Introduzione a Java

Metodologie di Programmazione A.A. 2008-09

Stefano Guerrini

Corso di Laurea in Informatica
Dipartimento di Informatica.
Sapienza Università di Roma
guerrini@di.uniroma1.it

Febbraio 2009

Hello, world! in Java

Il mio primo programma Java: stampa `Hello, world!`

```
1  /**
2   * Hello-world: versione senza oggetti.
3   *
4   * @author Stefano Guerrini
5   * @version 26/02/2009
6   */
7  public class Hello0
8  {
9      // variabile di classe
10     static private String s = "Hello, world";
11
12     /**
13      * Il main (metodo di classe)
14      */
15     public static void main(String [] args)
16     {
17         // stampa la stringa
18         System.out.println(s + "!");
19     }
20 }
```

Come compilarlo:

```
> javac Hello0.java
```

come eseguirlo:

```
> java Hello0
Hello, world!
```

Oppure, usando un IDE (Integrated Development Environment):

- BlueJ (didattico)
- NetBeans, Eclipse, JBuilder, KDevelop, Xcode... (professionali)

Hello, world! in C

Come l'ho (o lo avrei) scritto in C?

```c
/**
 * Hello-world
 *
 * @author Stefano Guerrini
 * @version 26/02/2009
 */

#include <stdlib.h>

/* Variabile con la stringa da stampare */
char *s = "Hello, world!";

int main(int argn, char *argv[])
{
    /* stampa la stringa */
    puts(s);
}
```

Come compilarlo:

```
> gcc hello.c -o hello
```

come eseguirlo:

```
> ./hello
Hello, world!
```

Oppure, usando un IDE (Integrated Development Environment):

- NetBeans, Eclipse, KDevelop, Xcode, Dev-C++... (professionali)

Il main (1)

- In entrambi i casi viene eseguito il codice di una funzione di nome main.

  Java:
  ```
  15  public static void main(String [] args)
  ```

  C:
  ```
  13  int main(int argn, char *argv[])
  ```

- Il main del programma C non è racchiuso in nessun blocco, quello del programma Java è racchiuso all'interno di una definizione di classe:

  ```
   7  public class Hello0
   8  {
          :
  20  }
  ```

  In Java, le funzioni sono dette metodi e vanno definite in una classe.

- Le classi e gli oggetti sono concetti nuovi di Java e della programmazione orientata agli oggetti (per brevità OO) rispetto al C.

Il main (2)

- Come in C, in Java può esserci solo un metodo di nome main. In Java questo metodo riceve come input un array di stringhe (tipo String []) e non restituisce nulla (tipo **void**).
  NB. Essendo ammesso l'overloading, possono comunque esserci altri metodi di nome main che però ricevono o restituiscono parametri di tipo o in numero diversi.
- Il main deve essere dichiarato **public static** .
  - **public** indica che il metodo è visibile da tutti;
  - **static** indica che il metodo è associato alla classe e non ai suoi oggetti (ne capiremo il significato dopo aver introdotto gli oggetti).
- Anche la classe che contiene il main deve essere public.
- In un file può esserci solo una classe public e il suo nome deve coincidere con quello del file.
- Per convenzione, i nomi delle classi iniziano con una lettere maiuscola, mentre tutti gli altri nomi iniziano con una lettera minuscola.

String

- La classe Hello0 contiene anche la definizione della variabile con la stringa da stampare (questa variabile è globale nel programma C)

```
10      static private String s = "Hello, world";
```

  La definizione della stringa è preceduta da **static private**.
  - **private** indica che la variabile è visibile solo all'interno della classe dove è definita;
  - **static** che si tratta di una variabile di classe, comune a tutti gli oggetti della classe, che esiste indipendente dall'esistenza degli oggetti della classe stessa (più chiaro in seguito, quando introdurremo gli oggetti).
- Notare che in Java abbiamo un tipo String per rappresentare le stringhe.
  - String è in realtà una classe.
  - Le stringhe sono rappresentate mediante oggetti di questa classe.
  - L'assegnazione alla variabile della stringa "Hello, world"
    - crea un nuovo oggetto della classe String contenente questa stringa;
    - associa alla variabile un riferimento (un puntatore) a questo oggetto.

Librerie

La funzione puts che stampa una stringa fa parte della libreria standard del C.
Per poterla utilizzare abbiamo incluso stdlib .h.
In Java, per stampare la stringa abbiamo scritto

```
18          System.out.println(s + "!");
```

Cosa significa?

- System è una classe della librerie della piattaforma Java;
- out è una variabile pubblica che contiene un oggetto che rappresenta lo standard output (in seguito vedremo la classe di questo oggetto);
- println è un metodo dell'oggetto out che riceve come parametro una stringa e la stampa sullo standard output (aggiungendo un carattere di fine linea).

Infine, si noti l'operatore di somma su stringhe: corrisponde alla concatenazione di stringhe.

La piattaforma Java

Con piattaforma Java (Java platform) si intende

- l'insieme delle librerie disponibili in ogni installazione Java;
- ovvero, l'insieme delle API (Application Programming Interface) fornite da Java.

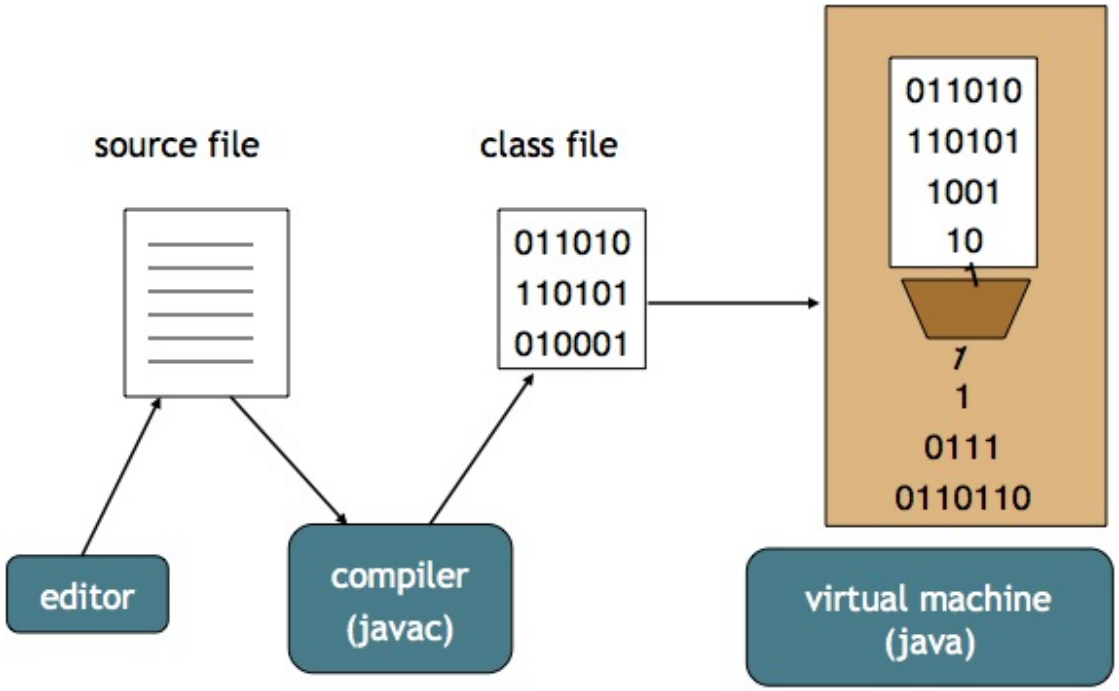
Questo insieme di librerie è

- molto esteso;
- copre gran parte delle strutture dati di base;
- fornisce funzionalità equivalenti a quelle usualmente fornite da un sistema operativo;
- è indipendente dall'architettura o dal sistema operativo.

Questo permette di realizzare programmi complessi e con funzionalità normalmente dipendenti dalla API fornite da un particolare sistema operativo senza far riferimento a nessun sistema operativo o architettura.
Un'applicazione scritta per la piattaforma Java può essere eseguita senza cambiamenti su un qualsiasi sistema operativo che supporta la piattaforma Java, ovvero sulla quale è disponibile la cosiddetta Java Virtual Machine.

Compilazione ed esecuzione: la JVM

- Il compilatore java (comando javac) compila le classi contenute nel file sorgente e per ogni classe "NomeClasse" produce un file "NomeClasse.class" contenente il codice Java Bytecode della classe.
- Java Bytecode è il linguaggio eseguito (più correttamente interpretato) dalla Java Virtual Machine o JVM.
- Il comando java esegue la Java Virtual Machine. Fornendogli come input una classe, la JVM comincia l'esecuzione chiamando il metodo main della classe, segnalando un errore nel caso in cui tale metodo non sia presente, o non sia visibile (public).



Commenti

```
1  /**
2   * Hello-world: versione senza oggetti.
3   *
4   * @author Stefano Guerrini
5   * @version 26/02/2009
6   */
```

```
12      /**
13       * Il main (metodo di classe)
14       */
```

I commenti in Java sono simili a quelli in C, anche se è ammessa una forma simile a quella del C++

```
17          // stampa la stringa
```

Notare la forma dei commenti principali: aperti da /** e chiusi da */.
Notare anche le righe contenenti i simboli speciali @author e @version.
Il contenuto di questi commenti e i valori associati a questi simboli speciali possono essere utilizzati per generare la documentazione della classe chiamando il programma javadoc.

Cominciamo ad usare gli oggetti

Una versione di "Hello, world" che utilizza un oggetto della classe Hello.

```
1  /**
2   * Class Hello:
3   *
4   * Hello-world program to demonstrate BlueJ.
5   */
6  class Hello
7  {
8      /**
9       * Method that does the work
10      */
11     public void go()
12     {
13         System.out.println("Hello, world");
14     }
15
16     /**
17      *  main method for testing outside BlueJ
18      */
19     public static void main(String[] args)
20     {
21         Hello hi = new Hello();
22         hi.go();
23     }
24 }
```

Greeter: una classe per generare saluti

```
1  /**
2     A class for producing simple greetings.
3  */
4
5  public class Greeter
6  {
7     /**
8        Constructs a Greeter object that can greet a person or
9        entity.
10       @param aName the name of the person or entity who should
11       be addressed in the greetings.
12    */
13    public Greeter(String aName)
14    {
15       name = aName;
16    }
17
18    /**
19       Greet with a "Hello" message.
20       @return a message containing "Hello" and the name of
21       the greeted person or entity.
22    */
23    public String sayHello()
24    {
25       return "Hello, " + name + "!";
26    }
27
28    private String name;
29 }
```

Il costruttore di Greeter

Si noti il metodo Greeter (stesso nome della classe) definito all'interno della classe:

- per questo metodo non è specificato il tipo del valore restituito;
- un metodo con lo stesso nome della classe è detto costruttore;
- un costruttore restituisce sempre un oggetto della classe corrispondente;
- in questo caso il costruttore prevede una stringa come parametro di input. Per una stessa classe si possono avere costruttori con diverse sequenze di parametri di input;
- un costruttore non viene chiamato come un metodo qualsiasi, ma attraverso il comando **new**, ad esempio

  new Greeter("World")

- se non è specificato nessun costruttore, allora la classe NomeClasse ha il costruttore di default NomeClasse() senza parametri di input (si veda l'esempio della classe Hello precedentemente definita);
- se si definisce un costruttore esplicito, il costruttore di default non è più accessibile.

Proviamo la classe Greeter

Per provare la classe Greeter, possiamo utilizzare BlueJ per

- creare nuovi oggetti di questa classe chiamando **new** Greeter(<string>)
- eseguire il metodo sayHello per generare una stringa di saluti;
- memorizzare o utilizzare la stinga prodotta.

Oppure, possiamo scrivere una classe che contiene un metodo main che esegue i precedenti passi.

```
1 public class GreeterTester
2 {
3     public static void main(String[] args)
4     {
5         Greeter worldGreeter = new Greeter("World");
6         String greeting = worldGreeter.sayHello();
7         System.out.println(greeting);
8     }
9 }
```

Gli oggetti

- Un oggetto è un pezzo di software cui possiamo associare uno stato e un comportamento.
- Gli oggetti sono spesso usati per modellare cose del mondo reale, o appartenenti al dominio di qualche problema (ad esempio, un'automobile familiare di colore rosso, o una lista di interi maggiori di 0).
- Quindi, dal punto di vista del disegno dei programmi, un oggetto modella o rappresenta una qualche entità concreta o astratta con un proprio stato e con un proprio comportamento.
- Gli oggetti sono creati dinamicamente (in Java dal comando **new**) e tutti gli oggetti hanno un proprio stato. (In Java non c'è bisogno di distruggere gli oggetti, spariscono automaticamente per garbage collection quando non più utilizzati).
- Lo stato di un oggetto è rappresentato dalle sue variabili, le cosiddette variabili di istanza o attributi dell'oggetto.
- Il comportamento degli oggetti è modellato dalla funzioni ad essi associati, i metodi dell'oggetto.
- I metodi possono avere dei parametri per ricevere informazioni necessarie alla loro esecuzione.

Metodi e messaggi

Spesso, nella programmazione OO, anziché parlare di invocazione dei metodi, si usa la metafora del passaggio di messaggi:

```
System.out. ptintln ("Hello, world !'')
```

- invia il messaggio corrispondente a println all'oggetto System.out;
- il messaggio porta una stringa ("Hello, world!") come informazione aggiuntiva.

Possiamo dire che:

- I metodi public corrispondono ai messaggi che un oggetto può ricevere.
- Un oggetto può avere metodi private non visibili fuori dell'oggetto, ma utilizzati dagli altri suoi metodi.
- Un oggetto risponde alla ricezione di un messaggio con un'azione (l'esecuzione del codice del corrispondete metodo) e restituendo un valore (il valore di ritorno del metodo).
- I parametri dei metodi corrispondono a informazione aggiuntiva necessaria all'esecuzione dell'azione associata al messaggio.
- I metodi possono accedere allo stato dell'oggetto che riceve il messaggio, ad esempio, possono leggerlo e/o modificarlo;
- in pratica i metodi hanno sempre un parametro implicito: l'istanza dell'oggetto che riceve il messaggio.

Lo stato di un oggetto

- Lo stato di un oggetto è rappresentato dal valore dei suoi attributi.
- Un attributo o variabile d'istanza è una variabile dichiarata all'interno di una classe e non preceduta dall'attributo **static**.
- Gli attributi possono essere public e quindi accessibili (in lettura e scrittura) anche all'esterno dell'oggetto. Ad esempio, se l'oggetto obj ha un attributo public attribute, si può accedere all'attributo con

  obj.attribute
- Gli attributi private sono visibili solo all'interno dell'oggetto, ovvero, possono essere letti e modificati solo dai suoi metodi.
- Dichiarando private gli attributi di un oggetto si nasconde il suo stato all'esterno (information hiding).
- Si possono però prevedere metodi appositi per accedere a tutto o parte dello stato di un oggetto o per ottenere il valore di un attributo composto calcolato a partire dallo stato dell'oggetto.
- Nascondendo gli attributi effettivamente dichiarati all'interno di una classe e rendendo l'oggetto accessibile solo attraverso alcuni metodi, possiamo astrarre rispetto all'effettiva implementazione dell'oggetto e mostrare all'utente un oggetto con uno stato e un comportamento indipendenti da inutili dettagli implementativi.

Le classi

- Le classi rappresentano gli oggetti di un certo tipo.
- Le classi sono una sorta di schema o prototipo dal quale creare gli oggetti.
- I metodi e gli attributi **static** sono comuni a tutti gli oggetti di una classe e sono quindi detti variabili o metodi della classe. Ad esempio,
  - out è una variabile static della classe System ed essendo public è accessibile con System.out;
  - la classe Math di Java ha vari metodi static per il calcolo delle più comuni funzioni matematiche, tra cui il metodo **int** abs(**int** a) per il calcolo del valore assoluto di un intero, che può essere chiamato con (se i è un int)

    Math.abs(i)
- Quando si definisce una classe si definisce automaticamente anche il tipo degli oggetti di quella classe, o meglio dei riferimenti a oggetti della classe. Ad esempio, in

  Greeter worldGreeter = **new** Greeter("World");

  la variabile worldGreeter ha tipo Greeter, ovvero contiene un riferimento a un oggetto della classe Greeter. Anche **new** restituisce un riferimento a un oggetto della classe del costruttore che segue new.
- Si osservino i differenti usi del nome di una classe:
  - per definire la classe e il file ad essa associato;
  - per definire i costruttori della classe;
  - per il tipo dei riferimenti agli oggetti della classe.

Ticket machine (1): descrizione

Costruiamo un esempio a partire da un oggetto del mondo reale.

Definiamo una classe che modella una emettitrice di biglietti.

- Ogni oggetto della classe corrisponderà ad una macchina emettitrice e potrà emettere solo un tipo di biglietto di prezzo assegnato.
- La macchina dovrà essere in grado di dare il resto e
- dovrà mantenere traccia del totale dei soldi incassati.

Lo stato della macchina dovrà pertanto mantenere le informazioni relative

- al prezzo del biglietto che la macchina può emettere;
- ai soldi immessi dal cliente durante l'operazione di acquisto in corso;
- ai soldi totali incassati.

Interagendo con la macchina dovrà essere possibile (almeno)

- conoscere il prezzo del biglietto e i soldi inseriti e non ancora utilizzati;
- stampare il biglietto;
- ottenere la restituzione dei soldi non utilizzati.

Si osservi come anche in casi molto semplici una classe deve modellare sia lo stato che il comportamento dei suoi oggetti.

Ticket machine (2): lo stato

```
1  /**
2   * TicketMachine models a ticket machine that issues
3   * flat-fare tickets.
4   * The price of a ticket is specified via the constructor.
5   * Instances will check to ensure that a user only enters
6   * sensible amounts of money, and will only print a ticket
7   * if enough money has been input.
8   *
9   * @author David J. Barnes and Michael Kolling
10  * @version 2008.03.30
11  */
12 public class TicketMachine
13 {
14     // The price of a ticket from this machine.
15     private int price;
16     // The amount of money entered by a customer so far.
17     private int balance;
18     // The total amount of money collected by this machine.
19     private int total;
```

- Lo stato della macchina è mantenuto privato e nascosto all'utente.

Ticket machine (3): il costruttore

```
21      /**
22       * Create a machine that issues tickets of the given price.
23       */
24      public TicketMachine(int ticketCost)
25      {
26          price = ticketCost;
27          balance = 0;
28          total = 0;
29      }
```

- Il prezzo del biglietto è inizializzato dal costruttore al momento della creazione dell'oggetto.
- Il costruttore inizializza anche gli altri campi, azzerando i soldi immessi e ancora non utilizzati e il totale contenuto nella macchina.

Ticket machine (4): leggere lo stato

```
31      /**
32       * @Return The price of a ticket.
33       */
34      public int getPrice()
35      {
36          return price;
37      }
38
39      /**
40       * Return The amount of money already inserted for the
41       * next ticket.
42       */
43      public int getBalance()
44      {
45          return balance;
46      }
```

- Il valore degli attributi della macchina è letto per mezzo di metodi appositi che possono accedere agli attributi private.

Setter e getter

- I metodi che leggono il valore di un attributo privato sono anche detti getter (o accessor).
- Normalmente il nome di un getter inizia con "get".
- Quando è necessario assegnare valori a degli attributi privati occorre prevedere dei metodi appositi detti setter (o mutator).
- Normalmente il nome di un setter inizia con "set".
  Ad esempio, se si vuole una emettitrice che può cambiare il prezzo dei biglietti che emette, si deve prevedere un metodo setter

```
1 public void setPrice(int newPrice)
2 {
3     price = newPrice;
4 }
```

che assegna il nuovo prezzo alla variabile privata price dell'oggetto.

Ticket machine (5): il biglietto

```
63      /**
64       * Print a ticket if enough money has been inserted, and
65       * reduce the current balance by the ticket price. Print
66       * an error message if more money is required.
67       */
68      public void printTicket()
69      {
70          if(balance >= price) {
71              // Simulate the printing of a ticket.
72              System.out.println("#####################");
73              System.out.println("# The BlueJ Line");
74              System.out.println("# Ticket");
75              System.out.println("# " + price + " cents.");
76              System.out.println("#####################");
77              System.out.println();
78
79              // Update the total collected with the price.
80              total = total + price;
81              // Reduce the balance by the prince.
82              balance = balance - price;
83          }
84          else {
85              System.out.println("You must insert at least: " +
86                                 (price - balance) + " more cents.");
87
88          }
89      }
```

Ticket machine (6): il resto

```
91      /**
92       * Return the money in the balance.
93       * The balance is cleared.
94       */
95      public int refundBalance()
96      {
97          int amountToRefund;
98          amountToRefund = balance;
99          balance = 0;
100         return amountToRefund;
101     }
102 }
```