Lezione 1

giovedì 27 settembre 2018 09:53

ALESSIO VECCHIO

alessio.vecchio@unipi.it

050 2217463

http://vecchio.iet.unipi.it/vecchio

http://vecchio.iet.unipi.it/programmazione-avanzata

Esame:

- prova scritta (al calcolatore)
  - 2 esercizi
  - 2,5 ore
- progetto (cimino)
  - 45 ore
  - discussione (cimino)
- Orale
  - circa 20 minuti
- Max 4 consegne su 7 appelli

Testi di riferimento (materiale didattico)

- The Java Programming Language (4th edition)
  K. Arnold, J. Gosling, D. Holmes
- Programmare in Java, vol. I, II, III
  Lettieri, Frosini, Vecchio
- Slide/dispense cimino

Ricevimento:

Giovedì: 13:30 - 15:30
oppure su appuntamento via e-mail

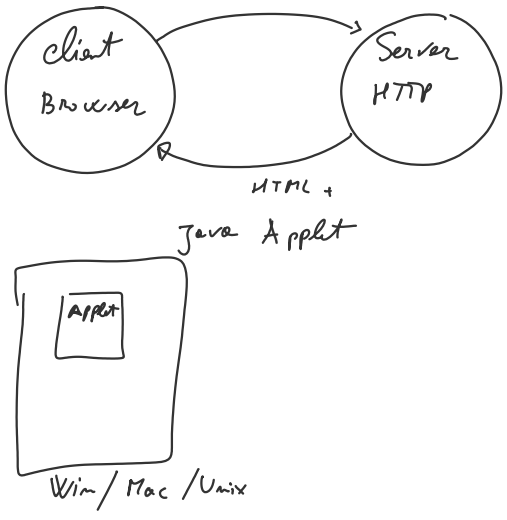
Obiettivi formativi

- esempi di prog. Object-Oriented
- " " prog. distribuita e concorrente (multi-thread)
- intera applicazione GUI, comunicazione di rete, gestione eccezioni, logica back-end
- Integrated Develop. Environment (IDE)

Linguaggio Java come strumento

Introduzione a Java

- Anni 90, Gosling, Sun Microsystems
- Oak
- Esplosione Web



Design goal:

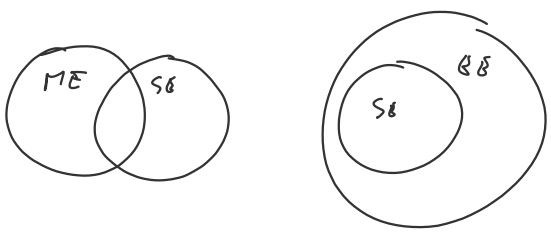
- Simple, Object Oriented, Familiar ← rimosse funzionalità "complesse" di altri linguaggi (per es. i puntatori) ← simile C/C++
- Robust, Secure ← Prog. procedurale non più possibile ← garbage collection
- Architecture neutral, Portable
- High Performance
- Interpreted, Threaded, Dynamic

- Versioni

JDK = Java Development Kit

96	JDK 1.0
97	JDK 1.1
98	J2SE 1.2
00	J2SE 1.3
02	J2SE 1.4
04	J2SE 5.0
06	Java SE 6
11	Java SE 7
14	Java SE 8
	⋮
	10

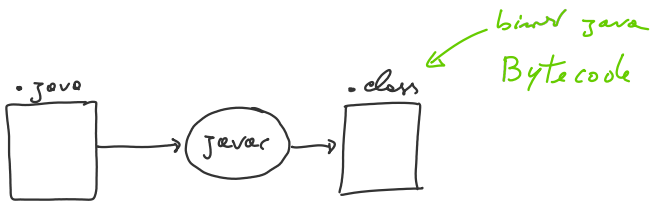
Java → SE Standard Edition
→ ME Micro Edition
→ EE Enterpr. Edition

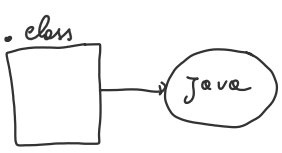


Sun acquisita da Oracle

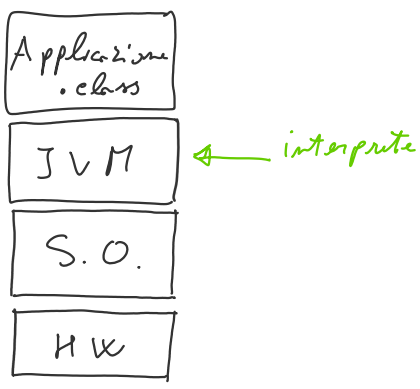
Java 10

JRE : Java Runtime Environment (libraries + interprete)



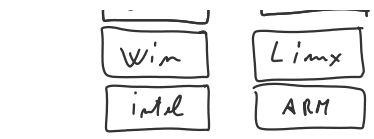


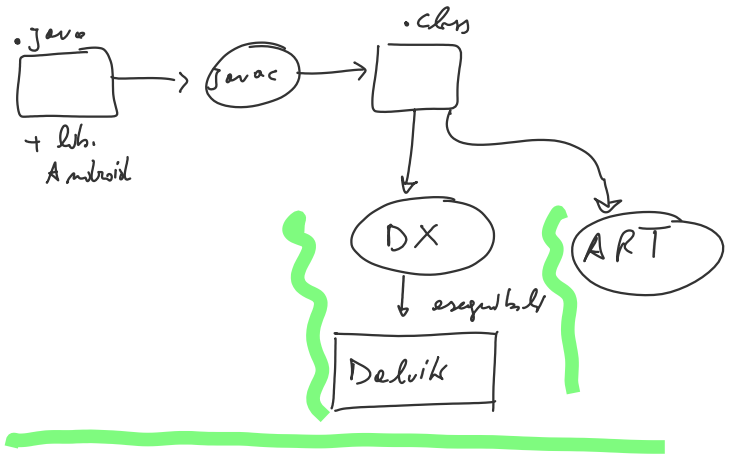
JVM = Java Virtual Machine



Macchina basata su stack

Just in Time





Java è strongly typed

Tipi
- primitivi
- riferimento

Tipi primitivi

	boolean	valori booleani, nessuna relazione con interi
tipi interi	byte	interi con segno 8 bit complemento a due
	short	" " " 16 bit " "
	int	" " " 32 bit " "
	long	" " " 64 bit " "
	char	16 bit, Unicode
tipi reali	float	reali 32 bit
	double	" 64 bit

I file sorgenti possono contenere caratteri Unicode

Unicode:
ogni carattere è identificato da un numero (codepoint)

Encoding: come rappresentare il codepoint (numero)

UTF-8 i caratteri dell'antico ASCII sono su 1 byte
altri su 2 byte
altri su 3 byte

UTF-16 tutti i caratteri del Basic Multilingual Plane (BMP) sono codificati su 16 bit, gli altri 32 bit

UTF-32 ogni codepoint è codificato usando esattamente 32 bit
vantaggio: 1 carattere -> sempre 4 byte
svantaggio: occupazione di memoria

Commenti

```
//  commento fino a fine riga

/*
    commento su più righe
 */

/**
    Commento javadoc
    su più righe
 */
```

Strumento Javadoc

```
/**
    Il metodo ...
    @param x è un intero ...
```

```
void m(int x) {
    =
}
```

HelloWorld

Tutto il codice Java è racchiuso dentro classi

Nome della classe *

```
class HelloWorld {

    public static void main(String[] args){

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

* per convenzione i nomi di classe cominciano per lettera maiuscola (e se composti anche le parole successive alla prima iniziano per maiuscola).

public: è un metodo visibile dall'esterno (fuori classe)

static: è un metodo della classe (e non delle sue istanze)

void: non restituisce un valore

main: nome del metodo che costituisce il punto di ingresso per l'esecuzione

String[] args: args è un array di stringhe

System.out: out è un membro statico della classe System (uscita standard)

println: metodo invocato sull'oggetto out stampa a video + new line

Salvare in HelloWorld.java

Ogni classe in un file separato

$ javac HelloWorld.java

ottenimo

HelloWorld.class

$ java HelloWorld ← no estensione

↑ Nome della classe

PA18_19 - HelloWorld

Definizione di variabili

```
int x;
boolean b;
float y;
   :
double d1 = 5.3;
int x1, x2;
double d2 = 5.3 + 1.1;
double d3 = d1 + 10.5;
double d4 = Math.sqrt(3.0);
   :
```

Costanti

parola chiave final

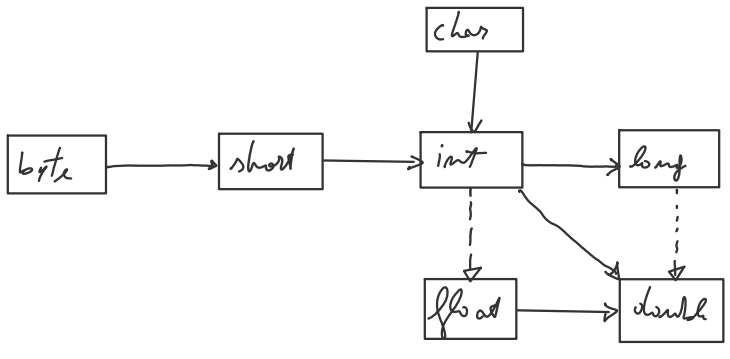
```
final int z = 10;
z = 12;        // errore

final int y;
y = 10;        // OK
y = 11;        // errore
```

Negli interi la divisione per zero genera ArithmeticException

Reali 1.0/0.0 risultato è $+\infty$ Double.POSITIVE_INFINITY ← Classe Wrapper

0.0/0.0 " è Double.NaN

Conversioni di tipo



——— conversione senza perdita di precisione
- - - - " con possibile perdita di precisione

Cast : (tipo a cui convertire)

Istruzioni condizionali

espr. booleana (No interi)

if (condizione) if (cond)

———

else

———

switch come c/c++

istruzioni ripetitive

while : come c/c++ (condizione booleana)
do-while : " " "
for : " " "

enhanced for

for (type loop_variable : set_expression)
Corpo

set_expression: definisce l'insieme di valori da scorrere

loop_variable: assume tutti i valori del set

type: tipo coerente con il tipo degli elementi del set

"seq - expression" deve essere

- un array
- un oggetto che implementa l'interfaccia java.lang.Iterable

```java
class Main {
  public static void main(String[] args) {
    int[] ar = new int[] {1, 2, 3};
    double r = media(ar);
    System.out.println(r);
  }

  static double media(int[] v) {
    if(v == null || v.length == 0)
      throw new IllegalArgumentException();
    double somma = 0.0;
    for(int valore: v) {
      somma += valore;
    }
    return somma / v.length;
  }
}
```

esempio for each

Equivalente a

```
for (int i=0;i<v.length;i++){
    int valore = V[i];
    somma += valore;
}
```

ImpishAnotherScientists

Può essere usato solo per "leggere"

Ingresso/Uscita semplificato

A ogni processo Java sono associati tre stream

- ingresso — System.in — tastiera
- uscita standard — System.out — video
- uscita errore — System.err — video

Possibile redirezione secondo le regole del sistema operativo in cui l'applicazione è eseguita

```
$ java App > out.txt
```

Per leggere dallo stream di ingresso dobbiamo usare la classe Scanner

```
Scanner sc = new Scanner(System.in);
```

l'oggetto scanner sc mette a disposizione dei metodi utili a leggere int, real, etc.

boolean	nextBoolean()	
float	nextFloat()	
double	nextDouble()	
int	nextInt()	
String	nextLine()	intera linea
String	next()	una parola

↑ Per la tastiera sono bloccanti

Per poter usare la classe Scanner all'inizio del file sorgente dobbiamo scrivere

```
import java.util.Scanner;
```

Per stampare a video

```
System.out.println( )
```

↑ Ci sono varie versioni del metodo (prendono in ingresso valori di tipo diverso).

Classi

In Java l'unità fondamentale di programmazione è la classe

- tutto il codice è dentro classi

Classi definiscono

- struttura degli oggetti* della classe
- codice (metodi) che lavora sugli oggetti della classe
- meccanismi per la costruzione degli oggetti (costruttori)

* o della classe = istanze della classe

Esempio

```
class Punto {
    double x;
    double y;
    Punto ( double a, double b){
        x = a;
        y = b;
    }
    double distanza () {
        return Math. sqrt (x*x + y*y);
    }
    void trasla ( double t){
        x += t;
        y += t;
    }
}
```

variabili istanza o var. membro (definiscono la struttura degli oggetti di tipo Punto)

costruttore

metodi

Punto p1; È un riferimento

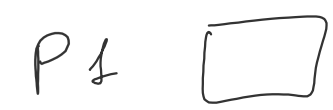
Non ci sono per ora oggetti il tipo Punto

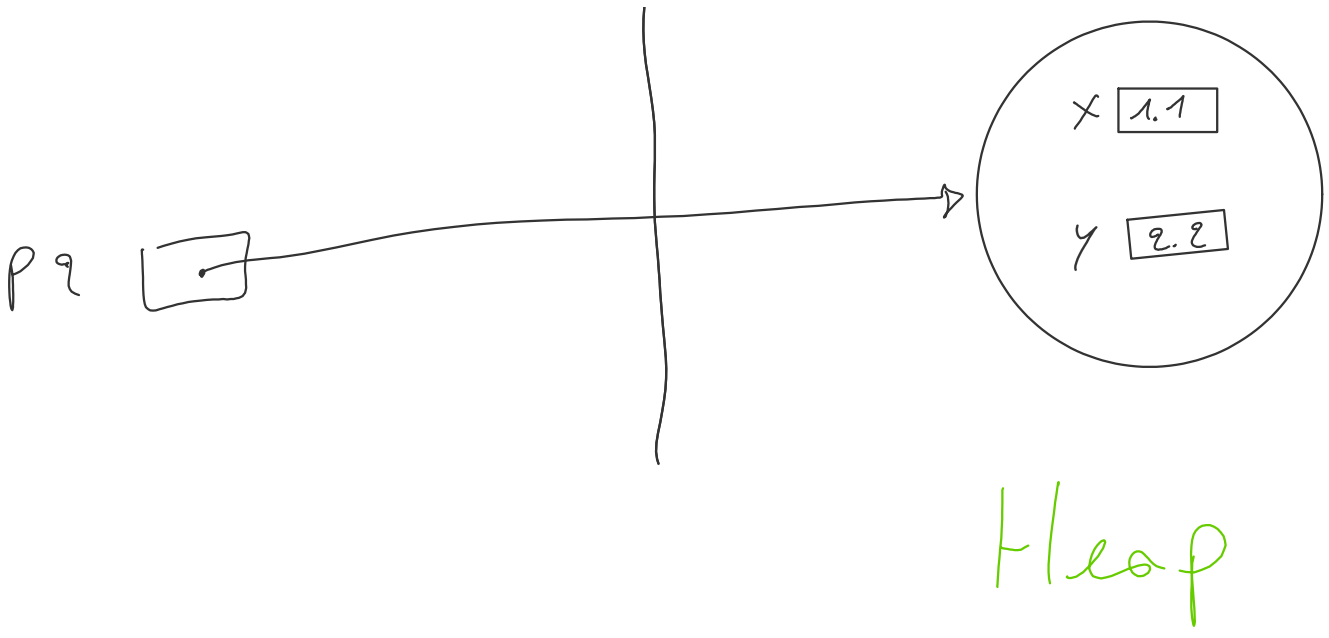
Punto p2 = new Punto (1.1, 2.2);

Riferimento

oggetto vero e proprio

Tutti gli oggetti sono allocati nello heap




p q
x 1.1
y 2.2
Heap