



Laboratorio – Primo Impatto

Scrivere un semplice programma
I primi concetti

22/10/2004

Programmazione e Laboratorio di Programmazione – Luca Tesei

1

Cosa fare prima di iniziare

- Effettuare il login scegliendo il sistema operativo preferito
- Localizzare la directory (cartella) in cui la J2SE SDK è installata
- Localizzare l'editor con cui si vogliono editare i programmi (consigliato: Ginipad, ma in teoria va bene un qualsiasi editor per file di testo)
- Aprire l'editor per cominciare a scrivere il primo programma Java

22/10/2004

Laboratorio di Programmazione – Luca Tesei

2

L'editor

- Se si è scelto di usare Ginipad o altri editor evoluti e specializzati per Java (emacs, Jedit, l'ambiente JavaBeans, bluej, ...) le operazioni di compilazione, correzione ed esecuzione dei programmi sono facilitate
- I programmi possono essere formattati automaticamente
- Le parole chiave e le componenti sintattiche sono evidenziate con colori diversi
- Familiarizzare bene con l'editor e personalizzarlo (si legga l'help in linea)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

3

Hello World

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

- Salvare
- Compilare (tasto compile su Ginipad)
- Eseguire (tasto run su Ginipad)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

4

Compilazione da riga di comando

- Alternativamente si può compilare e mandare in esecuzione un programma da una finestra di console/shell
- Da Windows: prompt dei comandi (o prompt di MSDOS)
- Da Linux: finestra di shell (bash, csh, tcsh, ...)
- La variabile di ambiente PATH deve contenere il path alla cartella di installazione della SDK, sottocartella "bin" (es. PATH =C:\Program Files\j2sdk_nb\j2sdk1.4.2\bin...)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

5

Compilazione da riga di comando

```
C:\Java> javac Hello.java
```

```
C:\Java>
```

- Il compilatore è rimasto "silenzioso" e quindi la compilazione è andata a buon fine.
- Nella stessa directory del file sorgente (.java) apparirà il file compilato (.class)
- Il file compilato ha sempre lo stesso nome del sorgente, ma con estensione diversa (es. Hello.class)
- Il comando "javac" fa partire il compilatore Java sui file .java indicati di seguito nella linea di comando (in questo caso solo uno)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

6

Avvio della Java Virtual Machine

```
C:\Java> java Hello  
Hello, World!
```

```
C:\Java>
```

- Il comando “java” inizializza la JVM ed esegue il file bytecode Hello.class (non occorre specificare .class)
- Il programma stampa la riga “Hello, World!” ed esce

22/10/2004

Laboratorio di Programmazione – Luca Tesei

7

Guardiamo il programma

Definisce una classe pubblica di nome Hello

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

- Ogni file sorgente può contenere al massimo una classe pubblica e si deve chiamare con lo stesso nome (in questo caso Hello.java)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

8

Nomi e lettere maiuscole/minuscole

- I nomi di classe, per convenzione:
 - Devono sempre iniziare con lettera maiuscola
 - Non devono contenere spazi
 - Se contengono più parole, la nuova parola comincia con lettera maiuscola. Es: HelloWorld, InsiemeDiCaratteri, ContoPersonalizzato
- Java è “case sensitive”: le lettere maiuscole e minuscole sono considerati differenti:
HelloWorld ≠ HelloWorld

22/10/2004

Laboratorio di Programmazione – Luca Tesei

9

Ancora convenzioni

- Le parole riservate Java (class, public, static, int, ...) sono in lettere minuscole
- I nomi di altre entità diverse dalle classi (variabili, nomi di metodi, campi) devono iniziare per lettera minuscola
- Se sono formati da più parole, queste devono essere attaccate come per le classi
- Esempi: x, int, println, indiceDiAscolto, sommaDaRiportare, ...

22/10/2004

Laboratorio di Programmazione – Luca Tesei

10

Indentazione

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

Le parole chiave e gli identificatori sono separati da spazi bianchi

- Spazi bianchi sono lo spazio, il tab, il newline
- n spazi bianchi equivalgono a uno

22/10/2004

Laboratorio di Programmazione – Luca Tesei

11

Indentazione

- Il compilatore non bada alla formattazione del testo del programma
- Il programmatore invece viene facilitato nella lettura/scrittura/modifica se il testo evidenzia la struttura logica del programma
- E' importante fissare e seguire delle regole di indentazione (l'editor le suggerisce)
- Comunque il compilatore compila anche questo:

22/10/2004

Laboratorio di Programmazione – Luca Tesei

12

Come NON va indentato un programma

```
public class
Hello { public
static
void main(String[]          args){
    // visualizza un messaggio sulla finestra
    // di console

    System.out.println

    ("Hello, World!");}
}
```

class??

- Un programma Java è, in generale, la definizione di un insieme di classi
- Una classe si dichiara con la parola riservata `class` seguita dal nome della classe e da un blocco (tutto ciò che appare tra due parentesi graffe)
- Opzionalmente, prima della parola `class` è possibile inserire la parola `public` che indica che la classe è utilizzabile dal “pubblico” (chiariremo in seguito)

Classi

- Una classe è, idealmente, la collezione di tutti i possibili oggetti che hanno la forma da lei definita
- Una classe è la definizione di un tipo a cui tutti gli oggetti della classe sono uniformati
- Una classe è una mera definizione
- I programmi utilizzano questa definizione per creare e usare oggetti di quella classe
- Ogni oggetto della classe è diverso dagli altri oggetti della stessa classe, ma tutti hanno la stessa struttura

Definizione di una classe

```
public class NomeClasse {
    . . .
}
```

- All'interno del blocco possono essere definiti
 - Variabili istanza (dette anche campi)
 - Metodi

Metodi

In questa particolare classe non ci sono variabili istanza.

Definisce un metodo di nome main

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

Corpo del metodo: un altro blocco

Il metodo speciale main

- Ogni applicazione Java deve contenere almeno una classe pubblica con un metodo:
 - Pubblico (parola chiave `public`)
 - Statico (parola chiave `static`)
 - Di nome “main”
 - Che ha come parametro un array di stringhe: (`String[] args`) (il nome `args` del parametro non importa, può essere un nome qualsiasi, l'importante è il tipo “array di stringhe”)

Metodi

- Un metodo contiene al suo interno una sequenza di istruzioni
- In generale i metodi operano su *oggetti*
- I metodi statici, come main, non operano su oggetti, ma si riferiscono a tutta una classe
- Un metodo può avere, in generale, zero o più parametri di ingresso (appaiono tra le parentesi tonde che seguono il nome)
- Per ogni parametro va indicato un tipo e un nome

22/10/2004

Laboratorio di Programmazione – Luca Tesei

19

Il metodo speciale main

- Nel caso di main il parametro è un array (una sequenza) di stringhe che la JVM inizializza con le parole digitate sulla riga di comando, dopo `java Hello`, all'invocazione del programma
- Il contenuto del metodo main della classe pubblica invocata dal comando `java` rappresenta la sequenza principale di istruzioni che il programma deve eseguire

22/10/2004

Laboratorio di Programmazione – Luca Tesei

20

Commenti

Commento al codice:
ignorato dal
compilatore

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

22/10/2004

Laboratorio di Programmazione – Luca Tesei

21

Commenti

- I commenti nel codice sono parte integrante del programma anche se non sono considerati dal compilatore
- I commenti spiegano cosa fa il programma, il significato delle variabili, il comportamento dei metodi, ecc.
- È buonissima abitudine commentare il codice: esso risulta più leggibile da altri (e dallo stesso programmatore già qualche giorno dopo la sua scrittura)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

22

Formato dei commenti Java

- Commento di una linea:
 `// testo del commento`
 – Il compilatore ignora `//` e tutti i caratteri che seguono fino alla fine della linea
 – È utile per i commenti corti
- Commento su più linee:
 `/* testo del
 commento */`
 – Il compilatore ignora tutto ciò che è compreso tra `/*` e il successivo `*/`

22/10/2004

Laboratorio di Programmazione – Luca Tesei

23

Invocazione di un metodo

Il metodo `println` viene invocato passandogli la stringa `"Hello, World!"`.
Stampa la stringa sullo standard output

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

22/10/2004

Laboratorio di Programmazione – Luca Tesei

24

Invocazione di un metodo

- Un metodo non statico deve essere sempre invocato su un certo oggetto
- In questo caso l'oggetto è "out", lo stream standard di caratteri in uscita associato al programma (normalmente è la finestra di console da cui è stato lanciato il programma)
- L'oggetto out è reperibile a partire dalla classe speciale `System`, inizializzata dalla JVM, contenente diversi campi statici che si riferiscono ad oggetti relativi al computer che si sta usando

22/10/2004

Laboratorio di Programmazione – Luca Tesei

25

Uso dell'operatore "."

- Per selezionare un campo statico da una classe:
`System.out = oggetto di tipo PrintStream`
- Per selezionare un campo o per chiamare un metodo di un oggetto:
`System.out.println("Hello, World!");`
- `println(String s)` è un metodo definito nella classe `PrintStream` (inclusa nei packages della SDK) che invia i caratteri della stringa `s`, seguiti da un newline, sullo stream rappresentato dall'oggetto su cui si invoca il metodo (in questo caso l'oggetto `System.out`)

22/10/2004

Laboratorio di Programmazione – Luca Tesei

26

Stringhe

- Una stringa è una sequenza finita di caratteri racchiusa da ""
- Una stringa è un oggetto della classe `String`
- In Java il set di caratteri è Unicode (65536 caratteri dagli alfabeti di tutto il mondo)
- Sia gli identificatori che le stringhe possono contenere caratteri qualsiasi di Unicode

22/10/2004

Laboratorio di Programmazione – Luca Tesei

27

Stringhe

- `"Ciao \t Ciao \n"` Unicode `\\u007D\`: `\u007D`
- Corrisponde a:
`Ciao Ciao`
`"Unicode \u007D": }`
- `\t` è il carattere tab, `\n` è il carattere "a capo" newline
- `\u007D` è una sequenza di escape che rappresenta il carattere Unicode "}" corrispondente al numero esadecimale 007D = 125 decimale
- `\` sono i doppi apici e `\\` è il backslash
- I primi 128 caratteri Unicode corrispondono ai caratteri ASCII a 7 bit

22/10/2004

Laboratorio di Programmazione – Luca Tesei

28

Parametri

- Il metodo `println` della classe `PrintStream` viene invocato sull'oggetto `System.out` con un parametro di tipo `String` (L'oggetto della classe `String` corrispondente a `"Hello, World!"` è implicitamente creato scrivendo la stringa fra apici nel programma)
- Per la stessa classe di oggetti `PrintStream` sono definiti metodi di nome `println` che accettano anche un numero intero, un numero in virgola mobile, un carattere singolo, un intero oggetto!
- `System.out.println(3+5); // Stampa 8`

22/10/2004

Laboratorio di Programmazione – Luca Tesei

29

Errori di compilazione

- Proviamo a modificare il codice di `Hello.java` inserendo qualche imprecisione sintattica:

```
public class Hello {  
    public static void main(String[] args){  
        // visualizza un messaggio sulla finestra di  
        // console  
        System.ouch.println("Hello, World!");  
    }  
}
```

22/10/2004

Laboratorio di Programmazione – Luca Tesei

30

Errori di compilazione

```
C:\Java> javac Hello.java
Hello.java:4: cannot resolve symbol
symbol: variable ouch
location: class java.lang.System
    System.ouch.println("Hello, World!");
           ^
1 error
C:\Java>
```

- Il compilatore indica la riga (4) e il punto (^) in cui ha trovato errore e un breve messaggio sulla natura dello stesso

Errori di Compilazione

- Gli editor evoluti danno la possibilità di andare, cliccando sull'errore, direttamente alla linea di codice incriminata
- Gli editor evoluti si accorgono degli errori di sintassi anche durante la scrittura del programma
- I colori delle diverse componenti sintattiche sono dei validi aiuti per prevenire gli errori durante la scrittura

Errori di compilazione

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

Errori di compilazione

```
C:\Java> javac Hello.java
Hello.java:4: unclosed string literal
    System.out.println("Hello, World!);
                        ^
Hello.java:5: ')' expected
    }
    ^
2 errors
C:\Java>
```

- Il compilatore ha trovato due errori derivanti entrambi dall'omissione del " di chiusura e li segnala entrambi (dopo il primo errore trovato tenta di continuare ad interpretare il codice che segue)

Errori a runtime

```
public class Hello {
    public static void Main(String[] args){
        // visualizza un messaggio sulla finestra di
        // console
        System.out.println("Hello, World!");
    }
}
```

Errori a runtime

```
C:\Java> javac Hello.java

C:\Java> java Hello
Exception in thread "main" java.lang.NoSuchMethodError:
    main
C:\Java>
```

- Il compilatore non dà nessun errore
- Durante l'esecuzione della JVM non viene trovato il metodo main della classe e quindi viene sollevata un'eccezione (un meccanismo per la gestione degli errori) che fa terminare il programma

Errori logici

```
public class Hello {  
    public static void main(String[] args){  
        // visualizza un messaggio sulla finestra di  
        // console  
        System.out.println("Hell, World!");  
    }  
}
```



22/10/2004

Laboratorio di Programmazione – Luca Tesei

37

Errori logici

C:\Java> javac Hello.java

C:\Java>

- Il compilatore non ha trovato nessuna imperfezione sintattica nel programma
- Il programma stampa la riga: Hell, World!
- Questo tipo di errori non è rilevabile dal compilatore
- L'unico, difficile, modo per trovarli (a parte che in casi semplici come questo) è quello di analizzare attentamente il programma magari con l'aiuto di un debugger

22/10/2004

Laboratorio di Programmazione – Luca Tesei

38

Esercizi

- Scrivere un programma che stampi il proprio nome in un rettangolo disegnato coi caratteri ASCII tipo

```
+-----+  
|   Luca   |  
+-----+
```

22/10/2004

Laboratorio di Programmazione – Luca Tesei

39

Esercizi

- Scrivere un programma che stampi la scala

```
+-----+  
|       |  
+-----+-----+  
|       |       |  
+-----+-----+-----+  
|       |       |       |  
+-----+-----+-----+
```

22/10/2004

Laboratorio di Programmazione – Luca Tesei

40

Esercizi

- Scrivere un programma che stampi tutti i caratteri Unicode dal 120 al 130 (decimali)
- Scrivere un programma che stampi la somma dei primi 10 numeri naturali
- Scrivere un programma che stampi il primo (args[0]) dei suoi argomenti della linea di comando. Mandarlo in esecuzione con zero, uno, due argomenti sulla linea di comando
- Scrivere un programma che generi un congruo numero di errori di compilazione a causa di un limitato numero di errori di digitazione ☺

22/10/2004

Laboratorio di Programmazione – Luca Tesei

41