

Primi programmi in Java

Lezione IV

Scopo della lezione

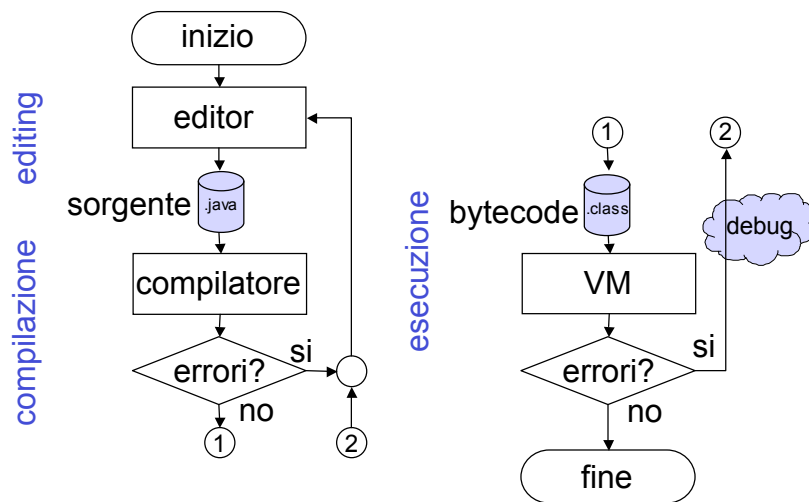
- Realizzare dei semplici programmi scritti in Java.
- Esercitarsi nelle operazioni necessarie per passare dalla scrittura di codice Java all'esecuzione del programma correlato.
- Utilizzare le classi di I/O.
- Familiarizzare con il concetto di variabile.

Il processo di sviluppo

- Passo 1: scrivere il programma
 - Software: un qualsiasi text editor
- Passo 2: compilare il programma
 - Software: Java Development Kit (JDK)
 - Comando: `javac HelloWorld.java`
- Passo 3: eseguire il programma
 - Software: JDK o Java Runtime environment
 - Comando: `java HelloWorld`

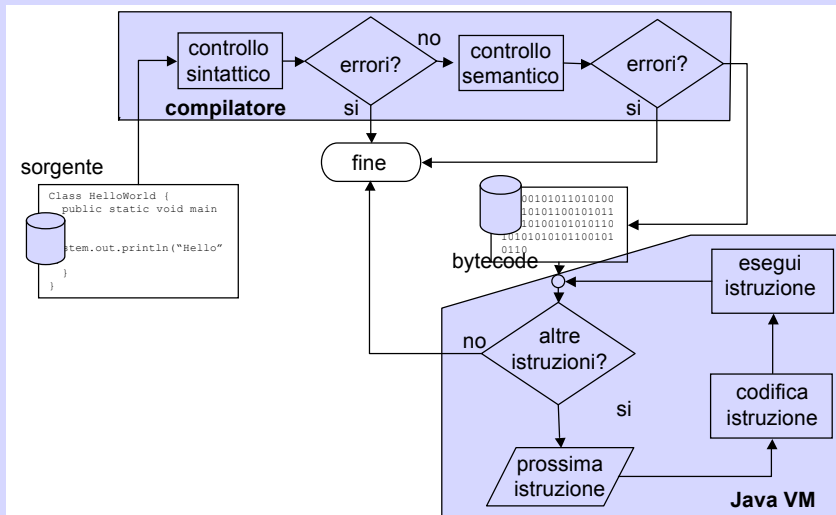
III.3

Virtual machine



III.4

Virtual machine



III.5

Scrivere un programma java

- Software: un text editor.
- Il **codice sorgente** deve avere come nome il nome della classe implementata e come estensione java.
- Nota: i nomi dei file sono case-sensitive.

III.6

Compilare un programma java

- La fase di compilazione traduce il sorgente java in **bytecode** (indipendente dalla piattaforma).
- La compilazione avviene eseguendo il comando `javac NomeClasse.java`.
- Se termina con successo, la compilazione genera una serie di file `.class` contenenti il bytecode delle varie classi definite nel codice sorgente.

III.7

Eseguire un programma java

- Il file contenente il bytecode della classe deve essere caricato in memoria ed interpretato da una java virtual machine (JVM).
- L'esecuzione avviene lanciando il comando `java NomeClasse`.
- Nota: per poter essere eseguita in questo modo, una classe deve implementare un metodo `main()`.

III.8

Prima di cominciare...

- Nello scrivere i nostri programmi ci atterremo alle seguenti regole
 - Faremo riferimento a una directory principale
 - Tutti i file relativi a un programma risiederanno in una sottodirectory con lo stesso nome di quest'ultimo (derogando in caso di modifiche di lieve entità)
 - Il nome di un programma sarà scritto in caratteri minuscoli, a eccezione delle iniziali delle parole

III.9

Inoltre

- Ci sforzeremo di dare ai nostri programmi dei nomi significativi

III.10

Quindi

- Sono nomi validi per i programmi
 - Programma, MioProgramma, UnProgrammaBellissimo
- NON sono nomi validi per i programmi
 - programma, ProGraMMa, PROGRAMMA, Programma1, Programma2, Programma3, ...

III.11

HelloWorld.java

```
/* Questo è il nostro primo programma
   scritto in java */

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

// Analizziamolo!
```

III.12

Compiliamolo!

```
[~]malchiod% cd HelloWorld
[~/HelloWorld]malchiod% ls
HelloWorld.java
[~/HelloWorld]malchiod% javac HelloWorld.java
[~/HelloWorld]malchiod%
```

III.13

Cosa è successo?

```
[~/HelloWorld]malchiod% ls
HelloWorld.class  HelloWorld.java
[~/HelloWorld]malchiod% ls
```

III.14

Eseguiamolo!

```
[~/HelloWorld]malchiod% java HelloWorld
Hello, world!
[~/HelloWorld]malchiod%
```

- L'unica (apparente) azione eseguita dal programma è quella di stampare il messaggio "Hello, world!" a video, andando a capo

III.15

Analizziamo il programma

```
/* Questo è il nostro primo programma  
scritto in java */
```

```
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello, world!");
    }
}
// Analizziamolo!
```

- Il testo racchiuso tra `/*` e `*/` rappresenta un commento e viene ignorato: provate a modificarlo e verificate che, ricompilando il programma, l'esecuzione non cambia

III.16

Analizziamo il programma

```
/* Questo è il nostro primo programma  
scritto in java */
```

```
class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello, world!");  
    }  
}  
// Analizziamolo!
```

- Ciò che segue // viene ignorato fino al termine della riga: provate a ricompilare il programma dopo avere modificato il commento, anche aggiungendo testo alla linea successiva

III.17

Commenti

- La presenza di commenti nel codice non ne modifica il comportamento ...
- ... ma ne aumenta notevolmente la leggibilità
- E' quindi utile inserire commenti per chiarire i punti fondamentali di un programma

III.18

Commenti

- Potremmo modificare il programma in

```
/* Questo è il nostro primo programma
   scritto in java */

class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello, world!"); // Output del programma
    }
}
// Analizziamolo!
```

- e l'esecuzione non varierebbe

III.19

Analizziamo il programma

- Commenti a parte, il programma è delimitato dalla parola chiave `class` seguita dal nome del programma e da una coppia di parentesi graffe

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

III.20

Analizziamo il programma

- In generale le parentesi graffe delimitano blocchi di codice
- Diversi blocchi di codice seguono una diversa **indentazione**

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

III.21

Il blocco `main()`

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

- Rappresenta il blocco di istruzioni che viene eseguito quando si esegue il programma
- Per ora non ci interesseremo delle parole chiave che precedono e seguono `main`

III.22

Output

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

- Quando viene eseguita l'istruzione `System.out.println`, il messaggio specificato entro i doppi apici viene stampato sullo schermo, seguito da un carattere di "a capo"
- Il comando `System.out.print` ha lo stesso comportamento, ma non va a capo

III.23

Aggiungiamo degli errori

- Provate a modificare ogni parola che compare nel sorgente del programma, e verificate in quali casi è ancora possibile compilare il programma
- In questi casi, verificate quando è anche possibile eseguire il programma e controllate cosa è variato nell'esecuzione

III.24

HelloWorldPrint.java

```
/* Questo è il nostro secondo programma
   scritto in java */

class HelloWorldPrint {
    public static void main(String args[]) {
        System.out.print("Hello, world!");
    }
}

[~/HelloWorld]malchiod%javac HelloWorldPrint.java
[~/HelloWorld]malchiod%java HelloWorldPrint
Hello, world![~/HelloWorld]malchiod%
```

III.25

Librerie

- Java mette a disposizione una serie di insiemi di comandi e di oggetti deputati a risolvere particolari compiti
- Oggetti con funzionalità simili o collegate sono raggruppati in un insieme che viene chiamato **package**
- Alcuni package possono far parte di altri package

III.26

Package

- Per poter utilizzare un package è generalmente necessario **importarlo** nel programma
 - `import java.util.*` include nel programma tutti i comandi del package
 - `import java.util.Date` include nel programma solamente l'oggetto `Date`
- Alcuni package, come `java.lang`, non necessitano di essere importati

III.27

HelloDate.java

```
/* HelloDate stampa la data corrente */  
import java.util.Date;  
  
public class HelloDate {  
    public static void main(String args[]) {  
        System.out.print("Sono le ");  
        System.out.println(new Date());  
    }  
}
```

III.28

Eseguiamolo!

```
[~/HelloDate]malchiod%javac HelloDate.java
[~/HelloDate]malchiod% java HelloDate
La data corrente e' Mon Oct 13 14:50:16 CEST 2003
[~/HelloDate]malchiod%
```

giorno mese ora fuso orario anno

III.29

Come funziona?

- `new Date()` istanzia un nuovo oggetto della classe `Date` e ritorna un riferimento ad esso.
- `System.out.println` è programmato per riconoscere i riferimenti a oggetti della classe `Date` passatigli come argomento e a trattarli in modo opportuno (cioé stampando la data corrente).

III.30

HelloDateLungo.java

```
/* HelloDateLungo stampa la data corrente */  
import java.util.*;  
  
public class HelloDateLungo {  
    public static void main(String args[]) {  
        System.out.print("Sono le ");  
        System.out.println(new Date());  
    }  
}
```

III.31

Eseguiamolo!

```
[~/HelloDate]malchiod% javac HelloDateLungo.java  
[~/HelloDate]malchiod% java HelloDateLungo  
La data corrente e' Mon Oct 13 14:50:19 CEST 2003  
[~/HelloDate]malchiod%
```

III.32

Dove sta la differenza?

```
[~/HelloDate]malchiod% ls -lsa
total 32
0 drwxr-xr-x  204 Oct 13 14:50 .
0 drwxr-xr-x  204 Oct 13 14:47 ..
8 -rw-r--r--  516 Oct 13 14:50 HelloDate.class
8 -rw-r--r--  185 Oct 13 14:49 HelloDate.java
8 -rw-r--r--  526 Oct 13 14:50 HelloDateLungo.class
8 -rw-r--r--  187 Oct 13 14:50 HelloDateLungo.java
```

permessi

dimensione

data/ora
di creazione

nome.estensione

Laboratorio di Programmazione

III.33

Modifichiamo il sorgente

- Eliminate la linea in cui viene importato il package e provate a ricompilare il programma. Cosa succede?
- Sulla base di quanto visto in precedenza, determinate una modifica che impedirebbe la compilazione, una che impedirebbe l'esecuzione e una che varierebbe l'esecuzione e implementatele

Laboratorio di Programmazione

III.34

Un package importante

- Per gestire l'input da tastiera e l'output a video utilizzeremo rispettivamente le classi
 - `ConsoleInputManager`
 - `ConsoleOutputManager`che fanno parte del package `prog.io` incluso nel libro di testo.

III.35

Attenzione!

- I contenuti del package `prog.io` non fanno parte della distribuzione di Java.
- Per potervi accedere è necessario:
 - copiare i contenuti del package sul proprio computer (dal CD venduto assieme al libro) in una zona opportuna;
 - modificare alcuni settaggi del sistema (tipicamente la variabile di ambiente `CLASSPATH`).

III.36

HelloWorldModificato.java

```
/* Impariamo a usare le classi di I/O */  
  
import prog.io.ConsoleOutputManager;  
  
class HelloWorldModificato {  
    public static void main(String args[]) {  
        ConsoleOutputManager video =  
            new ConsoleOutputManager();  
        video.println("Hello, world!");  
    }  
}
```

III.37

Eseguiamolo!

```
[~/HelloWorld]malchiod%javac  
    HelloWorldModificato.java  
[~/HelloWorld]malchiod% java HelloWorldModificato  
Hello, world!  
[~/HelloWorld]malchiod%
```

III.38

Differenze

- Non vi è alcuna differenza, nell'output prodotto, rispetto a HelloWorld.
- Vi è invece differenza nel modo in cui questo output è stato prodotto: in un caso utilizzando il metodo `println` della classe `System.out`, nell'altro utilizzando il medesimo metodo ma nella classe `ConsoleOutputManager`.

III.39

AreaRettangolo.java

```
/* AreaRettangolo calcola l'area di
   un rettangolo */

public class AreaRettangolo {
    public static void main(String args[]) {
        ConsoleOutputManager video =
            new ConsoleOutputManager();
        int base=3;
        int altezza=4;
        video.print("L'area e' ");
        video.println(base*altezza);
    }
}
```

III.40

Eseguiamolo!

```
[~/AreaRettangolo]malchiod%javac
  AreaRettangolo.java
[~/AreaRettangolo]malchiod% java AreaRettangolo
L'area e' 12
[~/AreaRettangolo]malchiod%
```

III.41

Alcuni dettagli

- L'argomento del metodo `println()` contiene
 - Uno spazio come ultimo carattere, per evitare che nell'output il numero 12 venga scritto attaccato alla frase che lo precede.
 - Il verbo “è” viene scritto “e’”, per evitare che il carattere di “e accentata” possa venire male interpretato nel caso il sistema su cui si eseguirà il programma sia diverso da quello su cui è stato sviluppato.

III.42

Altri dettagli

- Nell'argomento del metodo `print()` è specificata una moltiplicazione. Questa verrà valutata **prima** del metodo `print()`, in modo che il risultato dell'operazione di moltiplicazione diventi l'argomento di `print()`.

III.43

Dati e variabili

- In questo caso il programma deve fare riferimento a dei dati numerici che deve pertanto memorizzare
- Per potersi riferire a questi dati il programma alloca due aree di memoria a cui assegna nomi univoci (`base`, `altezza`)
- Chiamiamo tale aree **variabili**, e chiamiamo **dati** i valori in esso contenuti

III.44

Modifichiamo il sorgente

- Modificate i dati inseriti nelle variabili `base` e `altezza`, e verificate come cambia l'esecuzione del programma
- Che cosa succede se tentate di inserire in queste variabili dei numeri non interi? Chiedete ai docenti come trattare questo tipo di dati.
- Quale altro tipo di informazione potete pensare di associare a una variabile?

III.45

Interattività

- Un programma come `AreaRettangolo` sarebbe più utile se, invece di assegnare valori fissati a `base` e `altezza`, permettesse all'utente di specificarli a ogni esecuzione.
- Per fare questo è necessario introdurre dei comandi che permettano di ricevere questo input dalla tastiera.

III.46

AreaRettangoloInterattivo.java

```
import prog.io.ConsoleOutputManager;
import prog.io.ConsoleInputManager;

public class AreaRettangoloInterattivo {
    public static void main(String args[]) {
        ConsoleOutputManager video
            new ConsoleOutputManager();
        ConsoleInputManager tastiera
            = new ConsoleInputManager();
        int base=tastiera.readInt();
        int altezza=tastiera.readInt();
        video.print("L'area e' ");
        video.println(base*altezza);
    }
}
```

III.47

Eseguiamolo!

```
[~/AreaRettangolo]malchiod%javac
AreaRettangoloInterattivo.java
[~/AreaRettangolo]malchiod% java
AreaRettangoloInterattivo
5
6
L'area e' 30
[~/AreaRettangolo]malchiod%
```

III.48

tastiera.readInt()?

- Possiamo pensare a `ConsoleInputManager.readInt()` come a un comando che
 - attende che l'utente immetta un valore numerico
 - fornisce al programma tale valore
- E' la controparte di `ConsoleOutputManager.println()`

III.49

Abbellimenti

- In questo caso l'esecuzione del programma parte senza dare alcuna indicazione del fatto che il programma attende un'azione dell'utente (l'immissione di base e altezza)
- E' opportuno segnalare questo fatto all'utente tramite degli output esplicativi
- Provate a implementare il programma **SENZA** guardare la soluzione alla slide seguente.

III.50

AreaRettangoloAbbellito.java

```
public class AreaRettangoloAbbellito {
    public static void main(String args[]) {
        ConsoleOutputManager video
            new ConsoleOutputManager();
        ConsoleInputManager tastiera
            = new ConsoleInputManager();
        video.println("Inserisci la base");
        int base=tastiera.readInt();
        video.println("Inserisci l'altezza");
        int altezza=tastiera.readInt();
        video.print("L'area e' ");
        video.println(base*altezza);
    }
}
```

III.51

Eseguiamolo!

```
[~/AreaRettangolo]malchiod%javac
AreaRettangoloAbbellito.java
[~/AreaRettangolo]malchiod% java
AreaRettangoloAbbellito
Inserisci la base
4
Inserisci l'altezza
7
L'area e' 28
[~/AreaRettangolo]malchiod%
```

III.52

Testiamo l'usabilità

- Come è possibile fare in modo che l'esecuzione del programma venga alterata?

III.53

Sfida!

- Scrivete un programma che richiede all'utente di immettere il raggio di un cerchio, e calcola e stampa la sua area
- Attenzione! Per risolvere questo problema è necessario avere capito come trattare i numeri con la virgola. Se vi trovate in difficoltà, chiedete aiuto ai docenti!

III.54

Celsius vs. Fahrenheit

Fahrenheit

-40 0 40 80 120



-40 -20 0 20 40

Celsius

$$\text{fahrenheit} = \text{celsius} * 9/5 + 32$$

Provate a implementare un programma che converte dalla scala celsius alla scala fahrenheit SENZA guardare la soluzione alla slide seguente

III.55

ConvertiTemperature.java

```
public class ConvertiTemperature {
    public static void main(String args[]) {
        ConsoleOutputManager video =
            new ConsoleOutputManager();
        ConsoleInputManager tastiera =
            new ConsoleInputManager();
        int celsius;
        int fahrenheit;

        video.println("Inserisci la temperatura
            in gradi centigradi");
        celsius = tastiera.readInt();
        fahrenheit = celsius * 9/5 + 32;
    }
}
```

III.56

ConvertiTemperature.java

```
        video.print("La corrispondente temperatura  
        in scala fahrenheit e' di ");  
        video.print(fahrenheit);  
        video.println(" gradi.");  
    }  
}
```

III.57

Eseguiamolo!

```
[~/ConvertiTemperature] malchiod% javac  
ConvertiTemperature.java  
[~/ConvertiTemperature] malchiod% java  
ConvertiTemperature  
Inserisci la temperatura in gradi centigradi  
0  
La corrispondente temperatura in scala fahrenheit  
e' di 32 gradi  
[~/ConvertiTemperature] malchiod% java  
ConvertiTemperature  
Inserisci la temperatura in gradi centigradi  
37  
La corrispondente temperatura in scala fahrenheit  
e' di 98 gradi  
[~/ConvertiTemperature] malchiod%
```

III.58

Esercizi

- Realizzare un programma `AltezzaRettangoloInterattivo` che legga l'area e la base di un rettangolo e ne stampi l'altezza.
- Verificare come si comporta il programma
 - quando l'area inserita non è un multiplo esatto della base;
 - quando viene immesso 0 come valore per la base.

III.59

Esercizi

- Scrivere un programma `ConvertiDaFahrenheitACelsius` che realizzi la conversione dalla scala Fahrenheit alla scala celsius
- Scrivere un programma `ConvertiDaKilometriAMiglia` che realizzi la conversione tra chilometri e miglia, tenuto conto del fatto che un miglio equivale a 1.61 chilometri.

III.60

Esercizi

- Scrivere un programma `CentraNomi` che stampi a video i vostri nomi centrandoli in un rettangolo di dimensioni opportune, costruito utilizzando i simboli `-` e `|`. Ad esempio

```
-----  
|      unNome      |  
| unAltroNome      |  
|-----|
```

- **Attenzione!** Questo programma non deve accettare alcun input da tastiera: i vostri nomi devono essere scritti direttamente nel sorgente JAVA.