



**Agent and Object Technology Lab**  
Dipartimento di Ingegneria dell'Informazione  
Università degli Studi di Parma



# Ingegneria del software A

## Introduzione

**Prof. Agostino Poggi**

## ♦ Programma

- Fondamenti di ingegneria del software - Qualità del software. Il processo di produzione del software. Metodologie per lo sviluppo del software. Analisi di sistemi software. Progettazione di sistemi software. Strumenti per la progettazione e realizzazione di sistemi software.
- Sviluppo di sistemi software orientati agli oggetti - Analisi orientata agli oggetti dei sistemi software. Progettazione e sviluppo di sistemi software orientati agli oggetti. Design pattern.
- Il linguaggio Java - Oggetti e classi. Ereditarietà e polimorfismo. Eccezioni. Input/output. Collezioni di oggetti. Programmazione concorrente.

## ♦ Attività di esercitazione

- Le esercitazioni saranno mirate all'apprendimento delle tecniche e all'uso di strumenti per progettazione e realizzazione di sistemi software principalmente attraverso l'uso del linguaggio di programmazione Java.

- ◆ Prova scritta
  - Un insieme di domande a risposta aperta sugli argomenti presentati a lezione
- ◆ Progetto software
  - Può essere accoppiato alla prova pratica dell'esame di Reti di Calcolatori, all'Internato di Laboratorio e alla tesi finale
- ◆ La partecipazione attiva alle esercitazioni
  - La frequenza permette di avere dei punti aggiuntivi

- ◆ Java SE, JDK 6 (77 MB)
  - <http://java.sun.com/>
- ◆ Eclipse IDE for Java Developers (85 MB)
  - <http://www.eclipse.org/>
- ◆ Visual Paradigm SDE for Eclipse
  - <http://www.visual-paradigm.com/product/sde/ec/>
- ◆ Ant
  - <http://ant.apache.org/>
- ◆ JUnit
  - <http://www.junit.org>

- ◆ Slide del corso
  - <http://www.ce.unipr.it/people/poggi>
- ◆ Java: documentazione API e tool, tutorial, convenzioni sul codice, ...
  - <http://java.sun.com/javase/6/docs/>
- ◆ Eckel: “Thinking in Java” e “Thinking in Patterns with Java”
  - <http://www.mindview.net/Books/>

- ♦ Ian Sommerville. Ingegneria del software, 8/ed, Pearson Education, 2007.
- ♦ Simon Bennett, John Skelton, Ken Lunn, UML, Mc Graw Hill, 2001.
- ♦ E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns, Addison-Wesley, 1994.
- ♦ Jim Arlow, Ila Neustadt. UML 2 e Unified Process: Analisi e progettazione Object-Oriented, 2/ed, Mc Graw Hill, 2007.

♦ Office:

Palazzina 1 – Sede Scientifica Ingegneria  
Viale G. Usberti 181/A

♦ Phone number: 0521 90 5728

♦ Fax number: 0521 90 5723

♦ Email: [agostino.poggi@unipr.it](mailto:agostino.poggi@unipr.it)

- ♦ L'ingegneria del software tratta della realizzazione di sistemi software di dimensioni e complessità tali da richiedere uno o più team di persone
- ♦ La sua nascita e il suo sviluppo sono una diretta conseguenza dell'aumento di **complessità** del software



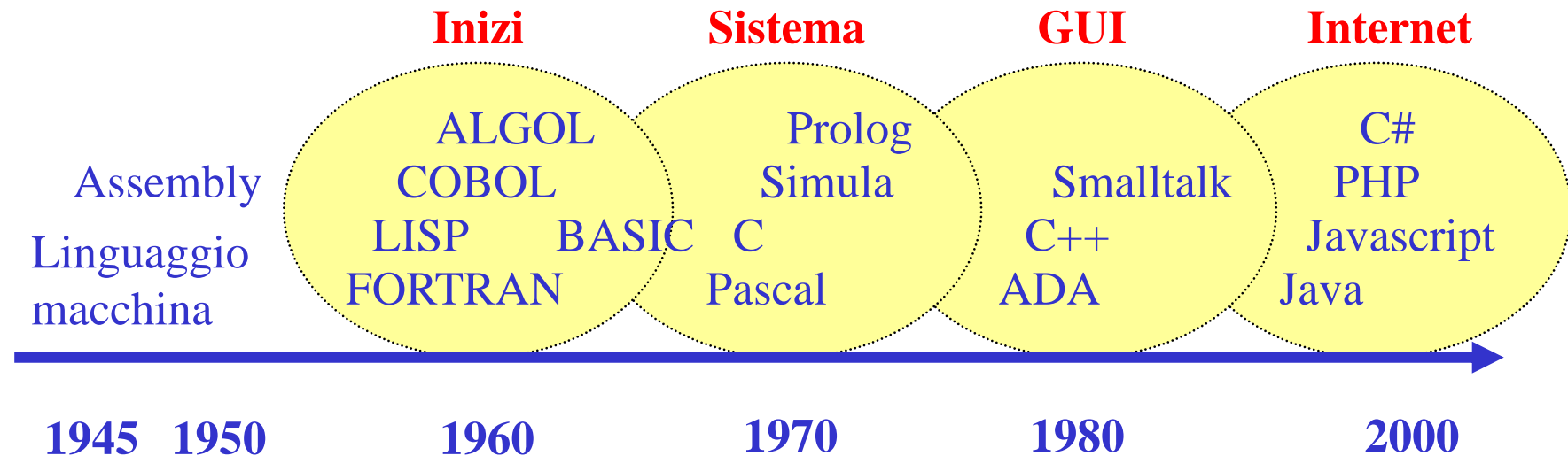


- ♦ L'ingegneria del software è l'approccio **sistematico** allo sviluppo, all'operatività, alla manutenzione ed al ritiro del software
- ♦ L'ingegneria del software è la disciplina **tecnologica e manageriale** per la produzione sistematica e la manutenzione dei prodotti software sviluppati e modificati entro **tempi e costi** preventivati
- ♦ L'ingegneria del software è un insieme di teorie, metodi e strumenti, sia di tipo tecnologico che organizzativo, che consentono di produrre applicazioni con le desiderate caratteristiche di **qualità**

- ◆ Creatività e metodo
  - Il software è un prodotto dell'ingegno e non di un processo industriale
  
- ◆ Prodotto e processo
  - L'ingegneria classica (civile, meccanica) progetta il prodotto e anche il processo industriale
  - L'ingegneria del software (spesso) progetta solo il prodotto e non utilizza un processo industriale formalizzato

- ◆ L'ingegneria del software nasce con la conferenza NATO del 1968
- ◆ I maggiori sviluppi si hanno dagli anni 1990 con lo sviluppo delle **tecnologie orientate agli oggetti**
  - Programmazione orientata agli oggetti
  - UML
  - Design pattern
  - Java

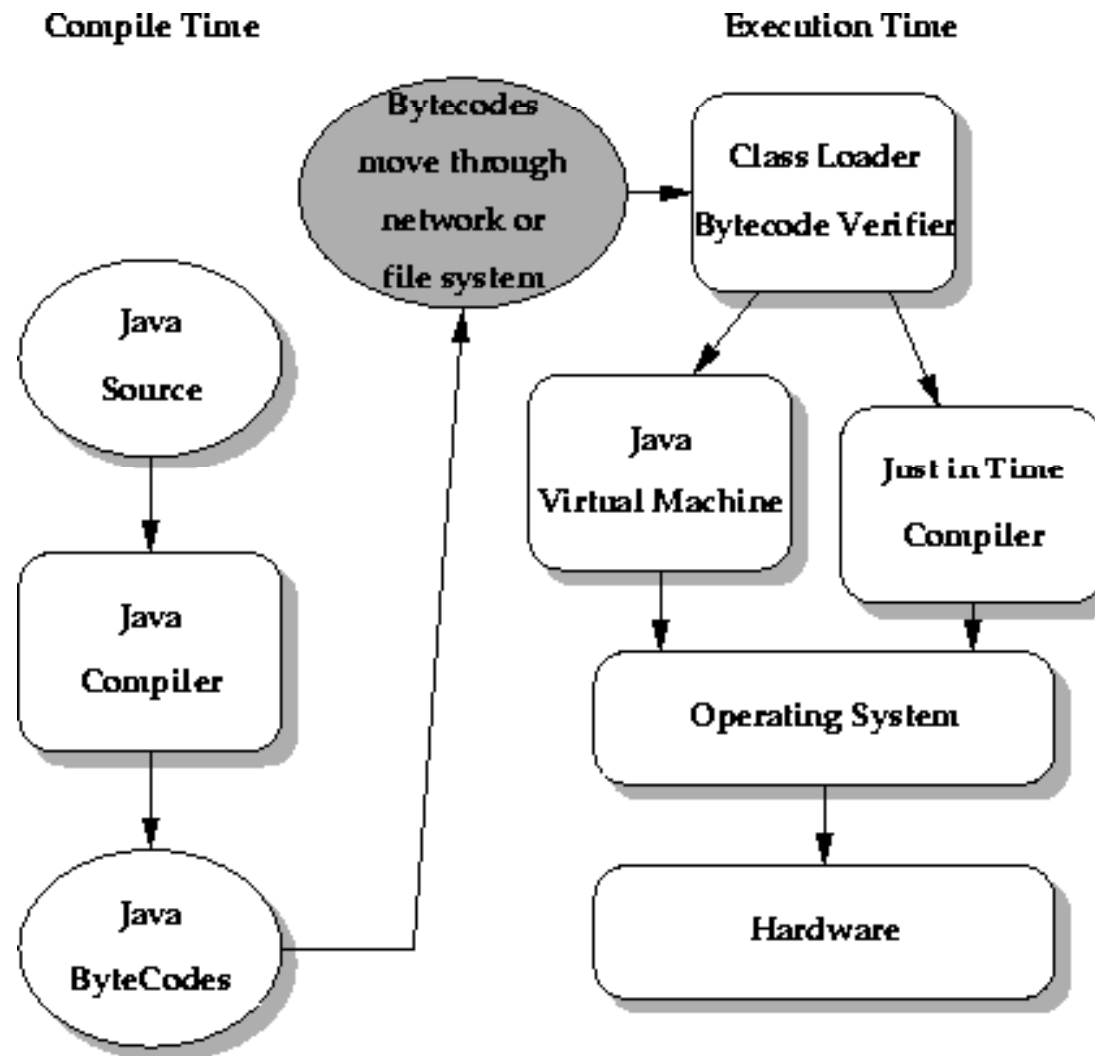




- ♦ [http://www.oreilly.com/news/graphics/prog\\_lang\\_poster.pdf](http://www.oreilly.com/news/graphics/prog_lang_poster.pdf)
- ♦ <http://www.levenez.com/lang/history.html>
- ♦ <http://www.byte.com/art/9509/sec7/art19.htm>
- ♦ [http://www.princeton.edu/~ferguson/adw/programming\\_languages.shtml](http://www.princeton.edu/~ferguson/adw/programming_languages.shtml)

- ◆ Insieme di tecnologie
  - Linguaggio orientato agli oggetti che migliora il C++
  - Strumenti sia a compile-time che a run-time
  
- ◆ Lancio ufficiale nel 1995
  - Piattaforma sviluppata da Sun Microsystems
  - Open source dal 2006-07, iniziativa OpenJDK
  
- ◆ Svincola software da hardware e sistema operativo
  - Sistemi web-oriented (lato client e server)
  - Sistemi embedded e sistemi mobili

- ◆ Problema della distribuzione su web: programmi strettamente legati ad una piattaforma
  - Piattaforma: combinazione di hardware e software di sistema
- ◆ Byte code
  - Un compilatore Java (javac) non produce codice eseguibile da una particolare piattaforma, come farebbe un compilatore C
  - A differenza del linguaggio macchina, il byte code Java resta uguale per tutte le piattaforme
- ◆ Macchina virtuale (VM)
  - La VM (Java) trasforma il byte code in codice nativo



- ♦ Java ha una sintassi simile al C
  - Ma non è una estensione del C, come il C++
  - Un compilatore Java non accetta codice C
  - Modifiche sostanziali per convertire un programma da C a Java
- ♦ Java è semplice: facile da capire e scrivere
  - Codice C medio, al rilascio, ha un bug ogni 55 righe di codice
  - Java ha sintassi minima - No “syntactic sugar” (es. operatori)
  - Ma più funzionalità del C, grazie all’ampia libreria di classi
- ♦ Allocazione e deallocazione automatica della memoria
  - Nessuna chiamata a malloc(), free() e distruttori
  - Metà dei bug in C e C++ sono legati alla gestione della memoria
- ♦ Riferimenti controllati
  - Nessuna algebra dei puntatori e accessi arbitrari alla memoria



- ♦ Un programma è una collezione di oggetti
  - Gli oggetti sono composti da variabili e metodi, e interagiscono attraverso lo scambio di messaggi
  - Gli oggetti sono definiti attraverso delle classi
- ♦ La programmazione è basata sull'**astrazione** dei dati e sull'uso dell'**ereditarietà** tra le classi
  - L'astrazione rende i programmi più semplici e facili da leggere
  - L'ereditarietà fornisce un migliore riutilizzo del codice
- ♦ Quindi lo sviluppo e rilascio del software è più veloce e il codice è modulare, più affidabile e mantenibile

- ◆ Java è cross-platform
  - Non solo codice sorgente – come C – ma anche bytecode
- ◆ Portare programmi Java su una nuova piattaforma
  - Portare la macchina virtuale (“interprete”)
  - Portare parte delle librerie (AWT, I/O, rete...)
  - Compilatore e gran parte delle librerie sono scritte in Java
- ◆ Eliminazione di costrutti non specificati o dipendenti dalla piattaforma
  - Interi sempre a 32 bit
  - Virgola mobile secondo lo standard IEEE 754

- ◆ Verifica del byte code e sandbox
  - Esecuzione sicura di codice scaricato da fonti non fidate
  - Le applet non possono accedere al disco o alla rete
  - Verifica preventiva sul byte code dell'applet, controlli a run time
  - Al più, una applet può bloccare la VM, ma non l'intero sistema
- ◆ No puntatori
  - I programmi Java non possono accedere arbitrariamente alla memoria
- ◆ Java ha controllo dei tipi forte
  - Si può convertire un int in long, un byte in short...
  - Ma non un int in boolean
  - Controllo su riferimenti e cast tra classi
- ◆ Gestione delle eccezioni per errori attesi o inattesi

- ◆ Il byte code può essere compilato al volo (just-in-time) per ottenere prestazioni simili a codice nativo
- ◆ Esistono compilatori che generano codice nativo per una particolare piattaforma
- ◆ Non si possono limare le prestazioni operando a basso livello come in C, ma si possono ottenere risultati adeguati per molti ambiti
- ◆ Grossi programmi scritti in Java
  - Eclipse (IDE), Limewire (file sharing), HotJava (browser), jEdit (text editor), JBoss (application server), Tomcat (web server), Xerces (parser XML), Xalan (processore XSLT), javac (compilatore)

- ◆ Un programma Java può avere diversi thread di esecuzione paralleli
- ◆ Reazioni rapide all'input dell'utente o alle richieste dei client
- ◆ Fonte di bug difficili da trovare (corse, deadlock, ...)
  - Problema inerente alla programmazione parallela
- ◆ Java fornisce
  - Una libreria molto semplice per gestire i thread
  - Una ricca libreria per accesso concorrente alle risorse

- ◆ Java non ha una fase di link esplicita
- ◆ Il compilatore
  - Genera un file di byte code (.class) per ogni classe del programma
  - Risolve le dipendenze e compila tutte le classi necessarie
- ◆ Il programma può caricare classi a runtime
  - Per esempio, un browser può caricare alcune applet
  - Le classi necessarie sono cercate e caricate dinamicamente

- ◆ Garbage collection: la raccolta della memoria non più utilizzata di Java
- ◆ Non c'è bisogno di allocare o deallocare la memoria esplicitamente
  - La memoria è allocata quando necessario e liberata quando non più utilizzata
  - Oggetti sempre in heap
- ◆ L'algoritmo usato per la garbage collection è diverso sulle varie VM