Il paradigma di programmazione a oggetti

Programmazione
Corso di laurea in Informatica

Paradigma a oggetti

- Formalizza mediante le classi il concetto di modulo che incapsula i dati e le procedure per modificarli
- Le classi si definiscono in una struttura gerarchica e ereditano caratteristiche e funzionalità
- Obiettivo: migliorare l'efficienza del processo di produzione e mantenimento del software

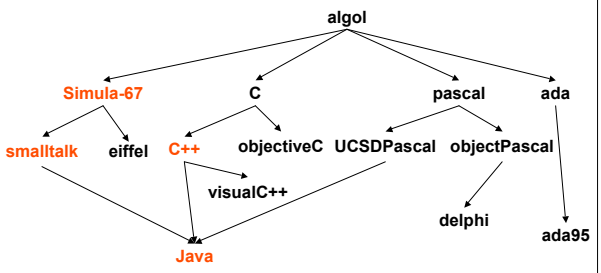
Concetti base della programmazione OO

- **Astrazione**
  - Il meccanismo con cui si specificano le caratteristiche peculiari di un oggetto che lo differenziano da altri
- **Incapsulamento dei dati**
  - Il processo con cui si nascondono i dettagli di definizione degli oggetti, solo le interfacce con l'esterno devono essere visibili
- **Ereditarietà**
  - Gli oggetti sono definiti in una gerarchia ed ereditano dall'immediato parente caratteristiche comuni, che possono essere specializzate
- **Polimorfismo**
  - Possibilità di eseguire funzioni specializzate per una particolare classe ma che hanno lo stesso nome

Java

- Java definito dalla Sun Microsystems, Inc.
- Introdotto nel 1995
- E' un linguaggio ***orientato agli oggetti***
- Derivato da Smalltalk e C++
- Definito per essere trasportabile su architetture differenti e per essere eseguito da browser

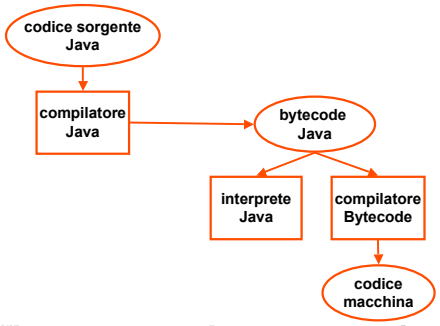
Pedigree di Java



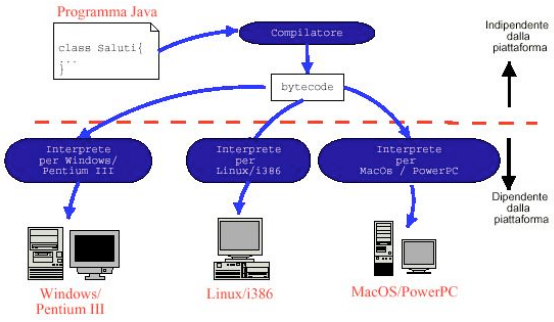
Traduzione e esecuzione di Java

- Il **compilatore** Java traduce il programma *sorgente* in una rappresentazione speciale detta ***bytecode***
- Il bytecode Java non è un linguaggio macchina di una CPU particolare, ma di una ***macchina virtuale Java***
- Il compilatore Java non è legato ad una particolare macchina
  - Java è *indipendente* dall'architettura della macchina
- L'***interprete*** traduce il bytecode nel linguaggio macchina e lo esegue

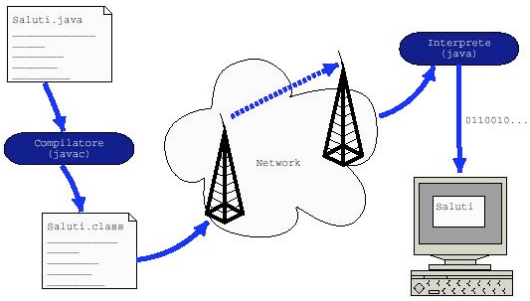
Java Virtual Machine - JVM



Portabilità del bytecode



Vantaggi del bytecode



Ambiente di sviluppo

- Ci sono diversi ambienti di sviluppo per programmi Java:
  - Sun Java Software Development Kit (SDK)
  - Borland JBuilder
  - MetroWork CodeWarrior
  - Microsoft Visual J++
  - Symantec Café
  - **Eclipse IBM** (freeware) www.eclipse.com, scritto in java
- I dettagli operativi di questi ambienti sono diversi, ma il processo di compilazione ed esecuzione è sostanzialmente identico

Struttura del programma Java

- Ogni programma Java è una raccolta di una o più ***classi***
- Una classe contiene uno o più ***metodi***
- Un metodo contiene le *istruzioni*
- Ogni programma deve avere una e una sola classe contenente il metodo speciale `main`
- Manzoni.java

Struttura del programma Java - 2

```
//  commenti sulla classe
public class Mio_programma
{

}
```

Intestazione della classe

Corpo della classe

I commenti possono essere aggiunti ovunque

Struttura del programma Java – 3

```
//  commenti sulla classe
public class Mio_programma
{
   //  commenti sul metodo
   public static void main (String[] args)
   {
   }
}
```

Corpo del metodo

Intestazione del metodo

Commenti

- I commenti di un programma sono spesso chiamati *documentazione inline*
- Devono essere inclusi per documentare lo scopo e le funzionalità del programma
- Non influenzano il funzionamento
- Vengono trascurati dal compilatore
- Possono avere due forme:

```
// commenti fino alla fine della riga
/*  commento che
    può stare su più righe        */
```


Spazi vuoti

- Gli spazi, righe vuote e le tabulazioni sono chiamati *spazi bianchi*
- Gli *spazi bianchi* sono usati per separare le parole e I simboli di un programma
- Gli *spazi bianchi* vengono ignorati dal compilatore
- Un programma Java può essere formattato come si desidera con gli spazi bianchi
- La formattazione migliora la leggibilità di un programma e va usata in modo consistente
- Es `Manzoni_2.java` e `Manzoni_3.java`

Lessico di Java

- **Alfabeto**
  - L'insieme dei caratteri specificato dal formato UNICODE a 16 bit
- **Parole riservate**
  - Un numero finito di parole che sono predefinite, non possono essere ridefinite e non possono essere usate diversamente

Parole riservate

- Sono gli identificatori predefiniti nel linguaggio:

abstract	default	goto	operator	synchronized
boolean	do	if	outer	this
break	double	implements	package	throw
byte	else	import	private	throws
byvalue	extends	inner	protected	transient
case	false	instanceof	public	true
cast	final	int	rest	try
catch	finally	interface	return	var
char	float	long	short	void
class	for	native	static	volatile
const	future	new	super	while
continue	generic	null	switch	

Identificatori

- Gli ***identificatori*** sono le parole introdotte dal programmatore usando i caratteri dell'alfabeto Java
  - Per definire classi, riferimenti a oggetti, variabili etc.
- Un identificatore è costituito da una sequenza di lettere e cifre che inizia con una lettera ed è composto da lettere, cifre, il carattere underscore **_** e il segno **$**
  - Non possono quindi iniziare con una cifra
- Java è sensibile alle maiuscole, *case sensitive*
  - `Totale` e `totale` sono identificatori diversi

Separatori e operatori

- Separatori: sono caratteri che permettono di separare o raggruppare parti di codice

  `( ) [ ] { } ; , .`

  Operatori: sono simboli o sequenze di simboli che denotano alcune operazioni

  `= > < ! ~ ? :`

  `== <= >= != && || ++ --`

  `+ - * / & | % ^ << >> >>>`

  `+= -= *= /= &= |= ^= %= <<= >>= >>>=`

Costanti

- Una ***costante*** è un identificatore il cui valore non può essere modificato dopo la sua dichiarazione iniziale
- Il compilatore segnala un errore se si cerca di modificare una costante
- Si dichiara con il modificatore `final`

```
final int ALT_MIN = 69;
```

  - Rendono chiara la semantica dei valori letterali altrimenti non evidenti
  - Rendono il codice modulare, facilitandone il cambiamento
  - Prevengono errori involontari

Riassumendo

- Alcuni identificatori sono definiti da noi (come `Manzoni`)
- Altri sono stati definiti da altri programmatori e noi li usiamo (come `println`)
- Alcuni identificatori speciali sono detti ***parole riservate*** e hanno un significato prestabilito
- Una parola riservata non può essere ridefinita

Errori

- Errori di sintassi, che vengono intercettati dal compilatore (***errori di compilazione***)
  - Se c'è un errore durante la fase di compilazione, non viene creato un programma eseguibile
- Errori generati durante l'esecuzione (***errori d'esecuzione***)
  - Tentativi di divisione per zero, che causano la fine anomala del programma
- Errori che producono risultati diversi da quelli desiderati (***errori logici***)

Errori sintattici o di compilazione

- Nel programma `Manzoni.java` si sostituisca `System.out.println ("il cielo di …");` con le seguenti espresioni

- `System.aut.println ("il cielo di …");`
- `System.out.println ("il cielo di …_);`
- `System.out.println ("il cilo di …");`

Errori sematici

- Nel programma `Divisione.java` si provi a passare in input `0` come divisore
- In esecuzione si genera il messaggio:

```
java.lang.ArithmeticException: /by zero
   at Divisione.main(Divisione.java:26)
Exception in thread "main" Process
Exit...
```

Errori logici

- Si vuole calcolare il MCD tra due numeri
- Si fornisce il programma `MCD_errato.java`
  - Non si ottengono errori in compilazione,
  - né errori in esecuzione,
  - ma non si ottiene neanche il risultato voluto.

Librerie di I/O

- Per esercizi anni scorsi
  - Scaricare il file **`cs1.jar`**
  - Salvarlo nella cartella **`ext`** sotto la directory **`jdk`**
  - **`\jdk\jre\lib\ext`**
- Per esercizi del testo Pighizzini
  - Scaricare il file **`corsoAlberti.jar`**
  - E collocarlo nella stessa cartella **`ext`**