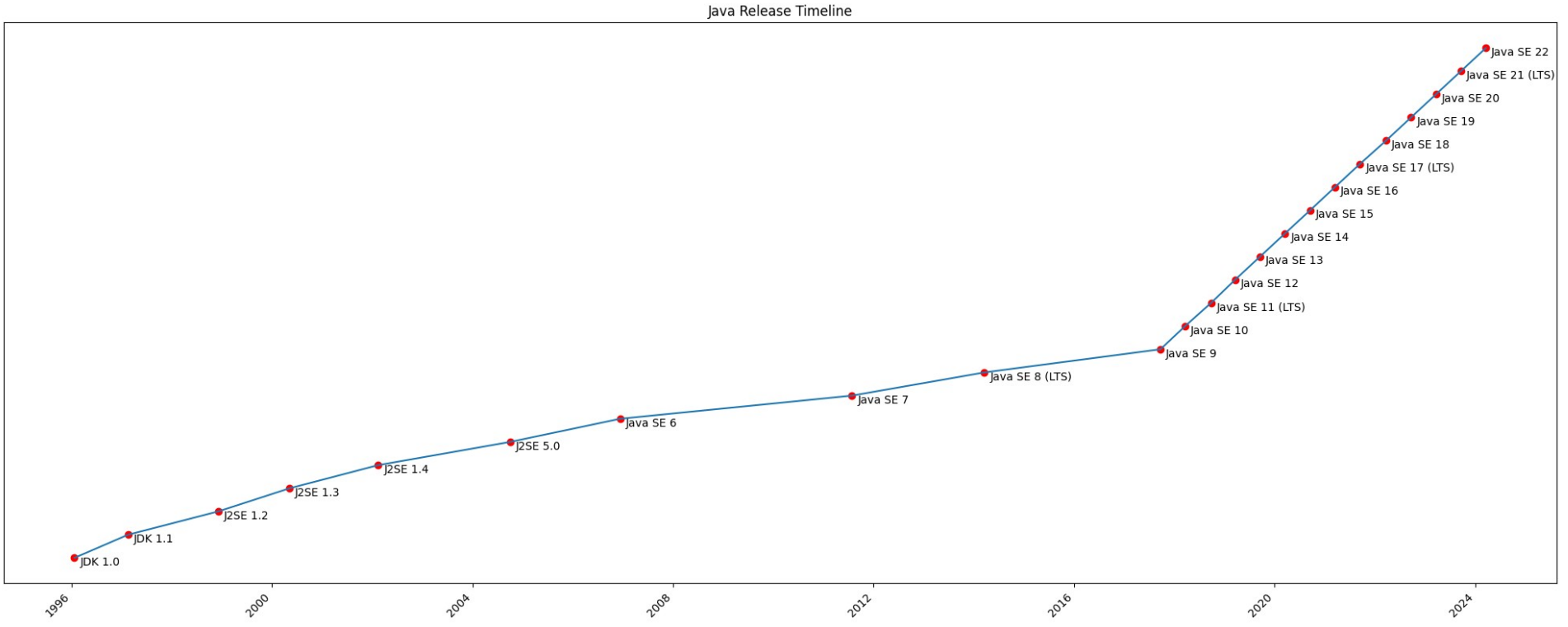
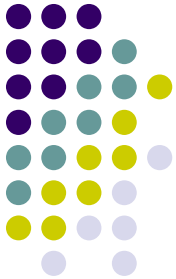


Grafico ottenuto da:

https://en.wikipedia.org/wiki/Java_(programming_language)

https://en.wikipedia.org/wiki/Java_version_history

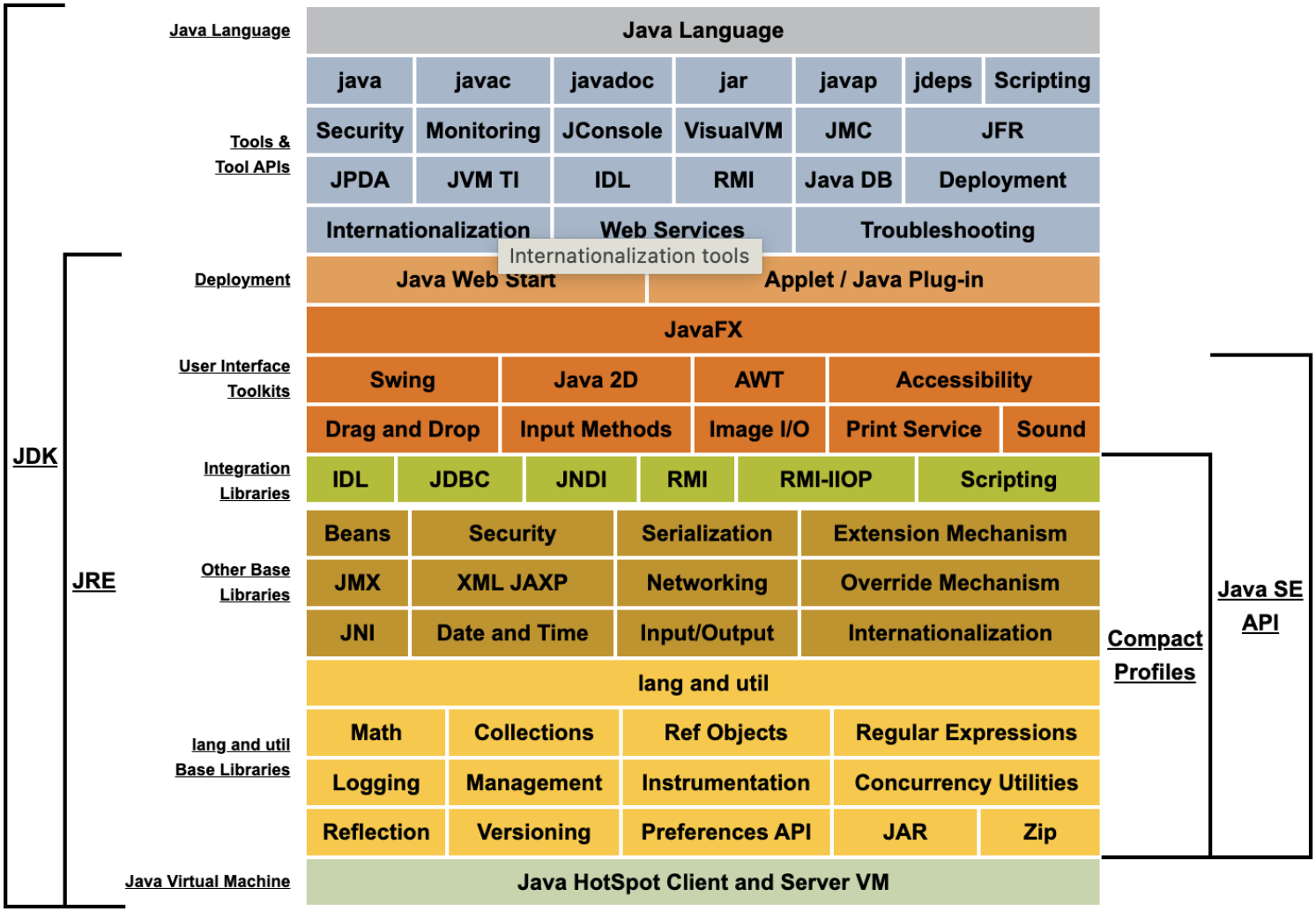
JDK e JRE



- **JDK** (**Java Development Kit**) comprende tutti gli strumenti necessari sia allo sviluppo che all'esecuzione di applicazioni Java, incluso il compilatore e la Java Virtual Machine. (JSE è la versione Oracle della JDK)
- **JRE (Java Runtime Environment)** è sostanzialmente la Java Virtual Machine e tutto ciò che è necessario alla esecuzione di applicazioni Java
- **Per eseguire** applicazioni Java è sufficiente installare la JRE
- **Per sviluppare** programmi Java è necessario installare la JDK

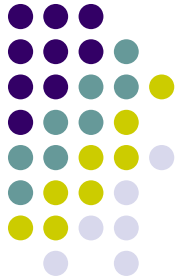
Tecnologie Java: JSE

Oracle Java **Platform** Standard Edition (8)



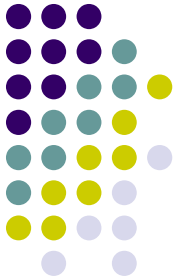
https://docs.oracle.com/javase/8/docs/index.html

La piattaforma Java, i servizi.



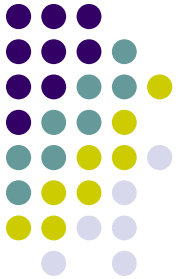
- L'ambiente Java è corredato da una estesa libreria di classi organizzata in "Pacchetti" (*packages*)
  - Possiamo vedere tali librerie come un insieme di servizi "pronti per l'uso" e possiamo classificarli secondo il livello di supporto che forniscono
    - Servizi di Base: il linguaggio (Java Core Classes)
    - Servizi di alto livello "orizzontali"
    - Servizi di alto livello "verticali"

Servizi di Base



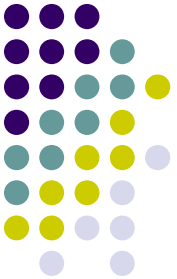
- package java.lang
  - Boolean, Byte, Character, Class, ClassLoader, Double, Float, Long, Math, Number, Object, Process, Runtime, RuntimePermission, SecurityManager, Short, String, StringBuffer, System, Thread...Void
- package java.io
  - BufferedReader, BufferedWriter, ByteArrayInputStream,... File, FileInputStream, FileOutputStream, FilePermission, FileReader, FileWriter, InputStream, LineNumberInputStream, RandomAccessFile Reader,Writer

Servizi di alto livello “orizzontali”



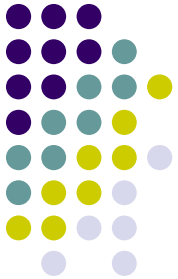
- Per la costruzione di interfacce grafiche
  - package java.awt, javax.swing …
- Per l’accesso ai DB
  - package java.sql
- Per l’accesso alla rete
  - package java.net
- Per le applet
  - package java.applet

Servizi di alto livello “verticali”



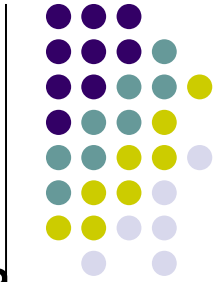
- Package di classi che possono essere aggiunti a quelli installati
  - Per la riproduzione dei filmati
    - JMF (Java Multimedia Framework)
  - Per il commercio elettronico
    - JECF (Java Electronic Commerce Framework)

Riferimenti



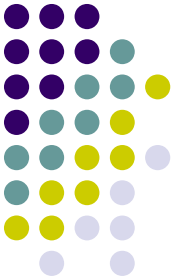
- Programmare in Java – Prima di cominciare
- Programmare in Java – Capitolo 1
- Risorse on line
  - http://java.sun.com
  - https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html
  - https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)
  - https://en.wikipedia.org/wiki/Java_(programming_language)
  - https://en.wikipedia.org/wiki/Java_version_history
  - https://docs.oracle.com/javase/8/docs/index.html

La piattaforma Java (1/2)

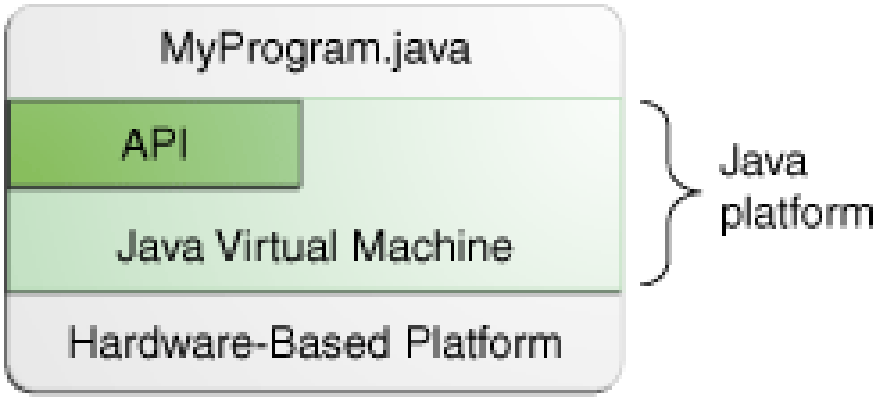


- L'insieme dell'hardware e del software necessario alla **esecuzione** di un programma è spesso indicato con il termine «piattaforma»
- Il nome Java è utilizzato sia per indicare il linguaggio di programmazione che la relativa *piattaforma*, che si distingue dal concetto generale perché *è costituita **solo** dal software necessario alla esecuzione di un programma Java*
- La piattaforma Java comprende:
  - *Java Virtual Machine*
  - *Java Application Programming Interface* (API)

La piattaforma Java (2/2)



- L'API è una vasta raccolta di componenti software pronti all'uso che forniscono molte funzionalità utili. È raggruppata in librerie di classi e interfacce correlate; queste librerie sono conosciute come *packages* (pacchetti)



https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html