

Primi programmi in Java

Lezione IV

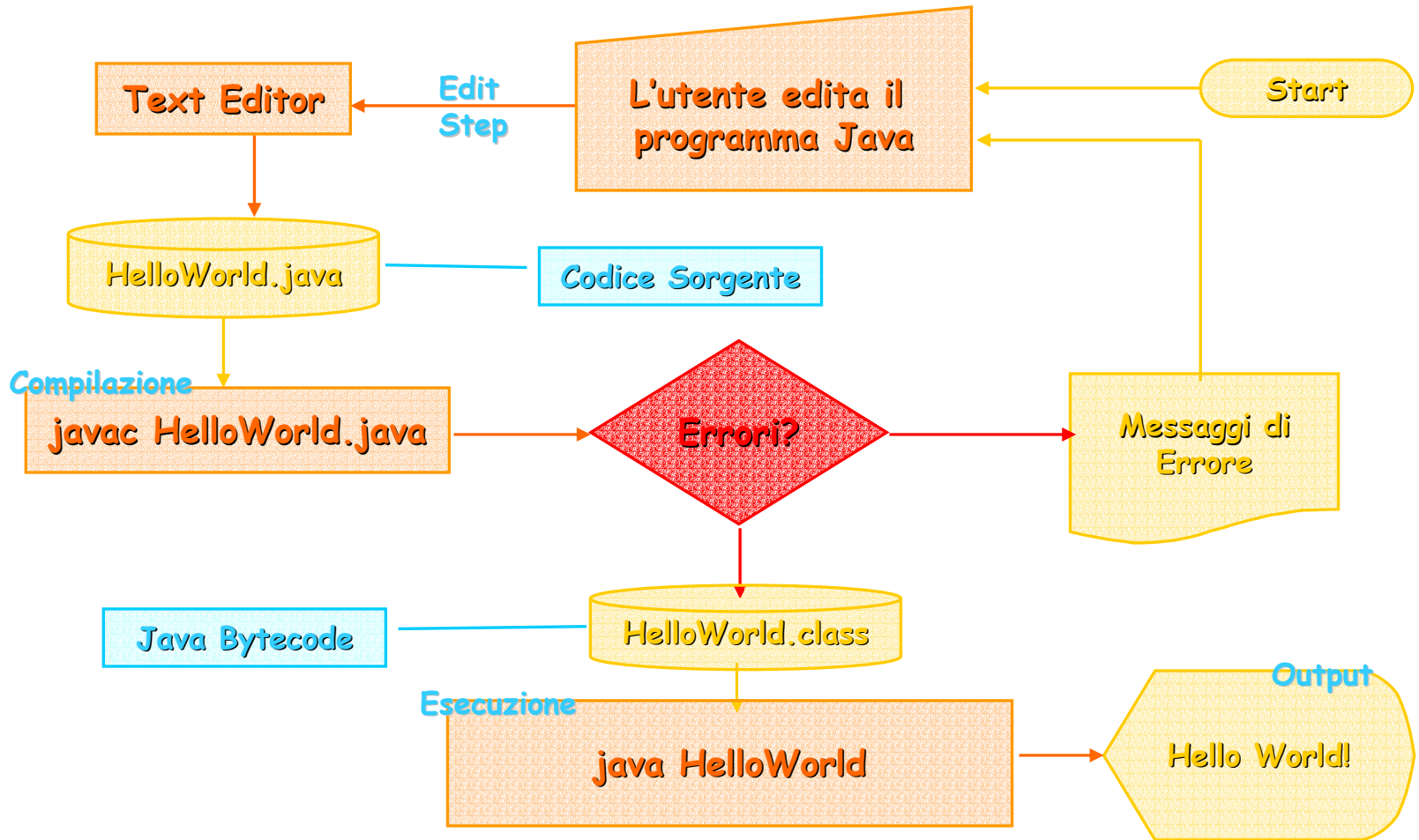
Scopo della lezione

- 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.

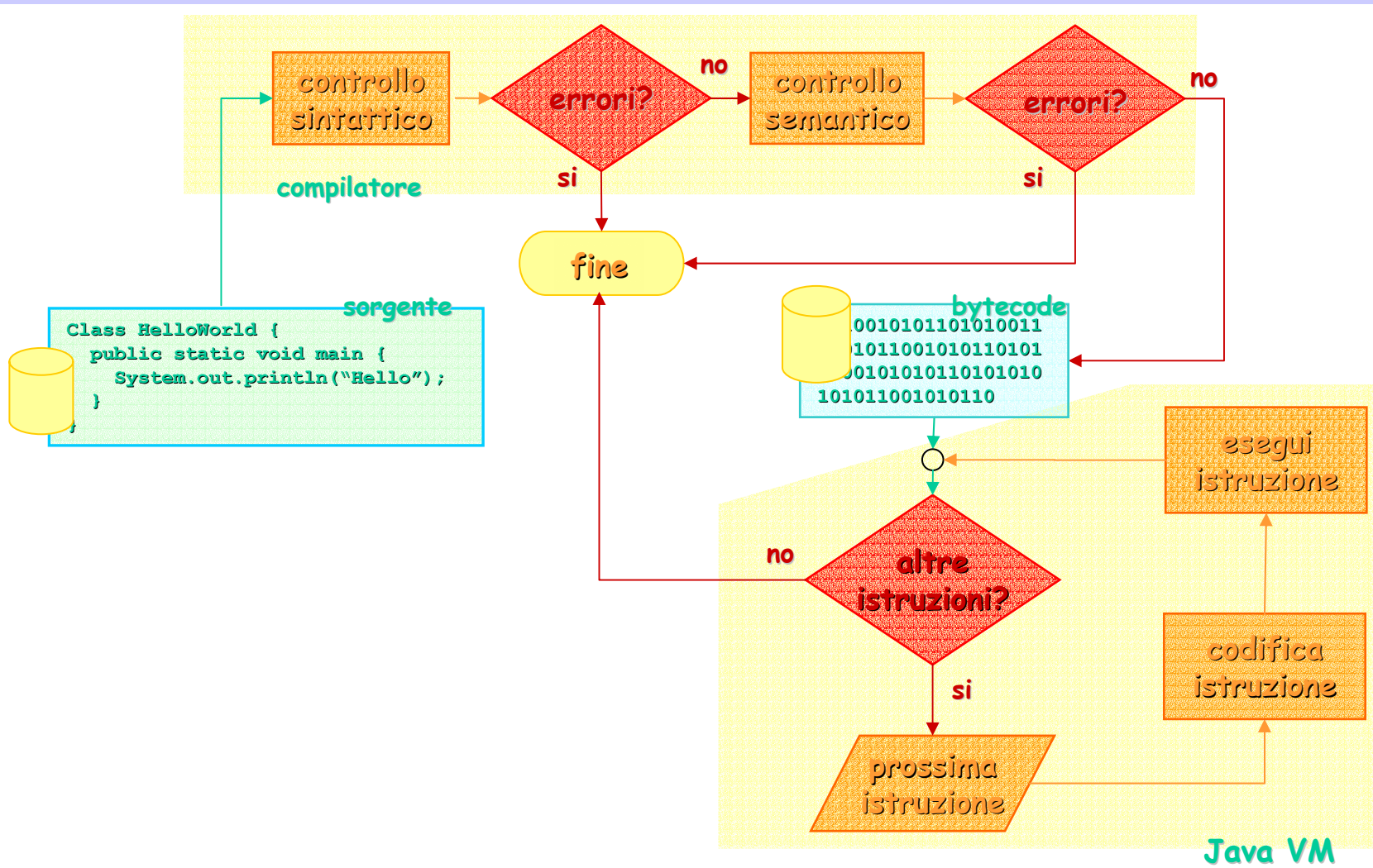
Il processo di sviluppo

- **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`

Compilazione ed Esecuzione



Virtual machine



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 e delle classi sono case-sensitive.

Compilare i Programmi Java

La fase di compilazione traduce il sorgente del programma Java in bytecode.

- Il bytecode è indipendente dalla piattaforma

Comando JDK: `javac HelloWorld.java`

Quando la compilazione termina con successo crea i file contenenti, ognuno, il bytecode di una delle classi definite nel codice compilato, ad es. `HelloWorld.class`

Eseguire un Programma Java

Il file contenente il bytecode della classe deve essere caricato in memoria ed interpretato dalla Java Virtual Machine (JVM)

Comando JDK: `java HelloWorld`

Nota per poter essere eseguita una classe deve definire un metodo di nome `main()`.

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

Inoltre

- Ci sforzeremo di dare ai nostri programmi dei nomi significativi

Quindi

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

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!
```

Compiliamolo!

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

Cosa è successo?

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

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

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

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 `//` rappresenta un commento e viene ignorato (fino al termine della riga)

Commenti

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

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

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!");  
    }  
}
```

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!");  
    }  
}
```

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`

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

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%
```

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

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

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());  
    }  
}
```

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

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 (cioè stampando la data corrente).

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());  
    }  
}
```

Eseguiamolo!

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

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

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.

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).

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!");  
    }  
}
```

Eseguiamolo!

```
[~/HelloWorld]malchiod%javac  
HelloWorldModificato.java
```

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

```
[~/HelloWorld]malchiod%
```

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`.

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);
    }
}
```

Eseguiamolo!

```
[~/AreaRettangolo]malchiod%javac  
AreaRettangolo.java
```

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

```
[~/AreaRettangolo]malchiod%
```

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.

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()`.

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

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.

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);
    }
}
```

Eseguiamolo!

```
[~/AreaRettangolo]malchiod%javac
  AreaRettangoloInterattivo.java
[~/AreaRettangolo]malchiod% java
  AreaRettangoloInterattivo

5
6
L'area e' 30

[~/AreaRettangolo]malchiod%
```

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
- È la controparte di `ConsoleOutputManager.println()`

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)
- È opportuno segnalare questo fatto all'utente tramite degli output esplicativi

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);
    }
}
```

Eseguiamolo!

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

[~/AreaRettangolo]malchiod%
```

Celsius vs. Fahrenheit

Fahrenheit

-40 0 40 80 120



-40 -20 0 20 40

Celsius

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

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;
```

ConvertiTemperature.java

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

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%
```

Esercizi

- **Riscrivete `HelloWorld` modificando**
 - il messaggio che viene stampato
 - aggiungendo altri messaggi
 - sostituendo `println` con `print` e vedendo come viene modificata l'esecuzione del programma (attenti agli spazi!)

Esercizi

- Cosa succede se durante l'esecuzione di `Area Rettangolo Interattivo` l'utente immette dei valori che non sono numerici? E se immette dei valori non interi?
- Scrivete un programma che accetta in input l'area e la base di un rettangolo e ne calcola l'altezza

Esercizi

- Che succede se nel programma precedente immettiamo 0 come valore per la base?
- Provate a inserire deliberatamente degli errori in alcuni dei programmi che abbiamo scritto, e verificate come cambia il comportamento del compilatore