SETTIMANA 1

Fondamenti di Informatica 1
Presentazione del Corso

Informazioni importanti

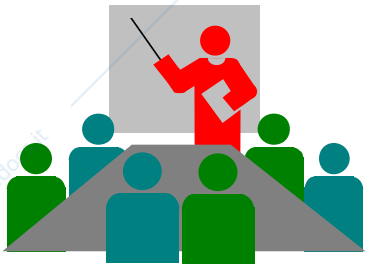
- Sono ammessi a seguire questo corso
  - Gli studenti iscritti al **primo anno** delle lauree area Ing. Dell'Informazione
    - il cui numero di matricola **termina con le cifre 4 o 5**
  - Gli studenti iscritti al **secondo o terzo anno**
    - il cui numero di matricola **termina con le cifre 4 o 5**
    - che ne facciano richiesta **mediante iscrizione tramite le bacheche elettroniche**

- **NON** sono ammessi a seguire questo corso
  - Tutti gli studenti che non soddisfano i requisiti precedenti

Informazioni importanti

- Chi **non possiede** ancora un numero di matricola ed **è iscritto** a Ing. dell'Inf. e Telecomunicazioni
  - È **provvisoriamente** ammesso a seguire questo corso
  - **Confluirà** nel corso di appartenenza non appena avrà un numero di matricola
- Chi possiede un numero di matricola **la cui penultima cifra è 0,1,2** ed è **iscritto** ad Ingegneria Informatica
  - Deve seguire il corso del Prof. Dalpasso (canale A, in teledidattica)

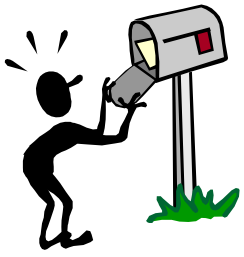
Docente



- Federico Avanzini
- DEI - Dip. di Ingegneria dell'Informazione
  - **Via Gradenigo 6/A** (**Padova**), edificio principale (DEI/G), quarto piano, ufficio 405a
- Ricevimento studenti
  - ***solo su appuntamento***
- Il metodo più efficace di interazione diretta con il docente è la ***posta elettronica***

  avanzini@dei.unipd.it
- Telefono: 049 827 7942

Attenzione
I recapiti cambieranno presto
Sono in fase di trasloco...



Sito Web del Corso

- Tutte le informazioni relative al Corso si trovano nel sito

  http://www.dei.unipd.it/~avanzini/fi1

- Sul sito si troverà anche, settimana per settimana, tutto il materiale didattico
  - copia di tutte le presentazioni
  - esercitazioni proposte per il laboratorio
  - prove di auto-valutazione

Altri Siti

- Tutte le informazioni relative alla **Aula Didattica Taliercio** si trovano nel sito

  http://www.adt.unipd.it/

- Informazioni ufficiali da parte del docente (appelli, iscrizioni, avvisi urgenti, ecc.) si trovano sulle Bacheche elettroniche

  http://sis.dei.unipd.it

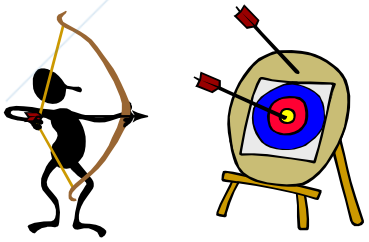
- Altre informazioni si trovano nel sito della **Facoltà di Ingegneria**

  http://www.ing.unipd.it/

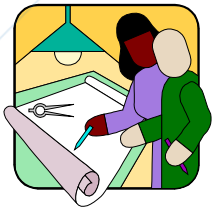
Frequenza alle lezioni

- Il regolamento didattico dice che la frequenza alle lezioni è obbligatoria
  - Lascia però libero ogni docente di decidere cosa fare con gli studenti che non frequentano
    - Si può anche impedire loro di fare l'esame
- Per questo corso, il docente ha deciso che non vengono presi provvedimenti nei confronti degli studenti che non frequentano
  - La frequenza alle lezioni e alle esercitazioni in laboratorio è **VIVAMENTE CONSIGLIATA**
  - Da un'analisi statistica, **chi non frequenta fa molta più fatica a superare l'esame**

Obiettivi del Corso



- Il Corso ha l'obiettivo di
  - presentare le **basi teoriche** dell'informatica
  - proporre un **approccio ingegneristico** e progettuale alla programmazione
  - fornire una **visione professionale** alla risoluzione dei problemi
  - **utilizzare concretamente** le caratteristiche di programmazione del linguaggio Java

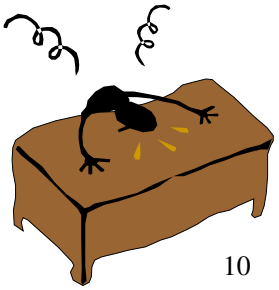


Programma del Corso (1)

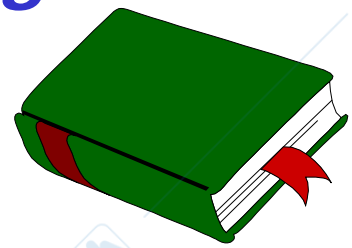
- Organizzazione di un elaboratore. CPU, dispositivi di memoria di massa, dispositivi di IO. Il sistema operativo
- Rappresentazione dell'informazione, sistemi di numerazione e conversioni.
- Linguaggi di programmazione. Compilazione e interpretazione. La macchina virtuale Java.
- Il linguaggio di programmazione Java. Tipi di dati elementari e oggetti, riferimenti, operatori ed espressioni, istruzioni di controllo. Gestione degli errori. Operazioni di ingresso e uscita.
- Introduzione alla programmazione orientata agli oggetti. Classi ed interfacce. Campi e metodi di una classe. Incapsulamento, ereditarieta`, polimorfismo

Programma del Corso (2)

- Ricorsione; eliminazione della ricorsione.
- Array. Algoritmi di ricerca in un array.
- Algoritmi di ordinamento: ordinamento per selezione, inserzione, mergesort.
- Introduzione all'analisi degli algoritmi, esemplari di un problema e loro taglie. Misura della complessita`. Notazione asintotica O-grande.
- Strutture dati e algoritmi, Tipi di dati astratti, e loro realizzazione mediante interfacce e classi.
- Liste, pile e code, realizzazione mediante un array o una catena di celle.
- Dizionari, insiemi. Realizzazioni.
- **Realizzazione di semplici progetti di programmazione nei laboratori didattici**



Libro di testo e libri consigliati



- Libro di testo:
  - Cay Horstmann, “Concetti di informatica e fondamenti di Java”, Quarta Edizione, Ed. Apogeo
- Altri libri di lettura o complemento sono segnalati sulla pagina web del corso

Quanto bisogna studiare?

- Risposta ovvia
  - **MOLTO**
- Risposta sincera
  - **MOLTISSIMO**
- Risposta tecnica
  - dipende dai crediti
- Questo è un esame da **9 crediti**

Quanto bisogna studiare?

- Un credito = 25 ore di lavoro
  - totale corso FI1 = 25 × **9** = 225 ore di lavoro
  - in aula + laboratorio = 12 × 9 = 108 ore
  - esempio: studio finale prima dell'esame = 27 ore
  - **studio settimanale = 10 ore**
- Lo studio settimanale comprende anche **pratica al calcolatore** (OLTRE al laboratorio)
  - **circa 5 ore di studio e 5 ore di pratica**

Modalità d'esame

- Sono previsti cinque appelli durante l'anno accademico
  - Due appelli a dicembre/gennaio
  - Un appello a luglio
  - Due appelli a settembre

- L'esame consiste in tre prove da sostenere nello stesso appello
  - prova scritta, prova pratica, prova orale
- Ulteriori dettagli sugli esami e le date delle prove si possono consultare **sul sito del corso**

Statistiche sugli esami 2005/06

- Promossi all'esame/iscritti all'esame
  - **Appello 1**
    - aa2005-06: 45/114 = **39%**
    - aa2006-07: 52/110 = **49%**
  - **Appello 2**
    - aa2005-06: 24/65 = **37%**
    - aa2006-07: 23/64 = **37%**
  - **Appello 3**
    - aa2005-06:1/12 = **8%**
    - aa2006-07: 3/19 = **15%**
  - **Appello 4**
    - aa2005-06: 4/13 = **31%**
    - aa2006-07: 2/11 = **18%**
  - **Appello 5**
    - aa2005-06: 5/16 = **31%**
    - aa2006-07: 9/19 = **49%**

Promossi/iscritti al corso
- aa2005-06: 79/160 = **50%**
- aa2006-07: 89/165 = **54%**

Qualche consiglio (1)

- A chi non ha mai avuto a che fare con computer, programmazione, ecc.
  - Questo corso **non ha prerequisiti**, chiunque lo può seguire con profitto
  - È **fondamentale** prendere subito confidenza con i ferri del mestiere
    - **Installare ed usare** da subito l'ambiente di programmazione java
    - Svolgere con regolarità gli **esercizi proposti nei laboratori**

Qualche consiglio (2)

- A chi ha già una solida esperienza di programmazione
  - Questo corso **non ha** lo scopo di formare dei programmatori professionisti
    - Studieremo anche: ricorsione, algoritmi di ricerca ed ordinamento, analisi della complessità di un algoritmo, tipi di dati astratti
  - Attenzione quindi a non considerarvi promossi in partenza!

Qualche consiglio (3)

- A chi non è iscritto al corso di laurea in Ingegneria Informatica
  - Non considerate questo corso come **estraneo** al vostro indirizzo di studio
    - I contenuti del corso vi serviranno (si spera)
      - Nel prosieguo del vostro corso di studi
      - Per la vostra tesi di laurea
      - Nel mondo del lavoro
  - ... Inoltre chi non supera l'esame di Fondamenti di Informatica 1 **non si laurea!**

Presentazione del Laboratorio Didattico

Avviso: primo laboratorio

- Il primo laboratorio si terrà **martedì 9 ottobre**
  - Lezione a cura del dott. Roberto Valli su concetti introduttivi. I contenuti della lezione sono disponibili sul sito web dell'aula Taliercio

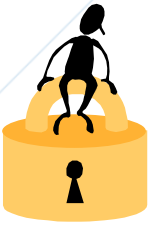
    http://www.adt.unipd.it
  - Primi semplici esercizi da svolgere. Il contenuto dell'esercitazione è disponibile sul sito del corso

    http://www.dei.unipd.it/~avanzini/fi1

Prendere confidenza con il computer

- Per molti di voi il laboratorio didattico di questo corso costituisce il primo approccio al computer
- I primi passi da compiere per acquisire confidenza con il nuovo strumento di lavoro sono:
  - accedere al computer (***fare log-in***)
  - capire i concetti di ***file*** e di cartella (***directory***)
  - scrivere un programma Java
  - individuare il compilatore e l'interprete Java
  - compilare un programma Java
  - eseguire programmi

Accedere al computer



- In un computer di uso personale spesso non è necessaria nessuna procedura di accesso
- Nei computer del laboratorio didattico l'accesso è invece sempre controllato ed è necessario eseguire la ***procedura di accesso*** (*login*)
- La procedura di accesso dipende dal sistema operativo, ma in generale richiede l'inserimento di un ***nome utente*** (*account*) e di una ***password*** (parola d'accesso), che i tecnici di laboratorio consegneranno a ciascuno studente

File

- Tutte le informazioni presenti in un computer sono memorizzate in ***file*** ("archivi") nella memoria secondaria
- Un file ha un **nome** e un **contenuto**
  - i nomi di file devono rispondere a regole che dipendono dal sistema operativo
  - di solito i nomi di file terminano con una parte, chiamata ***estensione***, che è separata da un punto
- Esempio: Un programma Java ***deve*** essere memorizzato in un file con estensione **.java**
  - ad esempio, **Test.java**

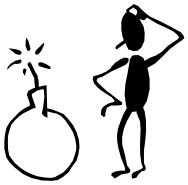
Directory

- I file sono conservati all'interno di cartelle o directory
- Una cartella può contenere
  - zero o più file e
  - zero o più altre cartelle
- È quindi possibile creare cartelle all'interno di altre cartelle, dando origine ad una ***struttura gerarchica***

Concetti introduttivi (Capitolo 1)

Legenda di alcuni simboli grafici

Errori frequenti



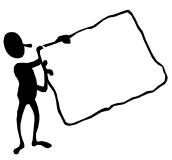
Accenni ad argomenti che verranno approfonditi in seguito

Consigli per la produttività

Regole di sintassi Java

Suggerimenti per la qualità

Note di cronaca



Argomenti avanzati

Le prime domande

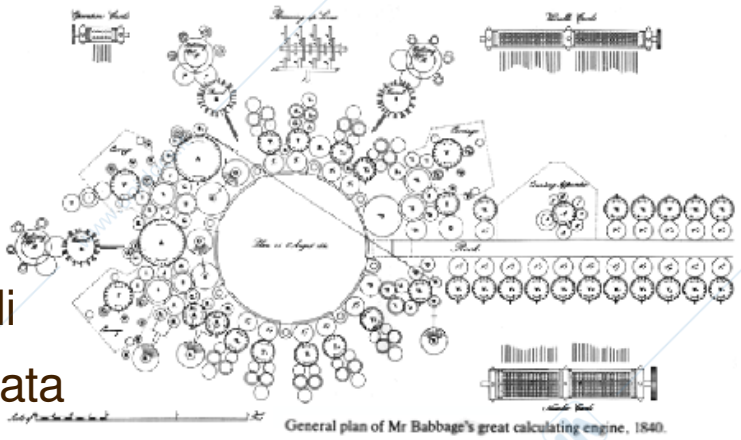
- Cos'è l'informatica?
- Cos'è un computer?
- Cos'è un programma?
- Cos'è la programmazione?
- Cos'è un algoritmo?

Un po' di storia: lo sviluppo tecnologico

- '600: dispositivi meccanici per effettuare in modo automatico computazioni (Pascal, Leibniz)
- '800: primi dispositivi meccanici "a programma": telai Jacquard, pianole, le macchine di Babbage
- 1896: Hollerith fonda la "Tabulating Machine Company" (poi IBM) che produce sistemi meccanografici a schede
- 1930: prime macchine elettromeccaniche di grandi dimensioni (Zuse in Germania, Mark 1 ad Harvard)
- 1946: ENIAC, elaboratore a valvole termoioniche e a programma filato
- 1950: prime macchine a programma memorizzato (von Neumann: ENIAC, IAS Princeton)

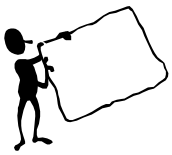
Un po' di storia: le macchine di Babbage

- Babbage (1791-1871) crea i primi computer meccanici
  - Erano dei mostri meccanici
  - Erano concettualmente molto simili ai computer moderni
- La macchina analitica (analytical engine)
  - È programmabile tramite schede perforate
  - Usa un linguaggio di programmazione equivalente a quelli attuali
  - ... non è mai stata realizzata



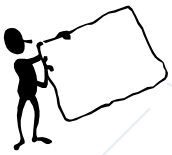
General plan of Mr Babbage's great calculating engine, 1840.

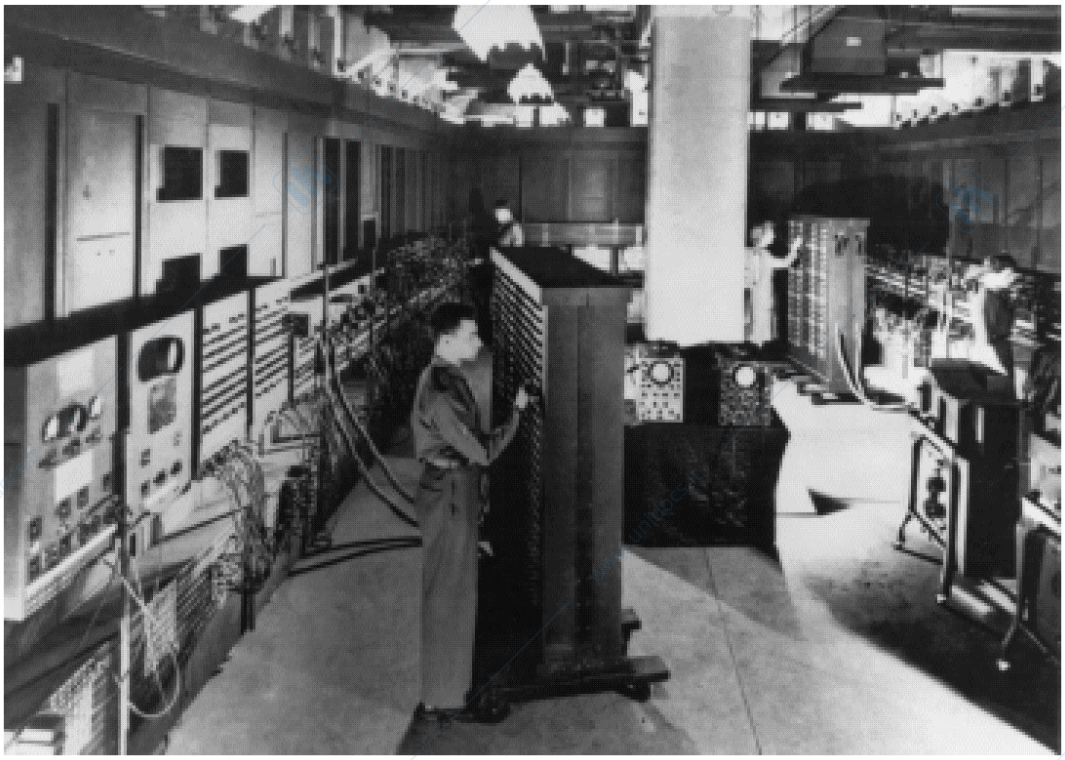
Un po' di storia: l'ENIAC



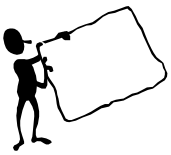
- L'ENIAC (*Electronic Numerical Integrator And Computer*) fu il primo computer elettronico, realizzato all'università di Pennsylvania nel **1946**
- Occupava una grande stanza ed era costituito da molti armadi con ***circa 18000 valvole*** (analoghe ai transistor), parecchie delle quali si bruciavano ***ogni giorno*** e dovevano essere sostituite
- Veniva ***programmato collegando cavi*** su appositi pannelli, simili a quelli dei centralini telefonici, ed andava nuovamente programmato in tal modo per ogni specifico problema

Un po' di storia: l'ENIAC



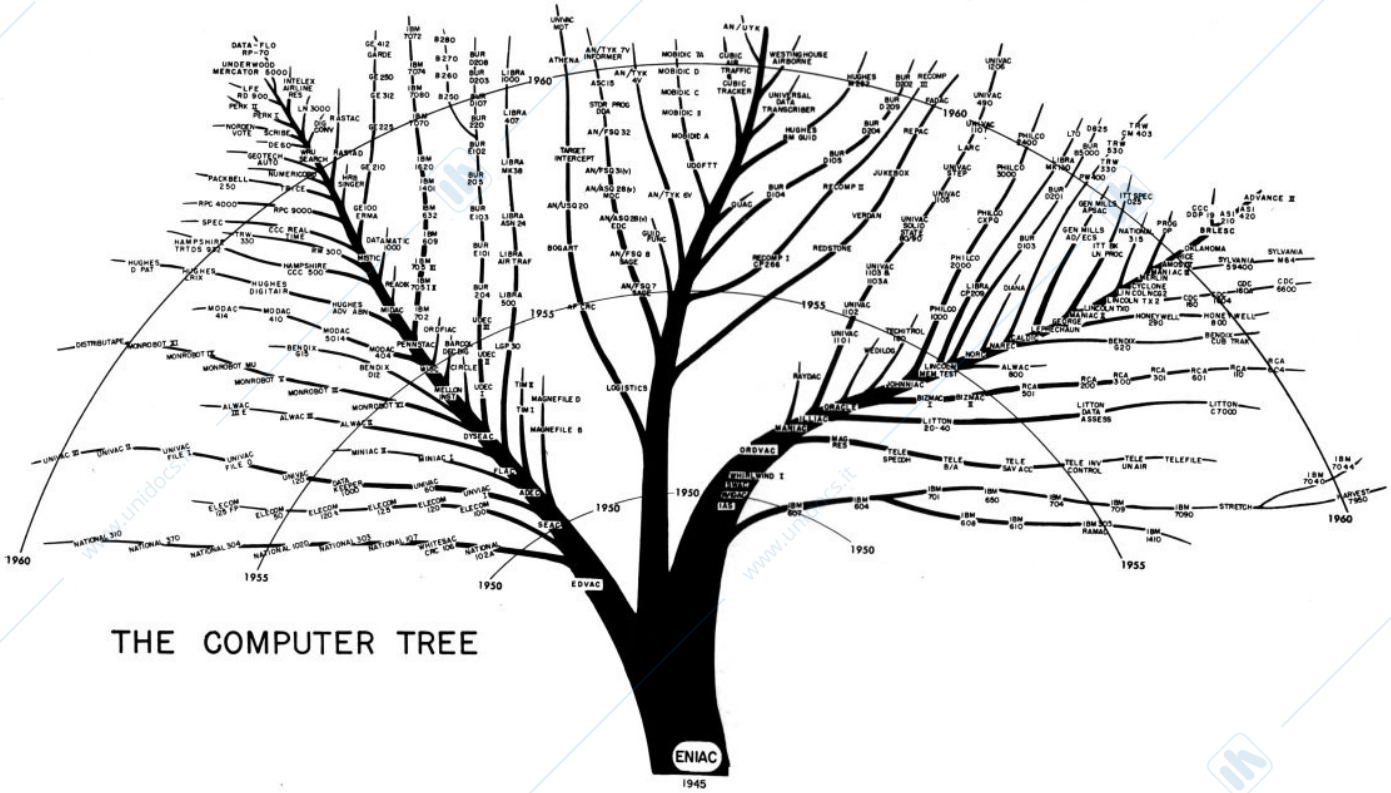


Un po' di storia: l'ENIAC



- Il progetto fu finanziato dalla Marina statunitense per il calcolo di tavole balistiche da usare per la stima della traiettoria di un proiettile in funzione della resistenza del vento, della velocità iniziale e delle condizioni atmosferiche
- Questo problema richiedeva la soluzione numerica di equazioni differenziali (***numerical integrator***)
- Questi calcoli venivano precedentemente svolti da esseri umani, detti ***computer***, letteralmente “calcolatori” (manuali...)
- Più tardi l’ENIAC venne usato per catalogare i dati del servizio demografico statunitense

Un po' di storia: lo sviluppo tecnologico



Informatica e Matematica

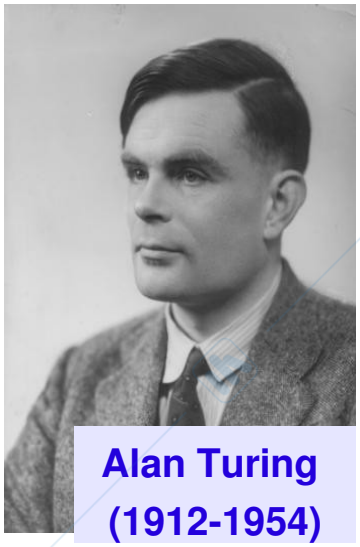
Hilbert e il formalismo

- Alla fine dell'800 la matematica si interroga sui propri fondamenti
- David Hilbert propone 23 problemi matematici fondamentali, da risolvere nel corso del XX secolo
  - Il *problema della decisione*: esiste un procedimento per decidere se una "proposizione" è o meno conseguenza logica di altre?
  - Hilbert è fiducioso: in matematica non esiste l'***Ignorabimus***!

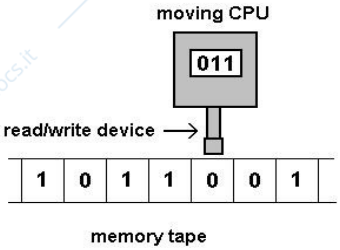
David Hilbert (1862-1943)

Turing, Church, e la teoria degli algoritmi

- Alan Turing e Alonzo Church dimostrano (~1936) che tale procedura non esiste!
  - Esistono proposizioni "indecidibili"
- Viene introdotto il concetto di *algoritmo*
  - Una sequenza di operazioni "elementari" che compongono una procedura di calcolo
- Viene inventata la macchina di Turing
  - Un computer immaginario
  - Una macchina universale, in grado di calcolare ogni funzione calcolabile



Alan Turing (1912-1954)

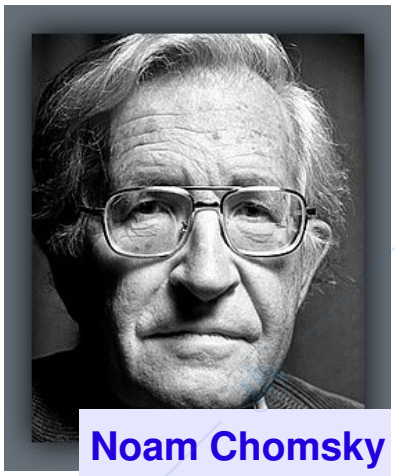


Turing, Church e la teoria della complessità

Alonzo Church
(1903-1995)

- Tra le proposizioni decidibili – o funzioni calcolabili – o problemi risolubili – alcune sono più "facili" di altre
  - Esistono diverse classi di complessità, corrispondenti al tempo necessario per risolvere i problemi
  - Alcuni problemi non sono risolubili in pratica, perchè possono essere risolti solo in tempi immensi (miliardi di anni)

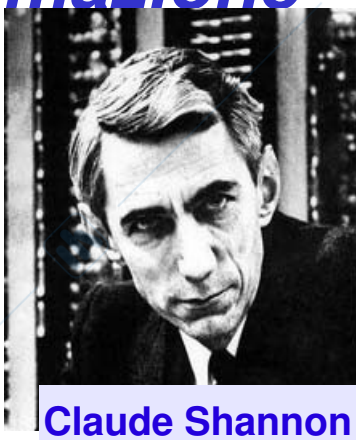
Chomsky e i linguaggi formali



Noam Chomsky
(1928)

- Nel '900 linguisti e matematici cercano di formulare un approccio strutturale allo studio del linguaggio
  - Linguaggi formali: definiti da insiemi di produzioni grammaticali che li generano
- Chomsky identifica quattro tipi di linguaggi formali, caratterizzati da diverse grammatiche
  - Universali, sensibili al contesto, liberi da contesto, regolari
  - Servono a classificare linguaggi informatici e automi

Shannon e la teoria dell'informazione



Claude Shannon
(1916-2001)

- Pone basi teoriche per la progettazione di circuiti digitali (reti logiche)
  - Costituiti da bit (binary digit), variabili che assumono solamente i valori 0 e 1
- Fonda la teoria dell'informazione
  - Problema di ricostruire in maniera affidabile informazioni trasmesse da un mittente e affette da rumore
  - Teoria alla base della costruzione di reti di calcolatori

Cos'è un computer?
Cos'è un programma?
Cos'è la programmazione?

Cos'è un computer

- Un computer è una macchina che
  - ***memorizza dati*** (numeri, parole, immagini, suoni...)
  - ***interagisce con dispositivi*** (schermo, tastiera, mouse...)
  - ***esegue programmi***

- Ogni programma svolge una diversa funzione, anche complessa
  - impaginare testi o giocare a scacchi

- I programmi sono ***sequenze di istruzioni che il computer esegue e di decisioni che il computer prende*** per svolgere una certa attività

Cos'è un computer?

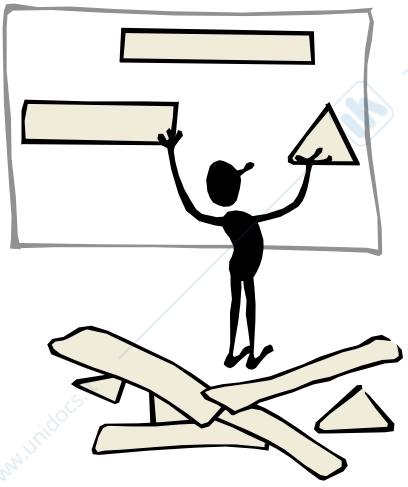
- L'***elevatissimo numero*** di tali istruzioni presenti in un programma e la loro esecuzione ad ***altissima velocità*** garantisce l'illusione di una interazione fluida che viene percepita dall'utente
- Il computer, in conclusione, è una macchina estremamente ***versatile*** e ***flessibile***, caratteristiche che gli sono conferite dai molteplici programmi che vi possono essere eseguiti, ciascuno dei quali consente di svolgere una determinata attività

Cos'è un programma?

- Nonostante i programmi siano molto sofisticati e svolgano funzioni molto complesse, le istruzioni di cui sono composti sono ***molto elementari,*** ad esempio
  - estrarre un numero da una posizione della memoria
  - sommare due numeri
  - inviare la lettera **A** alla stampante
  - accendere un punto rosso in una data posizione dello schermo
  - se un dato è negativo, proseguire il programma da una certa istruzione anziché dalla successiva (***decisione***)

Cos'è la programmazione?

- Un programma descrive al computer, in estremo dettaglio, la sequenza di passi necessari per svolgere un particolare compito
  - L'attività di ***progettare e realizzare un programma*** è detta ***programmazione***
  - **In questo corso imparerete a programmare un computer!**



Cos'è la programmazione?

- ***Usare*** un computer ***non*** richiede alcuna attività di programmazione
  - così come per guidare un automobile non è necessario essere un meccanico
- Al contrario, un ***informatico professionista*** solitamente svolge una intensa attività di programmazione, anche se la programmazione non è l'unica competenza che deve avere
- La programmazione è una parte importante dell'informatica, ed è un'attività che ***in genere*** affascina gli studenti e li motiva allo studio

Cos'è un algoritmo?

Cos'è un algoritmo?

- Quale tipo di problemi è possibile risolvere con un computer?
  - Dato un insieme di fotografie di paesaggi, qual è il paesaggio ***più rilassante***?
  - Avendo depositato ventimila euro in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo del conto arrivi al doppio della cifra iniziale?
- Il primo problema non può essere risolto dal computer. **Perché?**

Cos'è un algoritmo?

- Il primo problema non può essere risolto dal computer perché non esiste una ***definizione*** di **paesaggio rilassante** che possa essere usata per confrontare ***in modo univoco*** due paesaggi diversi
- ***Un computer può risolvere soltanto problemi che potrebbero essere risolti anche manualmente***
  - **è solo molto più veloce, non si annoia, non fa errori**
- Il secondo problema è certamente risolvibile manualmente, facendo un po' di calcoli...

Cos'è un algoritmo?

- Si dice ***algoritmo*** la ***descrizione*** di un metodo di soluzione di un problema che
  - **sia eseguibile**
  - **sia priva di ambiguità**
  - **arrivi ad una conclusione in un tempo finito**
- ***Un computer può risolvere soltanto quei problemi per i quali sia noto un algoritmo***

Un esempio di algoritmo

- **Problema**: Avendo depositato ventimila euro in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo del conto arrivi al doppio della cifra iniziale?
- **Algoritmo**:

1. Anno attuale è 0 e saldo attuale è 20000€
2. Ripetere i successivi passi 3 e 4 finché il saldo è minore di 40000€, poi passare al punto 5
3. Aggiungere 1 al valore dell'anno attuale
4. Il nuovo saldo attuale è il valore del saldo attuale moltiplicato per 1.05 (aggiungiamo il 5%)
5. Il risultato è il valore dell'anno attuale

Un esempio di algoritmo

- Il metodo di soluzione proposto
  - **è non ambiguo**, perché fornisce precise istruzioni su cosa bisogna fare ad ogni passaggio e su quale deve essere il passaggio successivo
  - **è eseguibile**, perché ciascun passaggio può essere eseguito concretamente (se, ad esempio, il metodo di soluzione dicesse che il tasso di interesse da usare al punto **4** è variabile in dipendenza da fattori economici futuri, il metodo non sarebbe eseguibile…)
  - **arriva a conclusione in un tempo finito**, perché ad ogni passo il saldo aumenta di almeno mille euro, quindi al massimo in 20 passi arriva al termine

A cosa servono gli algoritmi?

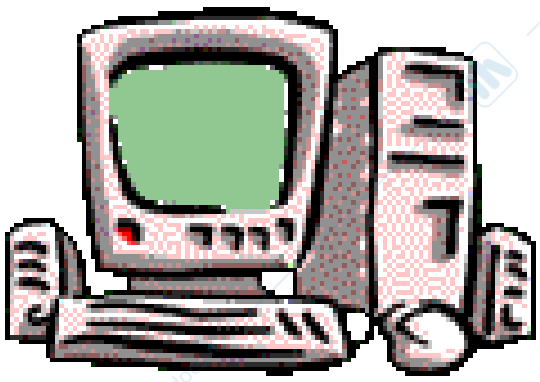
- L'identificazione di un algoritmo è un requisito indispensabile per risolvere un problema con il computer
- La scrittura di un programma per risolvere un problema con il computer consiste, in genere, nella traduzione di un algoritmo in un qualche ***linguaggio di programmazione***
- **Prima di scrivere un programma, è necessario individuare un algoritmo!**

Architettura di un computer: il modello di von Neumann

L'architettura di un computer

- Per capire i meccanismi di base della programmazione è necessario conoscere gli **elementi hardware** che costituiscono un computer

- Prendiamo in esame il ***Personal Computer*** (**PC**), ma anche i computer più potenti hanno un'**architettura** molto simile



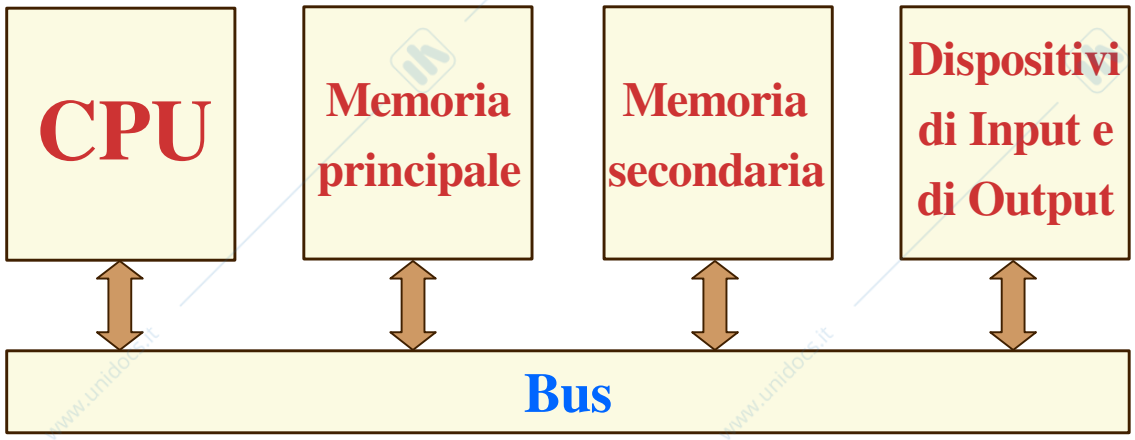
Il modello di John von Neumann

- Primo documento che descrive una macchina elettronica nella cui memoria vengono registrati dati e programma:
  - John von Neumann, *First draft of a report on the EDVAC*, Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945
- L'architettura dei moderni processori è molto simile a quella descritta nel documento, sono quindi dette ***Macchine di von Neumann***

John von Neumann (1903-1957)

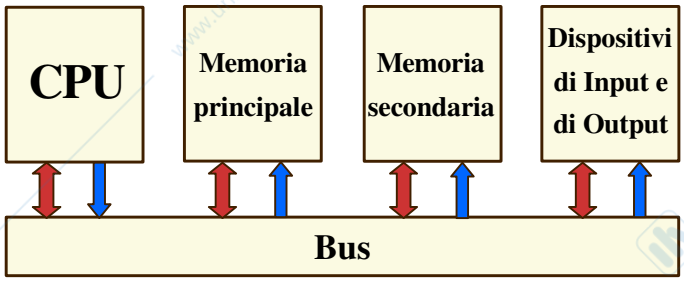
Il modello di John von Neumann

- L'architettura di von Neumann è composta da **quattro blocchi** comunicanti tra loro per mezzo di un **bus**, un canale di scambio di informazioni



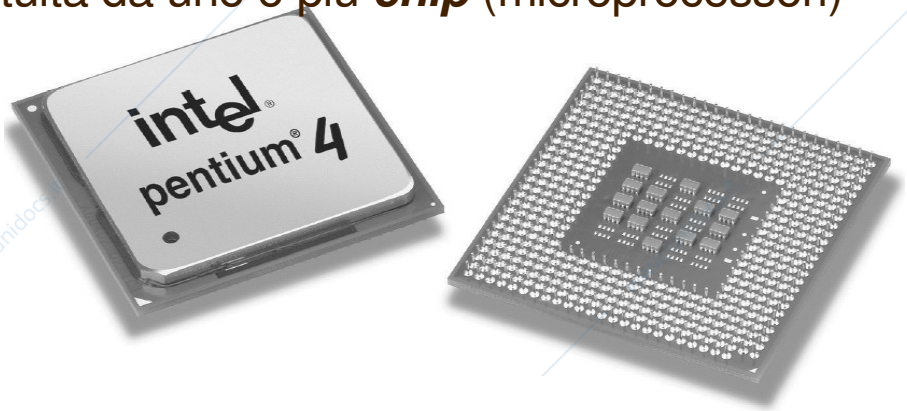
Il bus nel modello di von Neumann

- Il bus è in realtà costituito da tre bus distinti
  - bus dei ***dati***
  - bus degli ***indirizzi***
  - bus dei ***segnali di controllo***
- Sul bus dei dati viaggiano dati **da e verso la CPU**
- Sugli altri bus viaggiano indirizzi e segnali di controllo che **provengono soltanto dalla CPU**



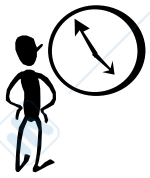
L'unità centrale di elaborazione

- L'unità centrale di elaborazione (**CPU**, *Central Processing Unit*) è il cuore del computer
  - individua ed esegue le istruzioni del programma
  - effettua elaborazioni aritmetiche e logiche con la sua unità logico-aritmetica (**ALU, *Arithmetic-Logic Unit***)
  - reperisce i dati dalla memoria esterna e da altri dispositivi periferici e ve li rispedisce dopo averli elaborati
  - è costituita da uno o più ***chip*** (microprocessori)



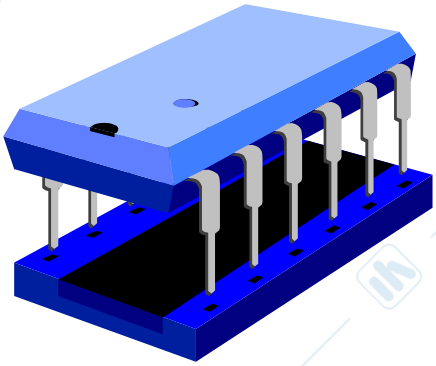
L'unità centrale di elaborazione

- Dal punto di vista logico, la CPU è costituita da tre parti principali
  - l'unità logico-aritmetica (ALU)
  - l'unità di controllo, che ne governa il funzionamento
  - un insieme di ***registri***, che sono spazi ad accesso molto veloce per la memorizzazione temporanea dei dati
- Il funzionamento della CPU è ***ciclico*** ed il periodo di tale ciclo viene scandito dall'orologio di sistema (***clock***), la cui ***frequenza*** costituisce una delle caratteristiche tecniche più importanti della CPU (es. **2 GHz**, due miliardi di cicli al secondo)



Il chip della CPU

- Un chip, o ***circuito integrato***, è un componente elettronico con connettori metallici esterni (*pin*) e collegamenti interni (*wire*), costituito principalmente di silicio e alloggiato in un contenitore plastico o ceramico (*package*)



- I collegamenti interni di un chip sono molto complicati; ad esempio, il chip Pentium4™ di Intel è costituito da circa ***50 milioni di transistor*** tra loro interconnessi

Ciclo di funzionamento della CPU

- Ogni ciclo di funzionamento è composto da tre fasi
  - **accesso**: lettura dell'istruzione da eseguire e sua memorizzazione nel ***registro istruzione***
  - **decodifica**: decodifica dell'istruzione da eseguire
  - **esecuzione**: esecuzione dell'istruzione

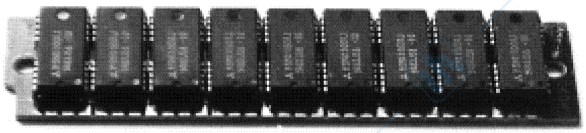
  Si parla di ciclo ***fetch-decode-execute***
- La posizione dell'istruzione a cui si accede durante la fase di *fetch* è contenuta nel ***contatore di programma*** (*program counter*, PC)
  - viene incrementato di un'unità ad ogni ciclo, in modo da ***eseguire istruzioni in sequenza***

La memoria del computer

- La memoria serve ad ***immagazzinare*** **dati** e **programmi** all'interno del computer
- È suddivisa in ***celle*** o locazioni di memoria, ognuna delle quali ha un ***indirizzo***
- Ogni cella contiene un numero predefinito di ***bit***
  - un ***bit*** è un dato elementare che può assumere due valori, *convenzionalmente* chiamati **zero** e **uno**
  - un insieme di otto bit si chiama ***byte*** ed è l'unità di misura della memoria (es. 512 MByte)
- Ci sono due tipi di memoria
  - **primaria** e **secondaria**

La memoria primaria

- La memoria ***primaria*** è **veloce** ma **costosa**
- È costituita da **chip di memoria** realizzati con la stessa tecnologia (al silicio) utilizzata per la CPU
  - memoria ***di sola lettura*** (**ROM**, *Read-Only Memory*)
  - memoria ad accesso casuale (**RAM**, *Random Access Memory*)
    - dovrebbe chiamarsi memoria ***di lettura e scrittura***, perché in realtà anche la ROM è ad accesso casuale, mentre ciò che le distingue è la possibilità di scrivervi
    - ***accesso casuale*** significa che ***il tempo per accedere ad un dato non dipende dalla sua posizione nella memoria***



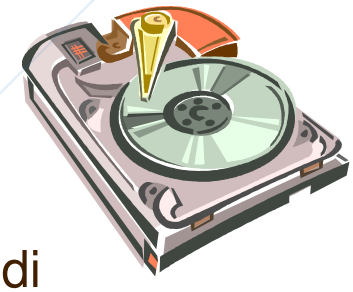
La memoria ROM

- La memoria di sola lettura, **ROM**
  - conserva i dati ed i programmi in essa memorizzati anche quando il computer viene spento
    - è una memoria ***non volatile***
  - contiene i programmi necessari all'avvio del computer, programmi che devono essere ***sempre disponibili***
    - nei PC, tali programmi prendono il nome di **BIOS** (**B**asic **I**nput/**O**utput **S**ystem)

La memoria RAM

- La memoria ad accesso casuale, **RAM**
  - è una memoria che consente la ***lettura*** e la ***scrittura*** dei dati e dei programmi in essa contenuti
  - contiene dati in fase di modifica e programmi che non devono essere sempre disponibili
  - **perde i dati quando si spegne il computer** (è un supporto ***volatile***)

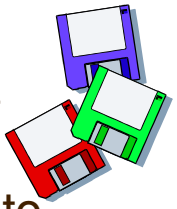
La memoria secondaria



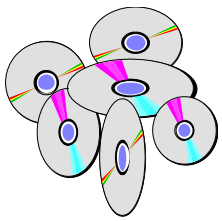
- La memoria ***secondaria*** (o *di massa*) è di solito un **disco rigido** (o disco fisso, *hard disk*) ed è un supporto ***non volatile*** e ***meno costoso*** della memoria primaria (circa cento volte)
  - programmi e dati risiedono sul disco rigido e vengono caricati nella RAM quando necessario, per poi tornarvi aggiornati se e quando necessario
- Un disco rigido è formato da piatti rotanti rivestiti di materiale magnetico, con testine di lettura/scrittura
- Processo simile a quello dei nastri audio o video

La memoria secondaria

- Sono (erano...) molto diffusi anche altri tipi di memoria secondaria a tecnologia magnetica
  - ***floppy disk*** (**dischetto** flessibile), di capacità limitata ma con il vantaggio di poter essere agevolmente rimosso dal sistema e trasferito ad un altro sistema (dispositivo di memoria ***esterno***)
  - ***tape*** (**nastri** per dati), di capacità elevatissima, molto economici, ma molto lenti, perché ***l'accesso ai dati è sequenziale*** anziché casuale (bisogna avvolgere o svolgere un nastro invece che spostare la testina di lettura sulla superficie di un disco)



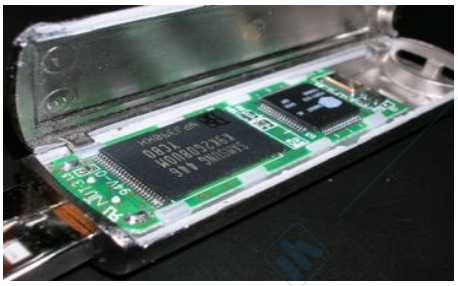
La memoria secondaria



- Sono molto diffusi anche altri tipi di memoria secondaria a tecnologia ***ottica***
  - ***CD-ROM*** (*Compact Disc Read-Only Memory*), viene letto da un dispositivo laser, esattamente come un CD audio; ha una elevata capacità ed è molto economico e affidabile; è un supporto di sola lettura, utilizzato per distribuire programmi e informazioni
  - ***CD-R*** (*Compact Disc Recordable*), utilizza una tecnologia simile al CD-ROM ma può essere scritto dall'utente (una sola volta; più volte se CD-RW)
  - ***DVD***, sta rapidamente sostituendo il CD-ROM nell'ambito di memoria a tecnologia ottica, ha capacità molto più elevata

La memoria secondaria

- La tecnologia di **memoria flash** è in rapidissima diffusione. Questo tipo di memoria è
  - permanente e riscrivibile (EEPROM), organizzato a blocchi
  - Privo di parti mobili, ed è quindi resistente a sollecitazioni ed urti, è leggero e piccolo
  - indicato per la trasportabilità (fotocamere digitali, lettori di musica portatili, cellulari, ecc.)
  - Indicato anche come possibile sostituto per hard disk di computer portatili
- Controindicazioni
  - Costi alti, anche se in forte diminuzione con il progredire della tecnologia
  - Limitato numero di scritture possibile



69

Dispositivi periferici di interazione

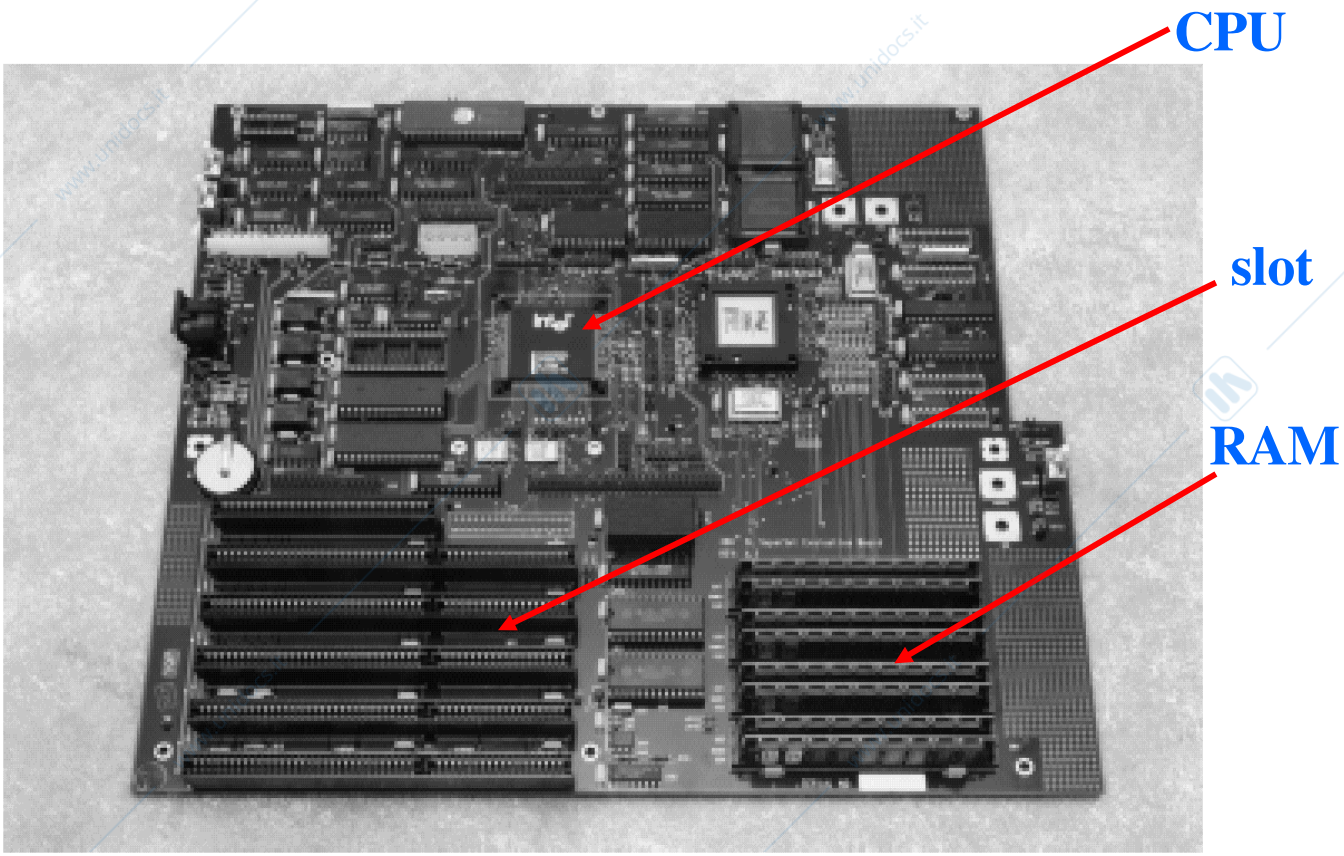
- L'interazione fra l'utente umano ed il computer avviene mediante i cosiddetti ***dispositivi periferici di Input/Output*** (dispositivi di **I/O**)
- Tipici dispositivi di ***input*** sono la **tastiera**, il **mouse** (dispositivo di puntamento), il **microfono** (per impartire comandi vocali), il **joystick** (per i giochi), lo **scanner** (per la scansione digitale di documenti e immagini)
- Tipici dispositivi di ***output*** sono lo **schermo** (*monitor*), le **stampanti**, gli **altoparlanti**

70

La scheda madre di un PC

- La CPU, la memoria primaria (RAM e ROM) e i circuiti elettronici che controllano il disco rigido e altri dispositivi periferici sono interconnessi mediante un insieme di linee elettriche che formano un ***bus***
- I dati transitano lungo il bus, dalla memoria e dai dispositivi periferici verso la CPU, e viceversa
- All'interno del PC si trova la ***scheda madre*** (***mother-board***), che contiene la CPU, la memoria primaria, il bus e gli alloggiamenti (***slot***) di espansione per il controllo delle periferiche

La scheda madre di un PC



È tutto chiaro? …

1. Dove viene memorizzato un programma che non sia attualmente in esecuzione?
2. Quale parte del computer esegue le operazioni aritmetiche, come addizioni e moltiplicazioni?

Programmazione in codice macchina

Le istruzioni macchina

- Le istruzioni elementari eseguite da un computer (cioè dalla sua CPU) si chiamano
  istruzioni macchina
- L'insieme di istruzioni macchina (***instruction set***) è **specifico** di una particolare CPU: quello di un Pentium™ è diverso da quello di uno SPARC™
- Una particolare CPU è la cosiddetta ***macchina virtuale Java*** (JVM, *Java Virtual Machine*)
  - la JVM non è una vera CPU… ma per il momento possiamo considerarla tale…

Le istruzioni macchina

- Esempio di alcune istruzioni macchina:
  - carica in un registro il valore contenuto nella posizione di memoria 40
  - carica in un altro registro il valore 100
  - se il primo valore è maggiore del secondo, prosegui con l'istruzione contenuta nella posizione di memoria 240, altrimenti con l'istruzione che segue quella attuale
- La codifica delle istruzioni macchina avviene sotto forma di ***configurazioni di bit*** conservate in memoria (che possono essere interpretate come numeri interi)
- Le precedenti istruzioni per la JVM diventano
  21 40 16 100 163 240

Le istruzioni macchina

- In tutte le CPU, le istruzioni macchina si possono suddividere nelle seguenti categorie (i nomi delle istruzioni sono solo degli esempi)
  - ***trasferimento dati***, tra i registri e la memoria principale
    - LOAD (verso un registro), STORE (verso la memoria)
  - ***operazioni aritmetiche e logiche***, eseguite dalla ALU
    - aritmetiche: ADD, SUB, MUL, DIV
    - logiche: AND, OR, NOT
  - ***salti***, per alterare il flusso di esecuzione sequenziale (viene modificato il Program Counter)
    - incondizionato (JUMP): salta in ogni caso
    - condizionato: salta solo se un certo valore è zero (JZ) o se è maggiore di zero (JGZ)

Le istruzioni macchina

- Per eseguire un programma in un computer è necessario ***scrivere all'interno della memoria primaria le configurazioni di bit corrispondenti alle istruzioni macchina del programma***
- Per fare ciò è necessario conoscere tutti i codici numerici delle istruzioni macchina
- Questa operazione lunga e noiosa (che veniva eseguita agli albori dell'informatica) è stata presto automatizzata da *un programma in esecuzione sul computer stesso*, detto ***assemblatore*** (***assembler***)

Programmazione con assembler

L'assemblatore

- Utilizzando l'assemblatore, il programmatore scrive il programma mediante dei ***nomi abbreviati*** (***codici mnemonici***) per le istruzioni macchina, molto più facili da ricordare
  - esempio precedente per la JVM

```
iload      40
bipush    100
if_icmpgt 240
```

- L'uso di nomi abbreviati è assai più agevole, ed il programma assemblatore si occupa poi di ***tradurre*** il programma in configurazioni di bit
- Tali linguaggi con codici mnemonici si dicono ***linguaggi assembly*** (uno diverso per ogni CPU)

L'assemblatore

- Un'altra caratteristica molto utile dell'assemblatore è quella di poter assegnare dei ***nomi*** agli ***indirizzi di memoria*** e ai ***valori numerici*** e di usarli nelle istruzioni

```
iload     rate
bipush    maxRate
if_icmpgt tooMuch
```

- In questo modo il programma è ***molto più leggibile***, perché viene evidenziato il ***significato*** degli indirizzi di memoria e dei valori numerici

I linguaggi assembly

- **Vantaggio**: rappresentarono un grosso passo avanti rispetto alla programmazione in linguaggio macchina
- **Problema**: occorrono ***molte istruzioni*** per eseguire anche le operazioni più semplici
- **Problema**: la sequenza di istruzioni di uno stesso programma ***cambia*** al cambiare della CPU

  ➡ è molto costoso scrivere programmi che possano funzionare su diverse CPU, perché praticamente bisogna riscriverli completamente

Linguaggi di programmazione ad alto livello

Linguaggi ad alto livello

- Negli anni '50 furono inventati i primi linguaggi di programmazione ***ad alto livello***
  - **FORTRAN**: primo "vero" linguaggio
    - **BASIC**, **COBOL**
  - Anni '60 e '70: programmazione strutturata
    - **Pascal** (Niklaus Wirth, 1968)
    - **C** (Brian Kernigham e Dennis Ritchie, 1970-75)
  - Anni '80 e '90, programmazione orientata agli oggetti
    - **C++** (Bjarne Stroustrup, 1979)
    - **Java** (James Gosling e Patrick Naughton, 1991)
- Il programmatore esprime la sequenza di operazioni da compiere, senza scendere al livello di dettaglio delle istruzioni macchina

```
if (rate > 100)
    System.out.print("Troppo");
```

Linguaggi ad alto livello

- Un programma, detto ***compilatore***, legge il programma in linguaggio ad alto livello e genera il corrispondente programma nel linguaggio macchina di una specifica CPU
- I linguaggi ad alto livello sono

  indipendenti dalla CPU

  ma il prodotto della compilazione (***codice eseguibile***), non è indipendente dalla CPU
  - occorre compilare il programma con un diverso compilatore per ogni CPU sulla quale si vuole eseguire il programma stesso

Linguaggi ad alto livello

- Esistono molti linguaggi di programmazione ad alto livello, così come esistono molte lingue
- L'esempio seguente è in linguaggio Java

```
if (rate > 100) System.out.print("Troppo");
```

e questo è l'equivalente in linguaggio Pascal

```
if rate > 100 then write ('Troppo');
```

- La sintassi di un linguaggio viene scelta dai progettisti del linguaggio stesso, come compromesso fra leggibilità, facilità di compilazione e coerenza con altri linguaggi

Linguaggi ad alto livello

- I linguaggi di programmazione, creati dall'uomo, hanno una sintassi molto più rigida di quella dei linguaggi naturali, per agevolare il compilatore
- Il compilatore segnala gli ***errori di sintassi***

```
if (rate > 100)) System.out.print("Troppo");
```

e ***non tenta di capire***, come farebbe un utente umano, perché sarebbe molto difficile verificare le ***intuizioni*** del compilatore

meglio la segnalazione di errori!

Il linguaggio di programmazione Java™

Il linguaggio Java

- Nato nel 1991 in Sun Microsystems, da un gruppo di progettisti guidato da Gosling e Naughton
- Progettato per essere ***semplice*** e ***indipendente dalla CPU*** (o, come anche si dice, dall'hardware, dalla piattaforma o dall'architettura)
- Il primo prodotto del progetto Java fu un ***browser***, **HotJava**, presentato nel 1994, che poteva scaricare programmi (detti ***applet***) da un server ed eseguirli sul computer dell'utente, ***indipendentemente*** dalla sua piattaforma

Il linguaggio Java

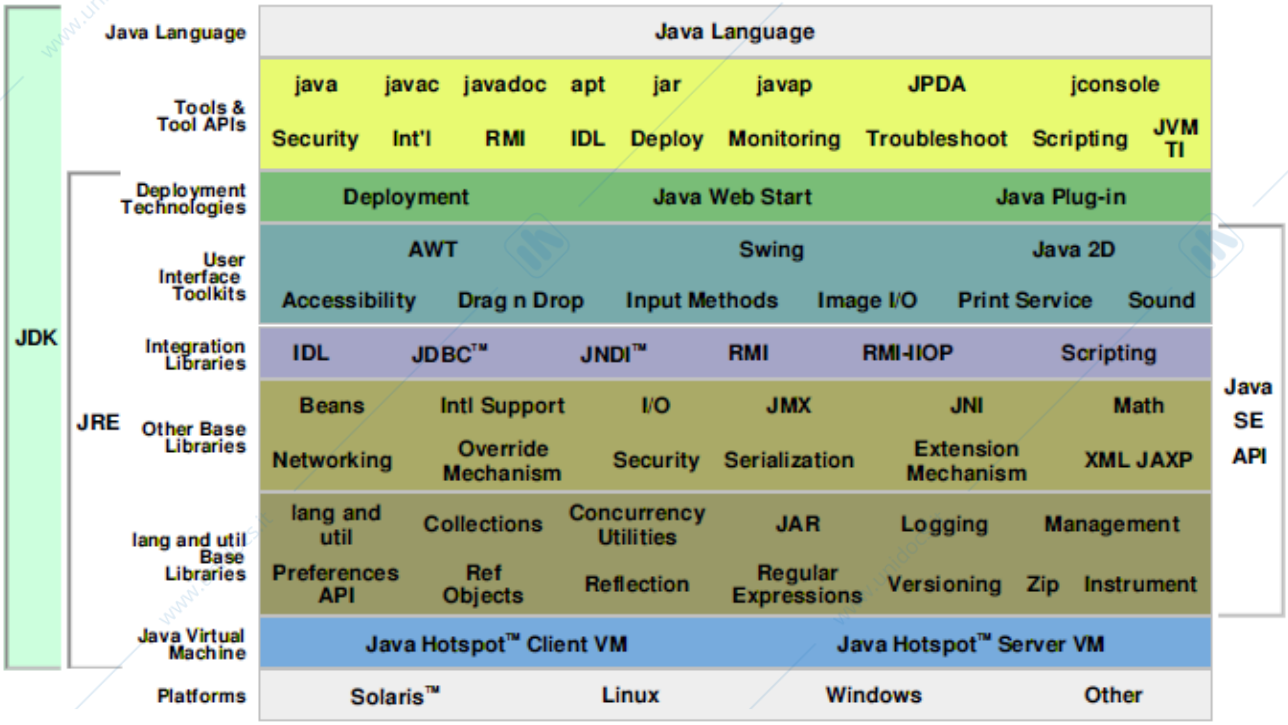
- Il linguaggio Java è stato adottato da moltissimi programmatori perché
  - è più ***semplice*** del suo diretto concorrente, C++
  - consente di ***scrivere e compilare una volta ed eseguire dovunque*** (cioè su tutte le piattaforme)
  - ha una ***ricchissima libreria*** che mette a disposizione dei programmatori un insieme vastissimo di funzionalità ***standard***, indipendenti anche dal sistema operativo

Il linguaggio Java per gli studenti

- Dato che Java non è stato progettato per la didattica
  - ***non è molto semplice scrivere programmi Java molto semplici!***
- Anche per scrivere programmi molto semplici è necessario conoscere parecchi dettagli "tecnici", un potenziale problema nelle prime lezioni
- Adotteremo lo stile didattico di ***usare alcuni costrutti sintattici senza spiegarli o approfondirli***, rimandando tali fasi al seguito

Java Development Kit (JDK)

- Da **scaricare** ed **installare** sul vostro PC!
  - Seguire le istruzioni sul sito del corso



Le fasi della programmazione

Le fasi della programmazione

- L'attività di programmazione si esegue in tre fasi
  - scrittura del programma (codice ***sorgente***)
  - compilazione del codice sorgente
    - creazione del codice ***eseguibile*** (codice macchina)
  - esecuzione del programma
- Per scrivere il codice sorgente si usa un ***editor di testo***, *salvando* (memorizzando) il codice in un file

  HelloTester.java

Individuare il compilatore Java

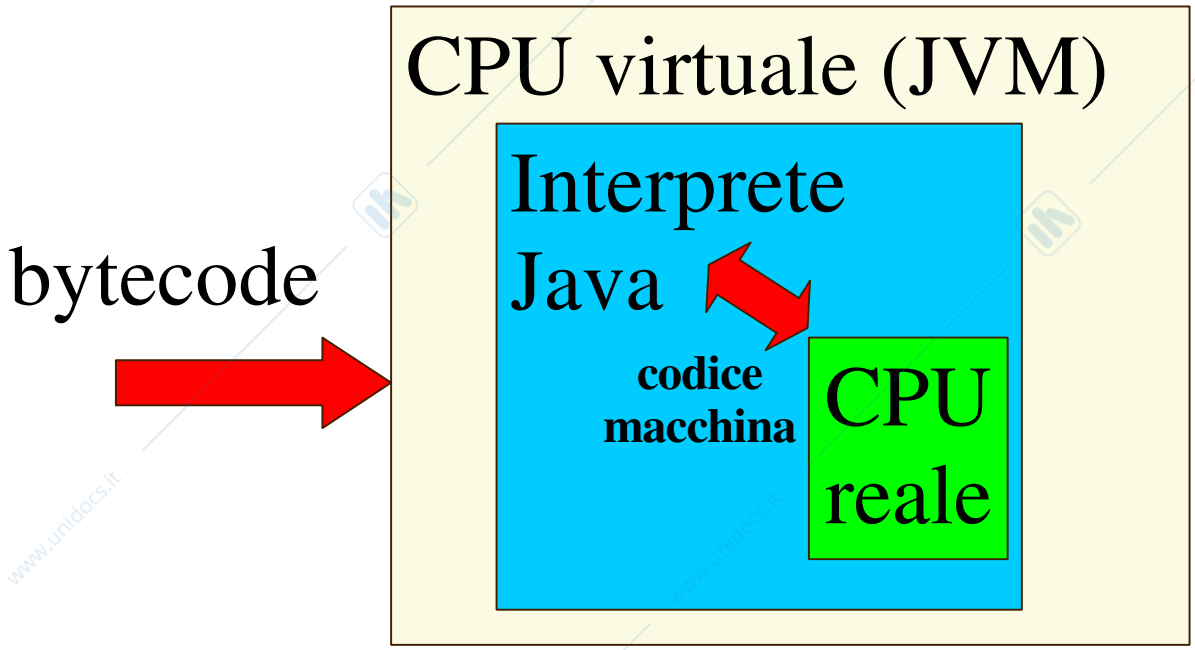
- Il modo di utilizzo del compilatore Java dipende dal sistema operativo
  - si seleziona con il mouse un'icona sullo schermo
  - si seleziona una voce in un menu di comandi
  - si compone il nome di un comando sulla tastiera
  - si utilizza un ambiente integrato per lo sviluppo software (IDE, *Integrated Development Environment*)
- Nel nostro corso useremo i comandi da tastiera del JDK (*Java Development Kit*) di Sun Microsystems

La compilazione del sorgente

- Compilando il codice sorgente di un programma (gli enunciati in linguaggio Java) si ottiene un particolare formato di ***codice eseguibile***, detto ***bytecode***, che è codice macchina per la Java Virtual Machine (JVM)

  javac HelloTester.java *genera* HelloTester.class
- Quindi il bytecode non è codice direttamente eseguibile, dato che la JVM non esiste…
- Il file con il codice bytecode contiene una ***traduzione*** delle istruzioni del programma

La Java Virtual Machine

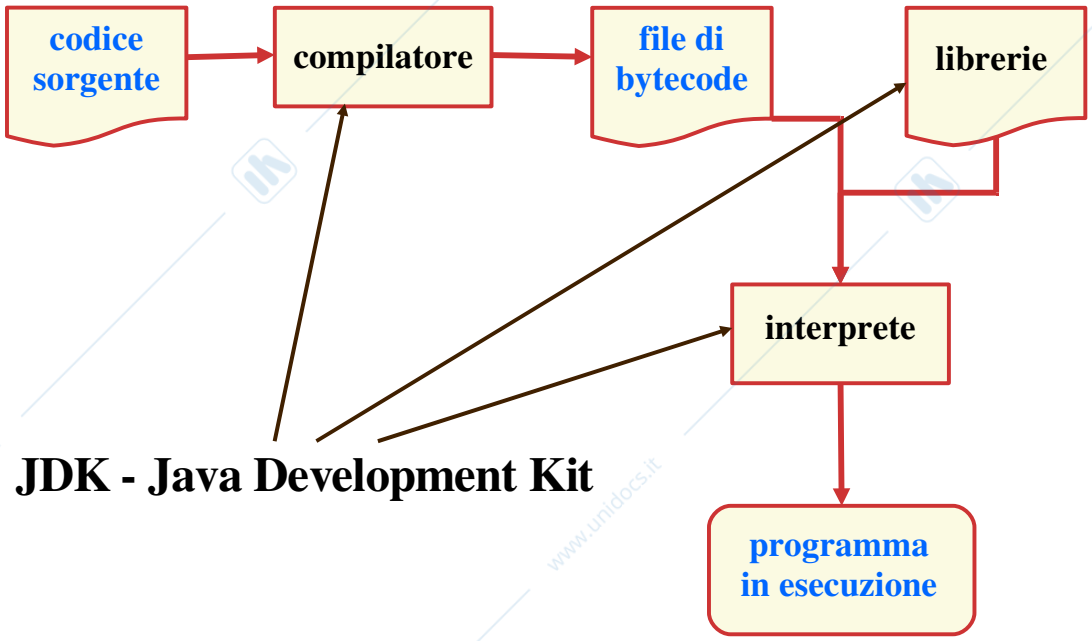


- Per ***eseguire*** un programma si usa l'***interprete*** Java, un programma eseguibile sul computer dell'utente

La libreria di classi standard

- Per scrivere sullo standard output è necessario ***interagire con il sistema operativo***, un'operazione ***di basso livello*** che richiede conoscenze specifiche
- Queste operazioni, per chi utilizza il linguaggio Java, sono state già realizzate dagli autori del linguaggio (Sun Microsystems), che hanno scritto delle classi apposite (ad esempio, **System**)
- Il bytecode di queste classi si trova all'interno di ***librerie standard***, che sono ***raccolte di classi***
- ***Non è necessario avere a disposizione il codice sorgente di queste classi, né capirlo!*** Comodo...

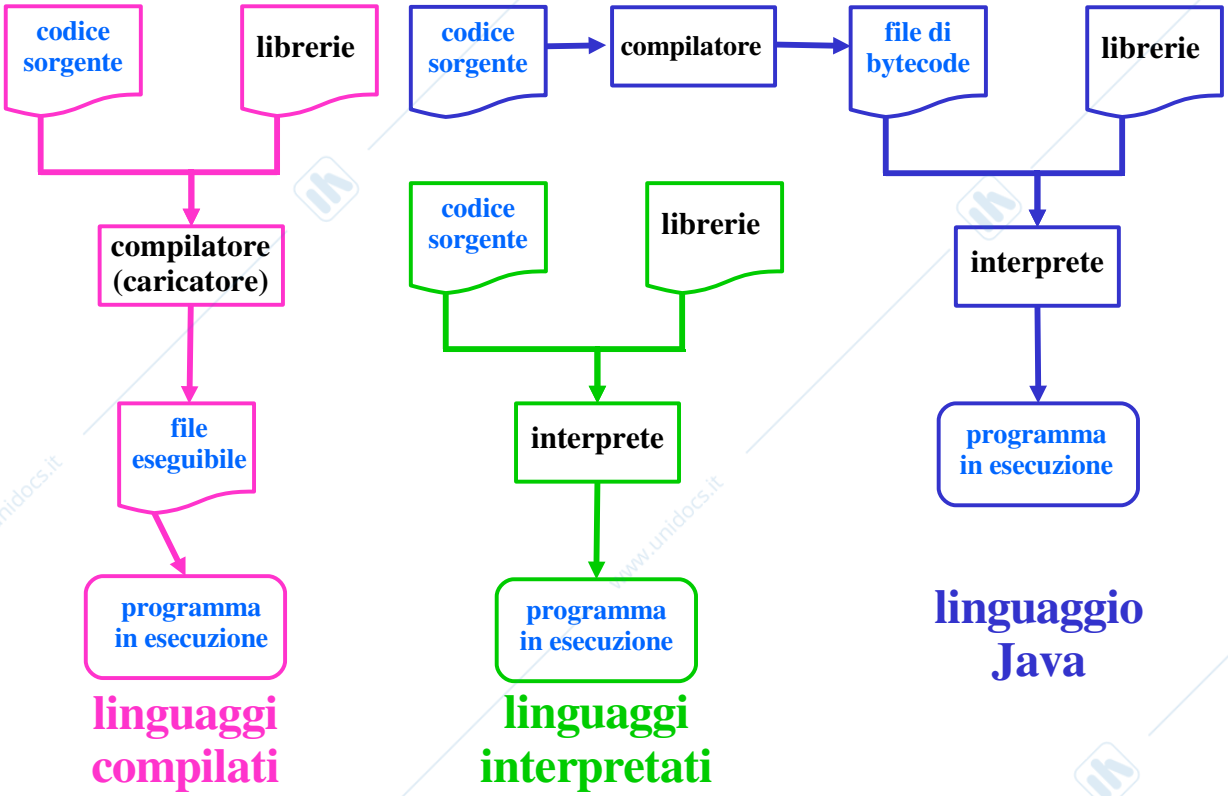
Il processo di programmazione in Java



Compilatore e/o interprete

- Per passare dalla scrittura del file sorgente in linguaggio Java all'esecuzione del programma su una particolare CPU, si usano il ***compilatore*** **e** l'***interprete***
- Con la maggior parte degli altri linguaggi di programmazione ad alto livello, invece, si usa soltanto il ***compilatore*** **oppure** soltanto l'***interprete***
  - con ***linguaggi compilati*** come Pascal, C e C++, si usa il ***compilatore*** per creare un file eseguibile, contenente codice macchina, a partire da file sorgenti (con l'eventuale ausilio di un ***caricatore***, *loader*, che interagisce con una libreria)
  - con ***linguaggi interpretati*** come BASIC e PERL, l'***interprete*** traduce "al volo" il file sorgente in codice eseguibile e lo esegue, senza creare un file eseguibile

Il processo di programmazione



Compilatore e/o interprete

- Il fatto che un linguaggio sia compilato o interpretato influisce fortemente su quanto è
  - facile eseguire lo stesso programma su computer aventi diverse CPU (***portabilità***)
  - veloce l'esecuzione di un programma (***efficienza***)
- Entrambi questi aspetti sono molto importanti nella fase di scelta di un linguaggio di programmazione da utilizzare in un progetto
- Il linguaggio Java, da questo punto di vista, è un ***linguaggio misto***, essendo sia compilato sia interpretato, in fasi diverse

Portabilità

- I programmi scritti in un linguaggio interpretato sono portabili
- I programmi scritti in un linguaggio compilato
  - sono portabili a livello di file sorgente, ma è necessario compilare il programma su ogni diversa CPU
  - non sono portabili a livello di file eseguibile, perché esso contiene codice macchina per una particolare CPU
- ***I programmi scritti in linguaggio Java sono portabili***, oltre che a livello di file sorgente, anche ***ad un livello intermedio***, il livello del bytecode
  - possono essere compilati una sola volta ed eseguiti da ***interpreti diversi*** su diverse CPU

Efficienza

- I programmi scritti in un linguaggio interpretato sono poco efficienti
  - l'intero processo di traduzione in linguaggio macchina deve essere svolto ad ogni esecuzione
- I programmi scritti in un linguaggio compilato sono molto efficienti
  - l'intero processo di traduzione in linguaggio macchina viene svolto prima dell'esecuzione, una volta per tutte
- ***I programmi scritti in linguaggio Java hanno un'efficienza intermedia***
  - parte del processo di traduzione viene svolto una volta per tutte (dal compilatore) e parte viene svolto ad ogni esecuzione (dall'interprete)

Portabilità ed efficienza

- Se si vuole soltanto la portabilità, i linguaggi interpretati sono la scelta migliore
- Se si vuole soltanto l'efficienza, i linguaggi compilati sono la scelta migliore
- Se si vogliono perseguire ***entrambi gli obiettivi***, come quasi sempre succede, ***il linguaggio Java può essere la scelta vincente***

È tutto chiaro? …

1. Quali sono i due principali vantaggi del linguaggio Java?
2. Quanto tempo occorre per imparare ad utilizzare l'intera libreria di Java?

Compilare un semplice programma

Il nostro primo programma Java

- Tradizionalmente, il primo programma che si scrive quando si impara un linguaggio di programmazione ha il compito di visualizzare sullo schermo un semplice saluto

```
Hello, World!
```

- Scriviamo il programma nel file

  HelloTester.java

  usando un editor di testi

Il nostro primo programma Java

```
public class HelloTester
{
  public static void main(String[] args)
  {
     // visualizza un messaggio sullo schermo
     System.out.println("Hello, World!");
  }
}
```

- Occorre fare molta attenzione
  - il testo va inserito esattamente come è presentato (per ora...)
  - ***maiuscole e minuscole sono considerate distinte***
  - il file ***deve*** chiamarsi **HelloTester.java**

Il nostro primo programma Java

- A questo punto ***compiliamo*** il programma

```
javac HelloTester.java
```

ed il compilatore genera il file **HelloTester.class**

- Ora ***eseguiamo*** il programma

```
java HelloTester
```

ottenendo la visualizzazione del messaggio di saluto sullo schermo

```
Hello, World!
```

Analisi del nostro primo programma in Java

Analisi del primo programma

- La prima riga

```
public class HelloTester
```

  definisce una nuova ***classe***
- Le classi sono ***fabbriche di oggetti*** e rappresentano un concetto fondamentale in Java, che è un linguaggio di programmazione ***orientato agli oggetti*** (**OOP**, ***O**bject-**O**riented **P**rogramming*)
- Per il momento, consideriamo gli ***oggetti*** come ***elementi da manipolare in un programma Java***

Analisi del primo programma

- La ***parola chiave*** **public** indica che la classe **HelloTester** può essere utilizzata da tutti (in seguito vedremo **private**…)

```
public class HelloTester
```

- Una parola chiave è una parola riservata del linguaggio che va scritta esattamente così com'è e che non può essere usata per altri scopi
- La parola chiave **class** indica che inizia la **definizione** di una ***classe***
- Ciascun ***file sorgente*** (parte di un programma Java) può contenere ***una sola classe pubblica***, il cui nome ***deve coincidere*** con il nome del file

Analisi del primo programma

- Per ora non usiamo le classi come fabbriche di oggetti, ma come ***contenitori di metodi***
- Un metodo serve a definire una sequenza di istruzioni o ***enunciati*** che ***descrive come svolgere un determinato compito*** (in altri linguaggi i metodi si chiamano *funzioni* o *procedure*)
- Un metodo ***deve*** essere inserito in una classe, quindi le classi rappresentano il meccanismo principale per l'organizzazione dei programmi

Analisi del primo programma

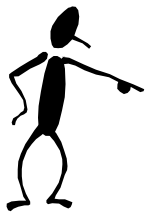
- La costruzione

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

  definisce il metodo **main** (*principale*)

- Un'applicazione Java ***deve*** avere un metodo **main**
- Anche qui, **public** significa ***utilizzabile da tutti***
- Invece, **static** significa che il metodo **main** ***non esamina e non modifica gli oggetti della classe*** **HelloTester** a cui appartiene

Programma semplice



- Sintassi:

```
public class NomeClasse
{  public static void main(String[] args)
   {  enunciati
   }
}
```

- Scopo: eseguire un programma semplice, descritto da ***enunciati*** e contenuto nel file ***NomeClasse.java***
- Nota: la parte in **blu** viene per ora considerata una ***infrastruttura necessaria***, approfondita in seguito

Analisi del primo programma

- Nel programma sono presenti anche dei ***commenti***, che vengono ***ignorati*** dal compilatore, ma che rendono il programma molto più comprensibile

```
// visualizza un...
```

- Un commento ***inizia con una doppia barra*** // e ***termina alla fine della riga***
- Nel commento si può scrivere ***qualsiasi cosa***
- Se il commento si deve estendere ***per più righe***, è molto scomodo usare tante volte la sequenza //
- Si può iniziare un commento con /* e terminarlo con */

```
// questo e' un commento
// lungo,inutile...
// ... e anche scomodo
```

```
/*
 questo e' un commento
 lungo ma inutile...
*/
```


Analisi del primo programma

- Gli enunciati del ***corpo*** di un metodo (*gli enunciati contenuti tra le parentesi graffe*) vengono eseguiti uno alla volta ***nella sequenza in cui sono scritti***

- Ogni enunciato termina con il carattere ;
- Il metodo **main** del nostro esempio ha un solo enunciato, che visualizza una riga di testo

```
System.out.println("Hello, World!");
```

- Ma ***dove*** la visualizza? Un programma può inserire testo in una finestra, scriverlo in un file o anche inviarlo ad un altro computer attraverso Internet...

Analisi del primo programma

```
System.out.println("Hello, World!");
```

- Nel nostro caso la destinazione è l'***output standard*** (o "flusso di uscita standard")
  - è una proprietà di ciascun programma che dipende dal sistema operativo del computer
- In Java, l'output standard è rappresentato da un ***oggetto*** di nome **out**
  - come ogni metodo, ***anche gli oggetti devono essere inseriti in classi***: **out** è inserito nella classe **System** della **libreria standard**, che contiene oggetti e metodi da utilizzare per accedere alle ***risorse di sistema***

  - per ***usare*** l'oggetto **out** della classe **System** si scrive

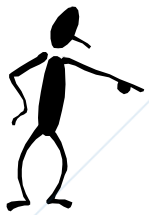
```
System.out
```


Analisi del primo programma

```
System.out.println("Hello, World!");
```

- Quando si usa un oggetto, bisogna specificare ***cosa*** si vuol fare con l'oggetto stesso
  - in questo caso vogliamo ***usare un metodo*** dell'oggetto **out**, il metodo **println**, che stampa una riga di testo
  - per ***usare*** il metodo **println** dell'oggetto **System.out** si scrive `System.out.println(parametri)`
  - la coppia di parentesi tonde racchiude le informazioni necessarie per l'esecuzione del metodo (***parametri***)
- A volte il carattere ***punto*** significa "usa un oggetto di una classe", altre volte "usa un metodo di un oggetto": dipende dal contesto...

Invocazione di metodo



- Sintassi:

```
oggetto.nomeMetodo(parametri)
```

- Scopo: invocare il metodo ***nomeMetodo*** dell'***oggetto***, fornendo ***parametri*** se sono richiesti
- Nota: se non sono richiesti parametri, le parentesi tonde devono essere indicate ugualmente
- Nota: se ci sono due o più parametri, essi vanno separati l'uno dall'altro con una ***virgola***, all'interno delle parentesi tonde

Analisi del primo programma

- Una sequenza di caratteri racchiusa tra virgolette si chiama stringa (*string*)

```
"Hello, World!"
```

- C'è un motivo per dover racchiudere le stringhe tra virgolette
  - come farebbe altrimenti il compilatore a capire che volete, ad esempio, stampare la parola **class** e non che volete iniziare la definizione di una nuova classe?
- Il metodo **println** può anche stampare numeri

```
System.out.println(7); System.out.println(3 + 4);
```

- C'è anche il metodo **print**, che funziona come **println** ma ***non va a capo al termine della stampa***

Una nota stilistica

- Diversamente da altri linguaggi (es. FORTRAN) il linguaggio Java consente una ***disposizione del testo a formato libero***: gli spazi e le interruzioni di riga ("andare a capo") non sono importanti, tranne che per separare parole
- Sarebbe quindi possibile scrivere

```
public class HelloTester{public static void
   main (String[  ] args) { System.out.
println("Hello, World!")             ;}}
```

- Bisogna però ***fare attenzione alla leggibilità!***

È tutto chiaro? …

1. Come si può modificare il programma HelloTester in modo che visualizzi le parole "Hello" e "World!" su due righe consecutive?
2. Il programma continua a funzionare anche senza la riga che inizia con // ?
3. Cosa viene visualizzato dai seguenti enunciati?

```
System.out.print("My lucky number
   is");
System.out.println(3 + 4 + 5);
```

Errori di programmazione

Errori di programmazione

```
System.out.println("Hello, World!");
```

- L'attività di programmazione, come ogni altra ***attività di progettazione***, è soggetta ad errori di vario tipo
  - **errori di sintassi o di compilazione**

```
System.aut.println("Hello, World!");
```

```
System.out.println("Hello, World!);
```

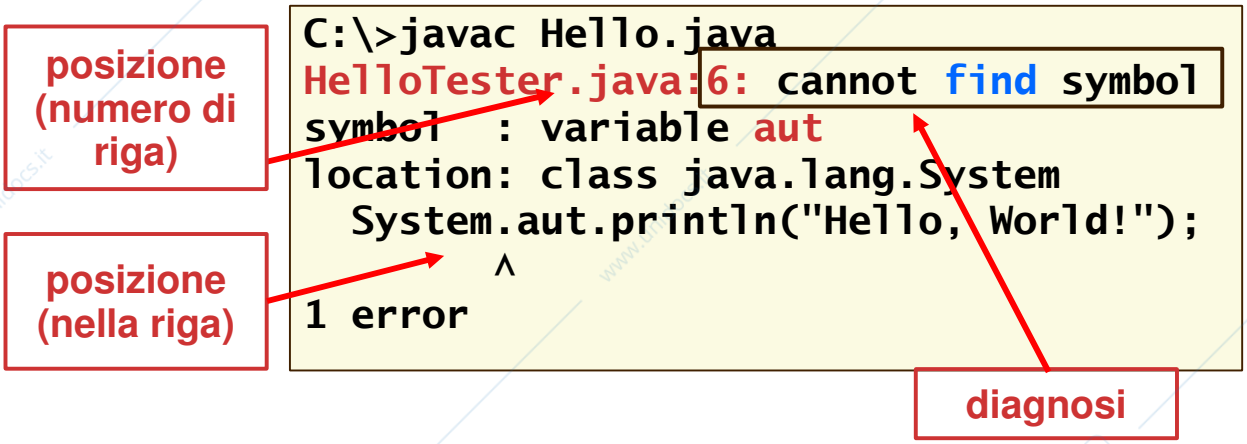
  - **errori logici o di esecuzione**

```
System.out.println("Hell, World!");
```

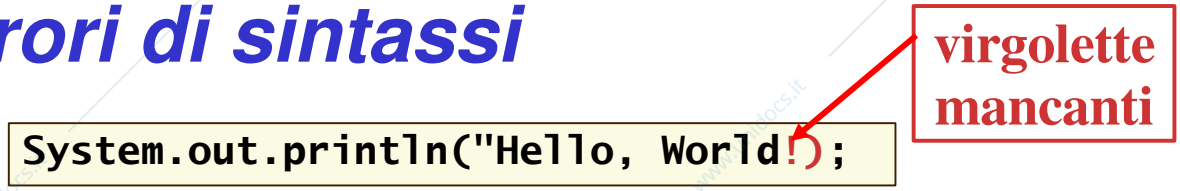
Errori di sintassi

```
System.aut.println("Hello, World!");
```

- In questo caso il compilatore riesce agevolmente ad individuare e segnalare l'errore di sintassi, perché identifica il nome di un oggetto (***simbolo***) che non è stato definito (**aut**) e sul quale non è in grado di "***decidere***"



Errori di sintassi



- Questo è invece un caso molto più complesso: viene giustamente segnalato il **primo errore**, una stringa non terminata, e viene evidenziato il punto dove ***inizia*** la stringa

```
C:\>javac Hello.java
HelloTester.java:6: unclosed string literal
  System.out.println("Hello, World!);
                     ^
HelloTester.java:7: ')' expected
  }
  ^
2 errors
```

Errori di sintassi

- Viene però segnalato anche un **secondo errore**
  - il compilatore ***si aspetta di trovare una parentesi tonda chiusa***, in corrispondenza di quella aperta
  - ***la parentesi in realtà c'è***, ma il compilatore l'ha inserita all'interno della stringa, cioè ***ha prolungato la stringa fino al termine della riga***

```
C:\>javac HelloTester.java
HelloTester.java:7: unclosed string literal
  System.out.println("Hello, World!);
                     ^
HelloTester.java:7: ')' expected
  }
  ^
2 errors
```


Errori logici

```
System.out.println("Hell, World!");
```

manca un carattere

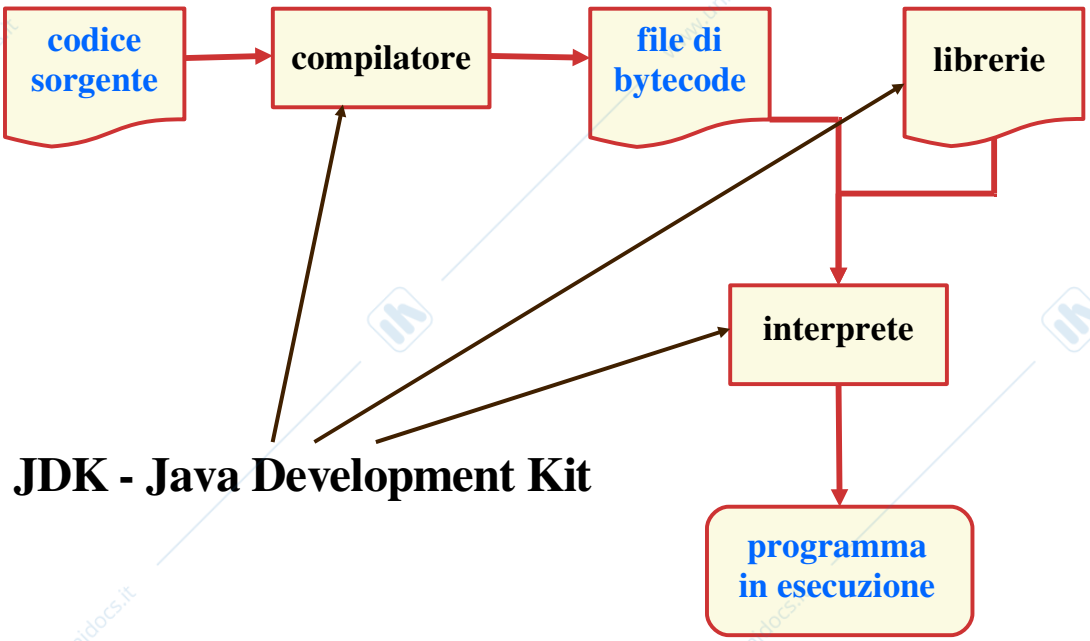
- Questo errore, invece, ***non viene segnalato dal compilatore***, che non può sapere che cosa il programmatore abbia intenzione di far scrivere al programma sull'output standard
  - ***la compilazione va a buon fine***
  - si ha un errore durante l'***esecuzione*** del programma, perché viene prodotto un output **diverso** dal previsto

```
C:\>java HelloTester
Hell, World!
```

Errori logici

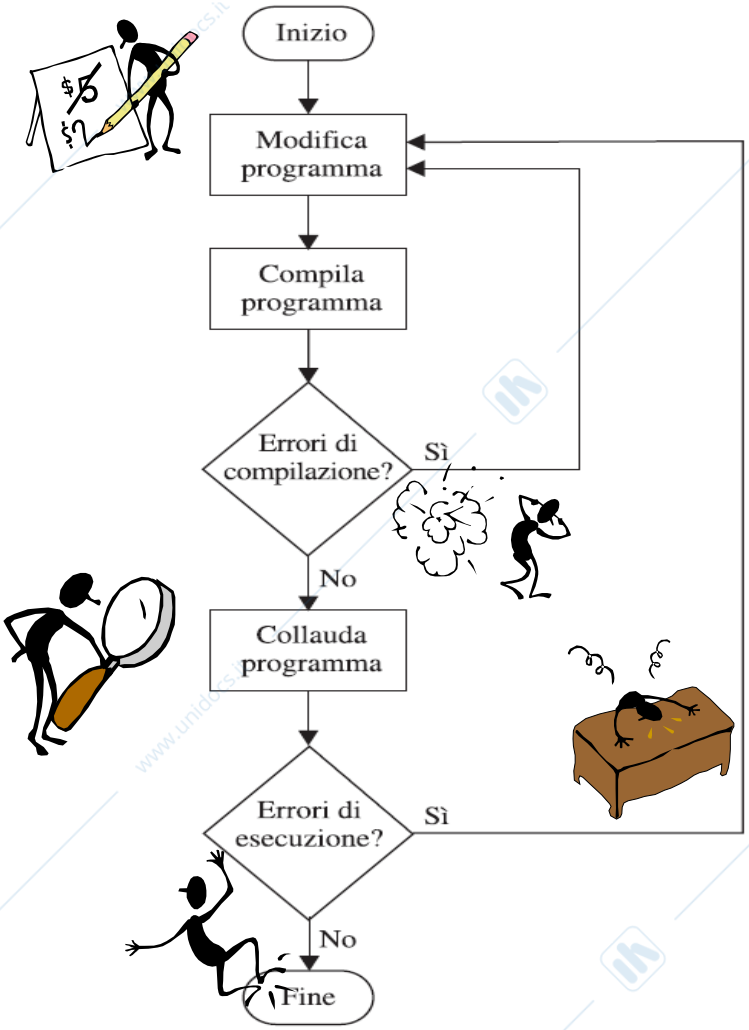
- Sono molto più insidiosi degli errori di sintassi
  - ***il programma*** viene compilato correttamente, ma ***non fa quello che dovrebbe fare***
- L'eliminazione degli errori logici richiede molta ***pazienza***, eseguendo il programma ed osservando con attenzione i risultati prodotti
  - ***è necessario collaudare i programmi, come qualsiasi altro prodotto dell'ingegneria***
- Si usano programmi specifici (***debugger)*** per trovare gli errori logici (***bug)*** in un programma
  - **noi non useremo un debugger**

Il processo di programmazione in Java



E se qualcosa non funziona?

Il ciclo Modifica-Compila-Collauda



È tutto chiaro? …

1. Se omettiamo i caratteri // dal file HelloTester.java senza eliminare la parte restante della riga otteniamo un errore di compilazione o di esecuzione?
2. Come si possono identificare gli errori logici di un programma?
3. Perché non posso collaudare un programma alla ricerca di errori di esecuzione se ci sono ancora errori di compilazione?