

# Fondamenti di informatica T-1 (A – K)

## Esercitazione 6: Eclipse

AA 2018/2019

Tutor

**Lorenzo Rosa**

[lorenzo.rosa@unibo.it](mailto:lorenzo.rosa@unibo.it)

# Esercitazione 6

Introduzione al calcolatore e Java

Linguaggio Java, basi e controllo del flusso

**Eclipse ed esercizi di consolidamento**

Stringhe ed array

Metodi, classi, oggetti

Ereditarietà e polimorfismo

Collezioni Java e interfacce

Esercizi d'esame

# Eclipse

- **Cos'è?**

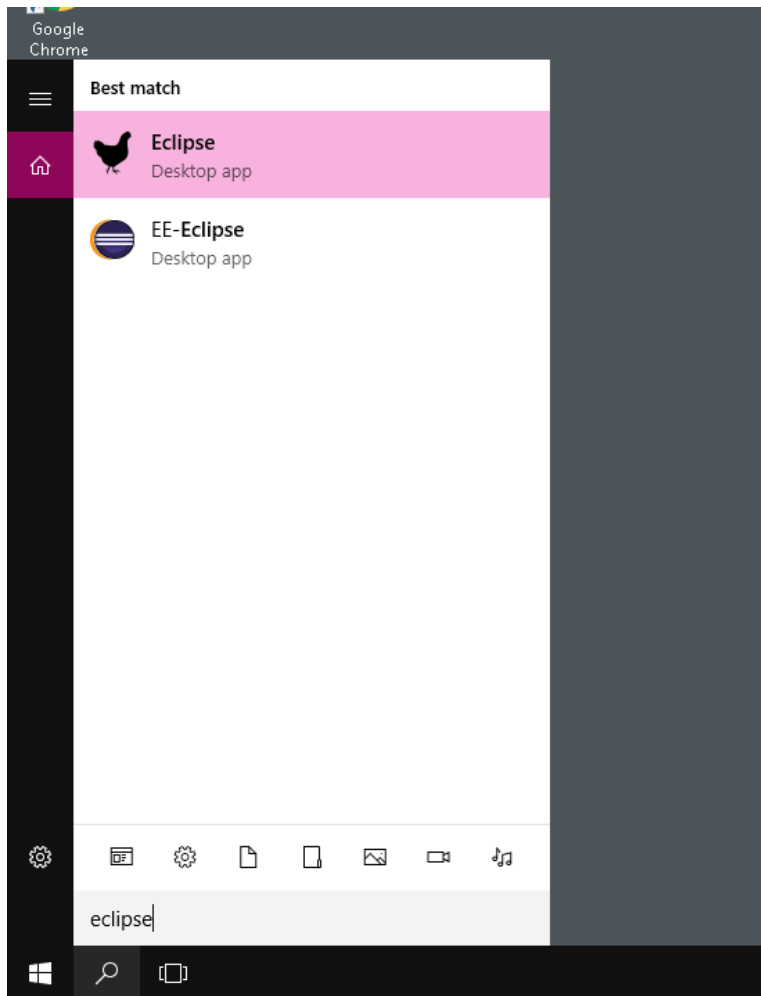
Un ambiente di sviluppo integrato (IDE) contenente:

- un editor di codice sorgente;
- un compilatore e un interprete;
- un debugger.

- **Come scaricarlo ed installarlo?**

- Scaricarlo gratuitamente dal sito (versione IDE for Java Developers): <http://www.eclipse.org/downloads/>
- Estrarre l'archivio
- Eseguire il .exe

# Eclipse



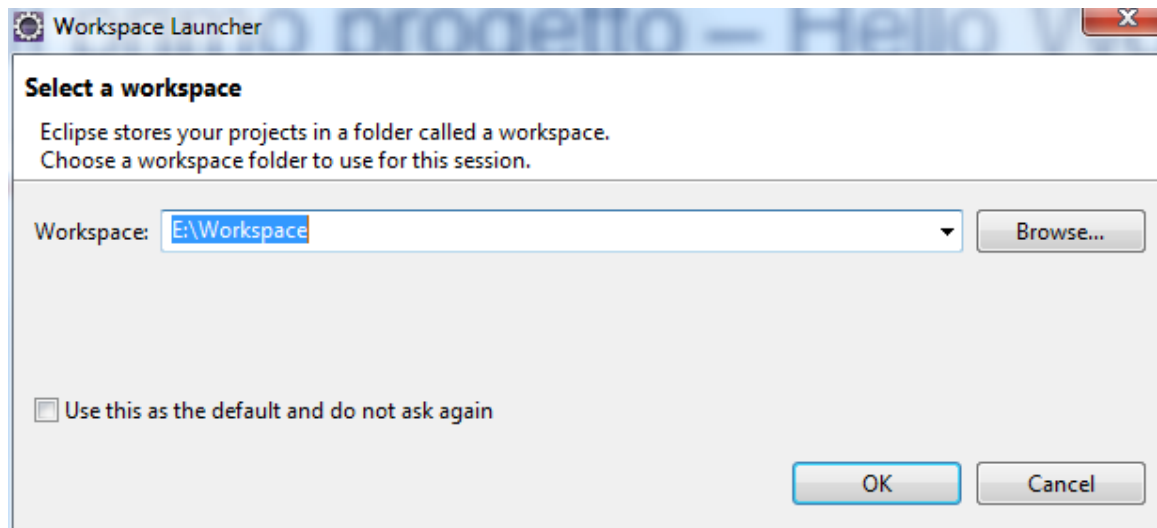
Avviate *Eclipse* tramite il file `eclipse.exe`.

- **In laboratorio** Eclipse ha l'icona mostrata a lato (primo risultato)
- **A casa** avrete un simbolo simile a quello mostrato ne secondo risultato qui a lato.

# Primo avvio, scelta workspace

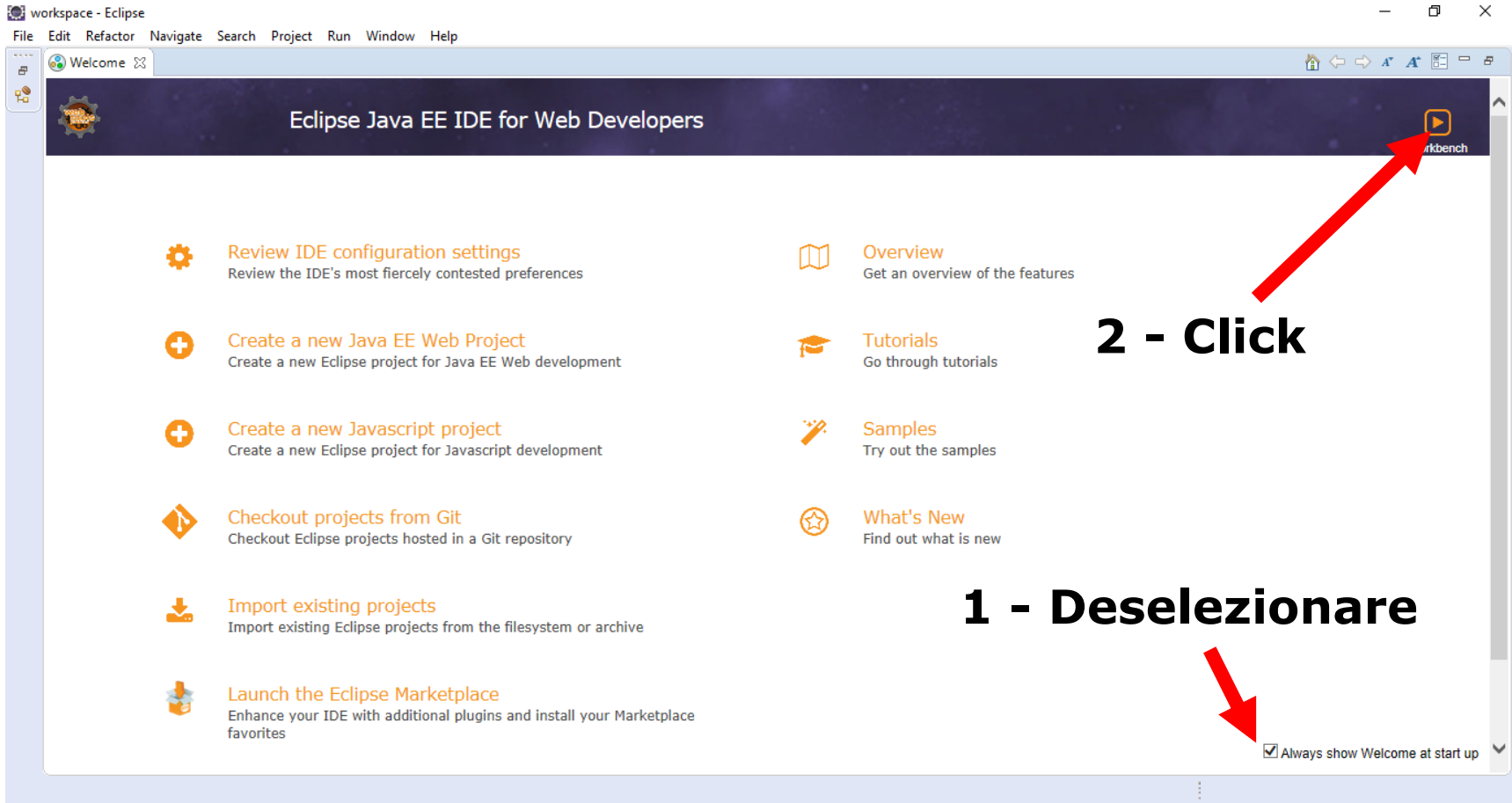
*Workspace*: cartella che contiene i progetti

- **In laboratorio** → la vostra home oppure un dispositivo di archiviazione esterno
- **A casa** → cartella a scelta, Eclipse sceglie automaticamente la home dell'utente (C:\Users\*nome\_utente*\Workspace)



Se non compare questa finestra, non c'è problema: Eclipse vi ha già selezionato una cartella come workspace.

# Primo avvio, messaggio di benvenuto













workspace - Eclipse

File Edit Refactor Navigate Search Project Run Window Help

Welcome

Eclipse Java EE IDE for Web Developers

Workbench

-  **Review IDE configuration settings**  
Review the IDE's most fiercely contested preferences
-  **Overview**  
Get an overview of the features
-  **Create a new Java EE Web Project**  
Create a new Eclipse project for Java EE Web development
-  **Tutorials**  
Go through tutorials
-  **Create a new Javascript project**  
Create a new Eclipse project for Javascript development
-  **Samples**  
Try out the samples
-  **Checkout projects from Git**  
Checkout Eclipse projects hosted in a Git repository
-  **What's New**  
Find out what is new
-  **Import existing projects**  
Import existing Eclipse projects from the filesystem or archive
-  **Launch the Eclipse Marketplace**  
Enhance your IDE with additional plugins and install your Marketplace favorites

**2 - Click**

**1 - Deselezionare**

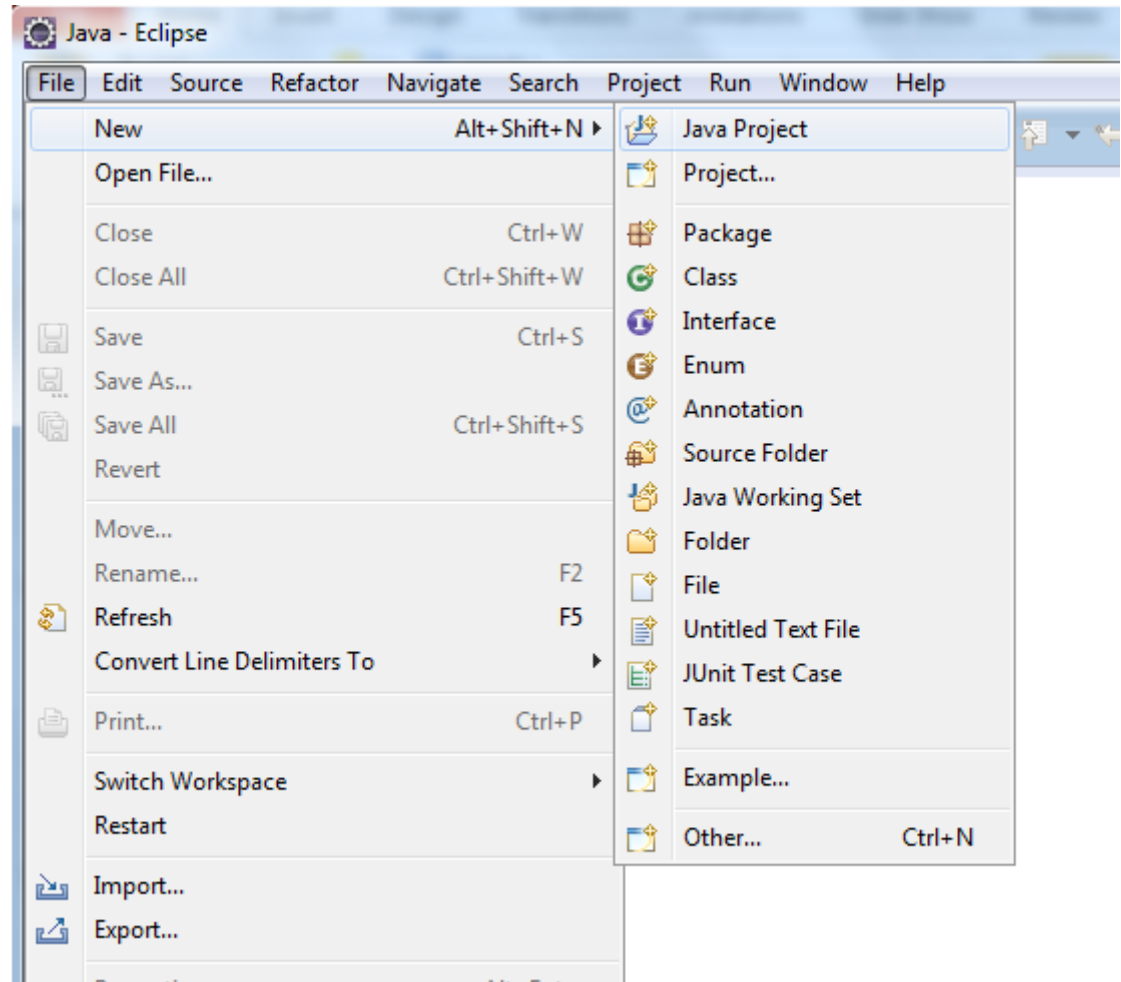
Always show Welcome at start up

# Creazione progetto

Il primo progetto:

- Creazione di un nuovo progetto

File → New →  
Java Project

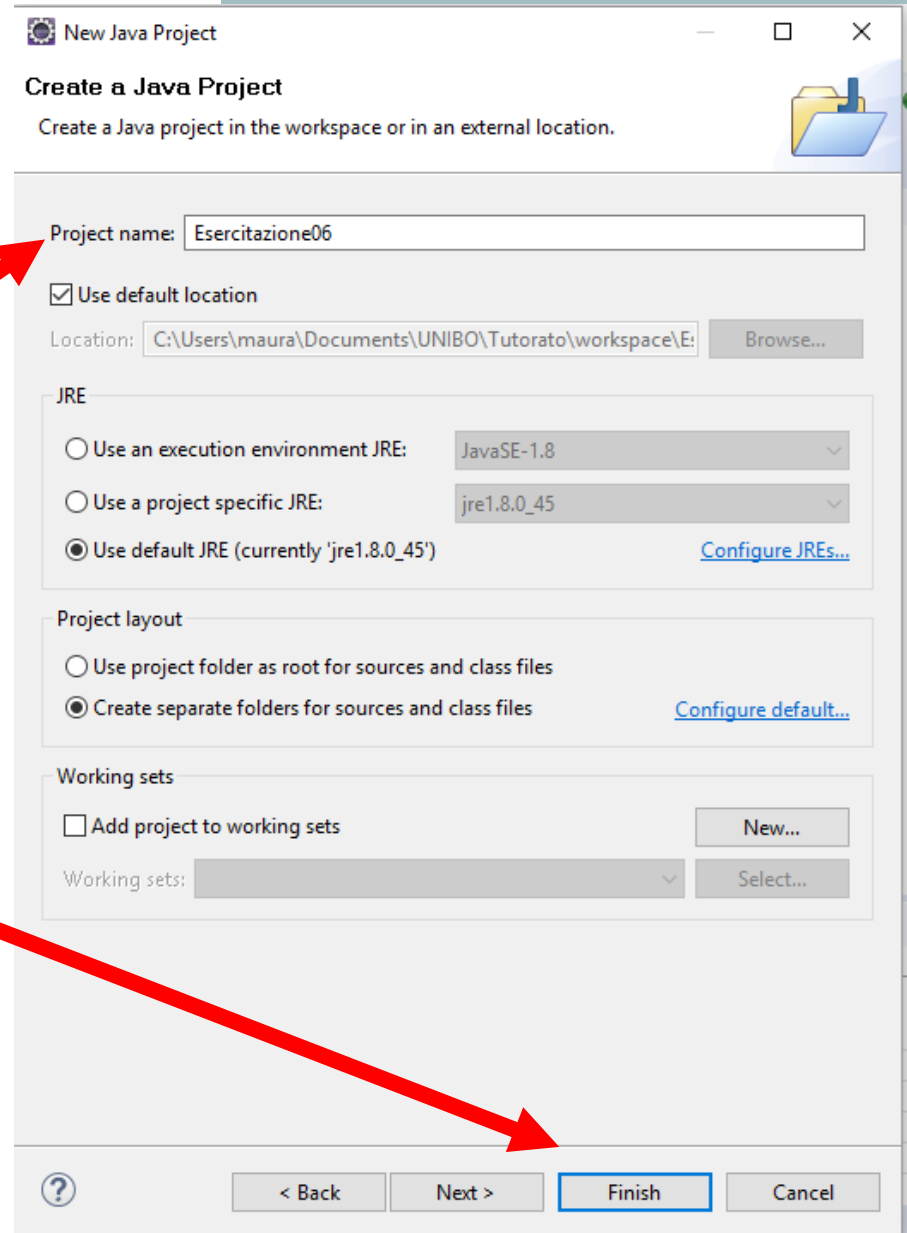


# Creazione progetto

Nome del progetto

Esercitazione06

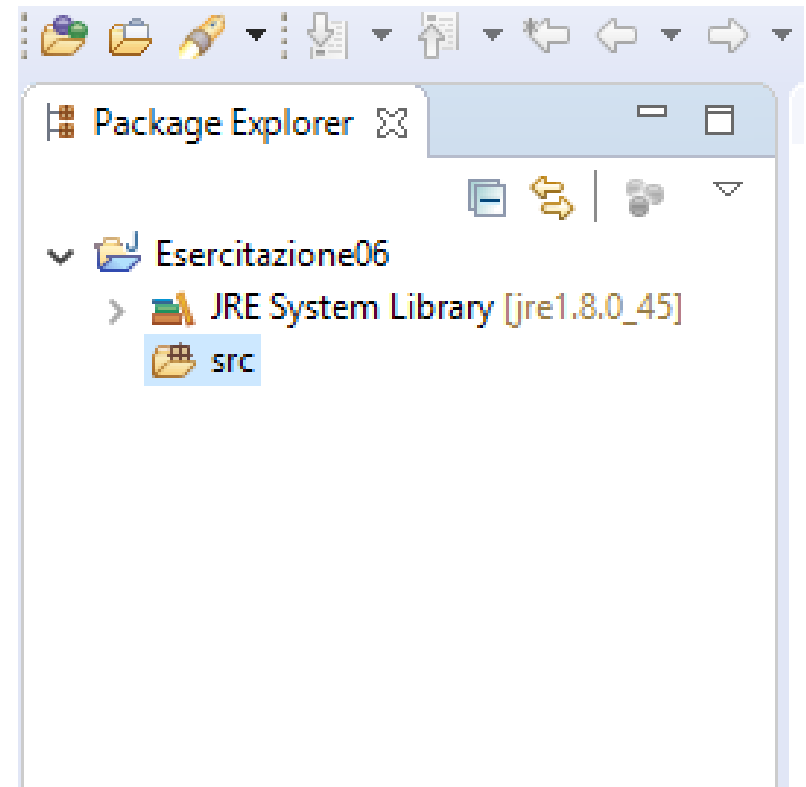
Poi click su "Finish"



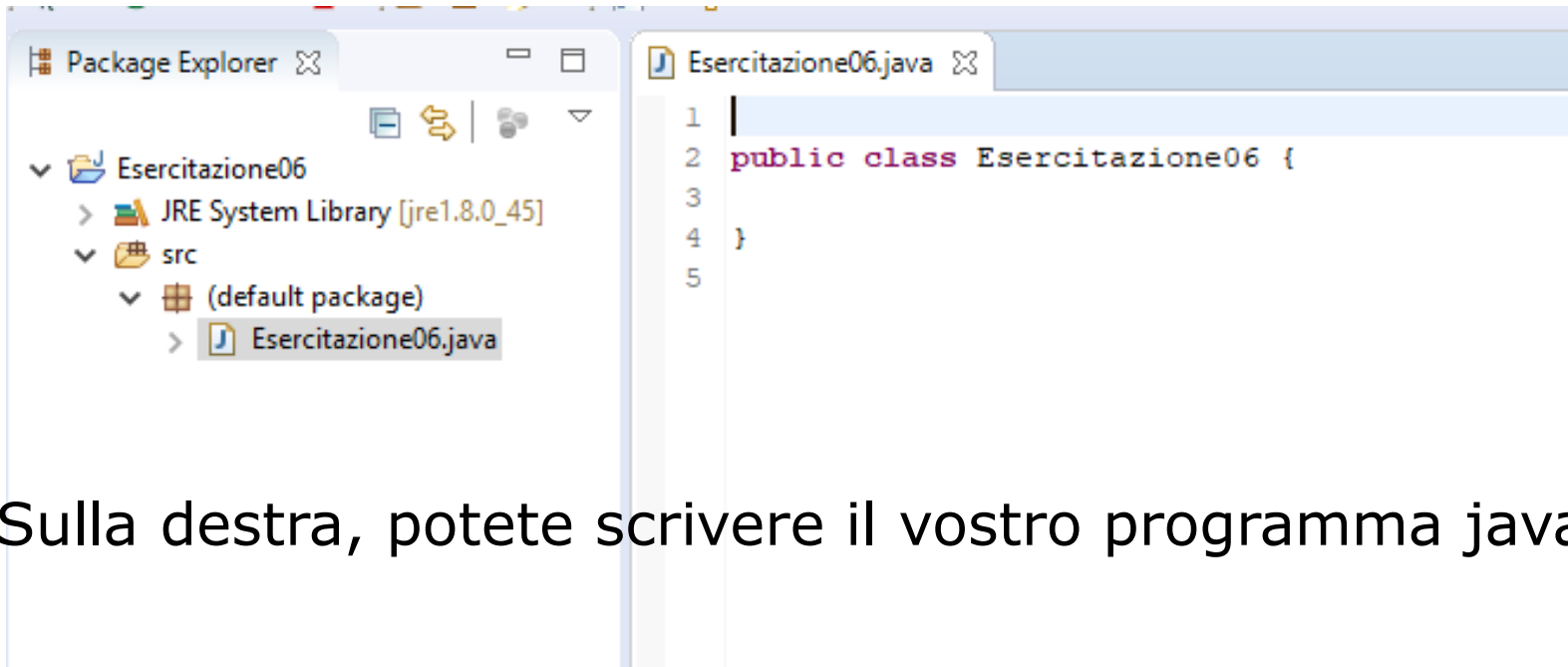


# Creazione classe

- Sulla sinistra, compare un albero di cartelle. La cartella "src" conterrà i vostri file sorgente.
- Selezionate "src" e aggiungete una nuova classe tramite:  
File → New → Class

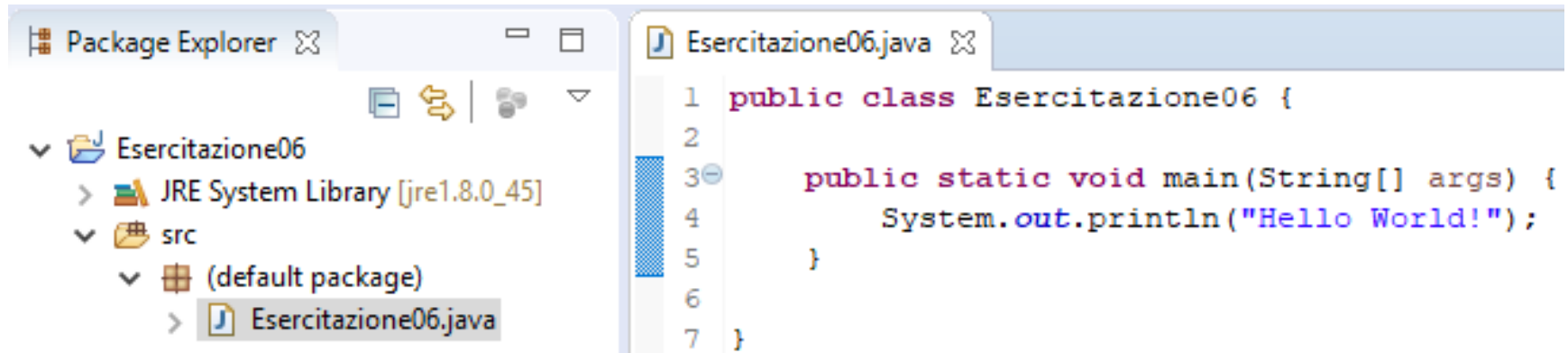


# Creazione classe



- Sulla destra, potete scrivere il vostro programma java.
- Rispetto a notepad, ci sono molti vantaggi, tra cui:
  - syntax highlighting
  - indentazione automatica (ctrl + A e poi ctrl + shift + F)
  - e molti altri...

# Hello World

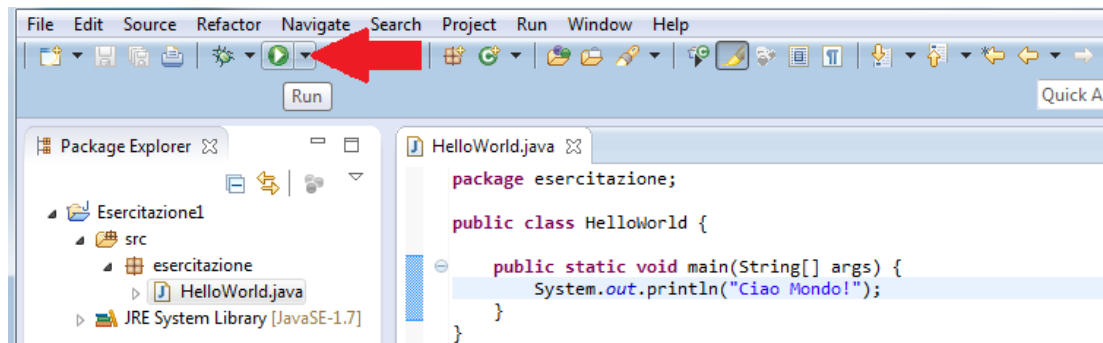


The screenshot shows an IDE interface with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project named 'Esercitazione06' with a 'src' folder containing a file 'Esercitazione06.java'. The code editor displays the following Java code:

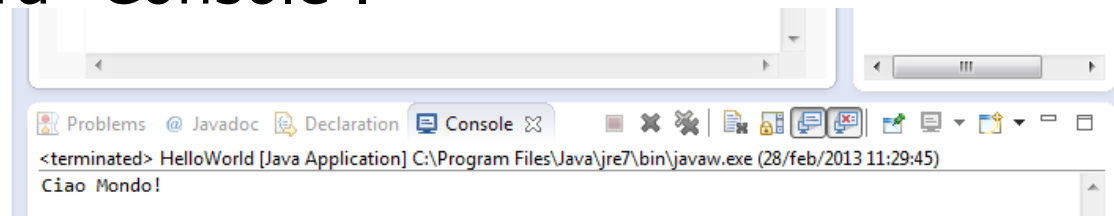
```
1 public class Esercitazione06 {  
2  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6  
7 }
```

# Compilazione ed esecuzione

- Non c'è bisogno di compilare ed eseguire il programma separatamente.
- Eclipse esegue **javac** in automatico. Esegue **java** alla pressione del tasto "play".







- L'output dell'eseguibile verrà stampato in basso nella finestra "Console".



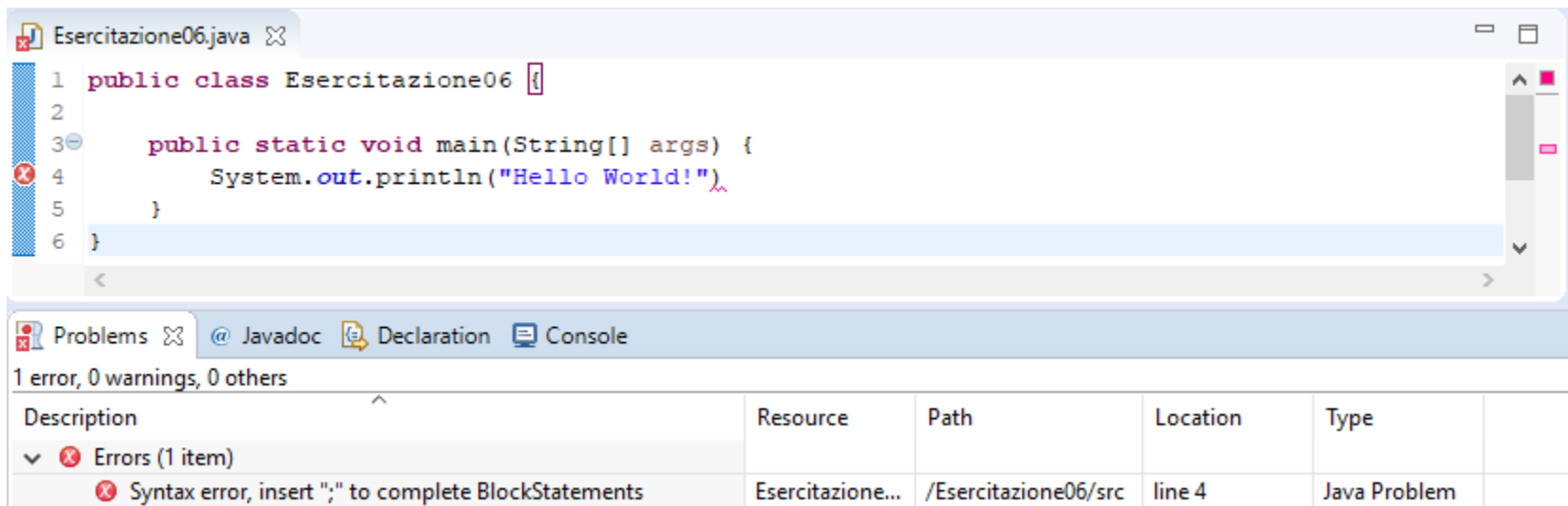
# Dove sono i miei file?

- Eclipse salva sul filesystem i file sorgenti, e anche i file `.class` che genera automaticamente.
- In questo caso, nel percorso `... \workspace\Esercitazione06` troveremo:
  - `src` contiene i file `.java` che avete scritto;
  - `bin` contiene i file `.class` che genera automaticamente: non c'è nulla di "magico"!

Name	Date modified	Type	Size
 bin	30/04/2018 16.24	File folder	
 src	30/04/2018 16.24	File folder	
 .classpath	30/04/2018 16.16	CLASSPATH File	1 KB
 .project	30/04/2018 16.16	PROJECT File	1 KB

# Eclipse: Errori

- Ogni volta che salvate una modifica, Eclipse prova a compilare automaticamente il codice.
- Se trova errori, ve li segnala. Per compilare, non devono esserci errori.



The screenshot shows the Eclipse IDE interface. The top editor window displays the code for `Esercitazione06.java`:

```
1 public class Esercitazione06 {  
2  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6 }
```

A red 'X' icon in the left margin indicates an error on line 4. The bottom of the IDE shows the **Problems** view with the following table:

Description	Resource	Path	Location	Type
1 error, 0 warnings, 0 others				
Errors (1 item)				
Syntax error, insert ";" to complete BlockStatements	Esercitazione...	/Esercitazione06/src	line 4	Java Problem

# Eclipse: Warning

- Un warning è meno grave di un errore?
  - In generale: sì, perché la compilazione può avvenire.
  - Nel nostro caso, NO: probabilmente abbiamo sbagliato qualcosa! (tranne alcuni casi particolari, che vedremo)
- Quindi è fondamentale capire e risolvere anche i warning del compilatore.

```
Esercitazione06.java ✕
1 public class Esercitazione06 {
2     public static void main(String[] args) {
3         System.out.println("Hello world!");
4         if(false)
5             System.out.println("Chi mi esegue?");
6     }
7 }
```

In questo caso Eclipse segnala che il codice che abbiamo scritto non ha speranze di essere mai eseguito.

Qui è banale, ma basta poco per complicare molto la situazione.

Problems ✕ @ Javadoc Declaration Console

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				
Dead code	Esercitazione...	/Esercitazione06/src	line 5	Java Problem

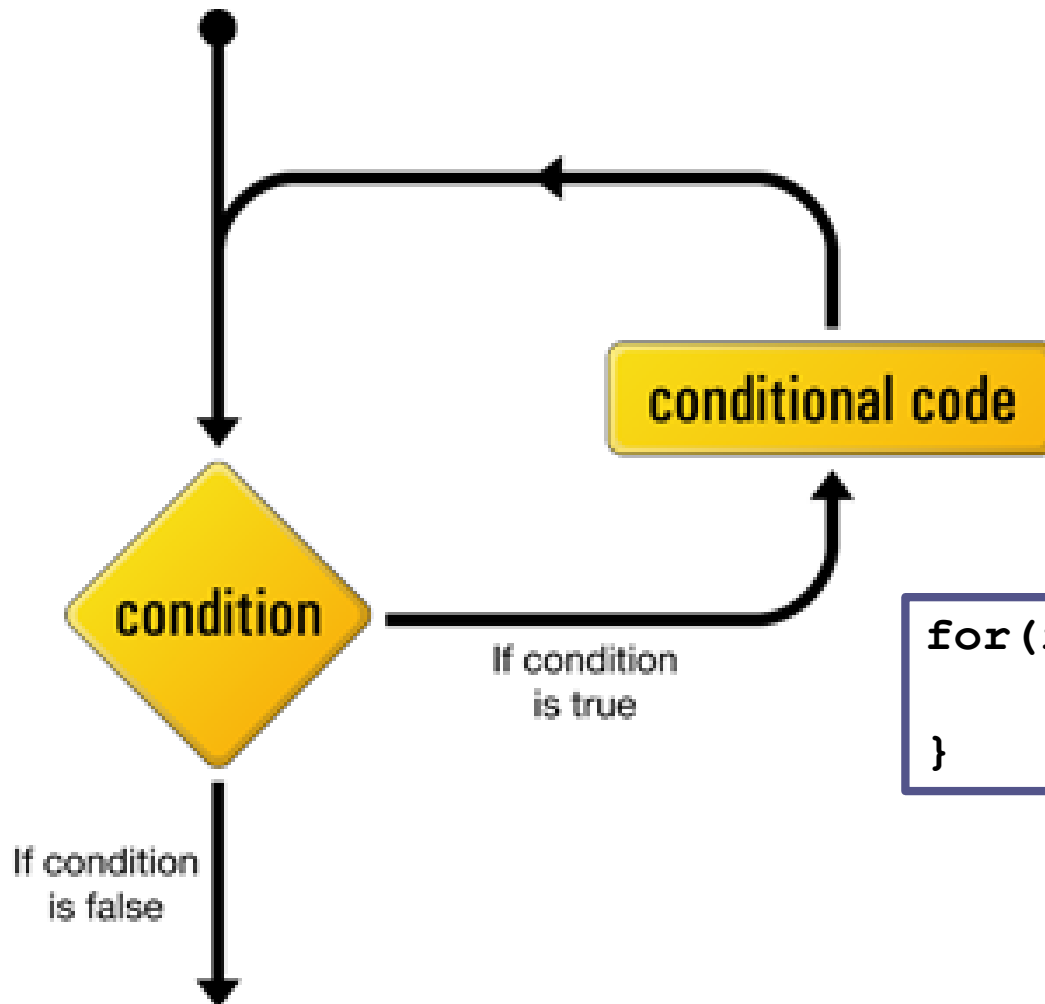
# Iterazione

Tre possibilità:

```
while( a > 5 ) {  
    ... ;  
}
```

```
do {  
    ... ;  
} while( a > 5 );
```

```
for(int i=0; i < 5; i++){  
    ... ;  
}
```





# Esercizio 1 – SommaSequenza (1/2)

- Realizzare un programma che prende in input una sequenza di cifre (tra 1 e 9) e calcola la somma massima fra le sottosequenze di cifre non decrescenti.
- Il programma termina quando viene inserito lo 0.
- Per semplicità, supponiamo di non controllare la correttezza dell'input.
- Esempio:

2	2	4	5	3	9	3	1	5	0
13				12		3	6		

```

Inserisci un intero: 2
Inserisci un intero: 2
Inserisci un intero: 4
Inserisci un intero: 5
Inserisci un intero: 3
Inserisci un intero: 9
Inserisci un intero: 3
Inserisci un intero: 1
Inserisci un intero: 5
Inserisci un intero: 0
Massima somma trovata: 13
  
```

# Esercizio 1 – SommaSequenza (2/2)

Di che valori devo tenere traccia?

Devo accorgermi di quando il valore appena letto è più piccolo del precedente. In tal caso, devo confrontare la somma corrente con quella massima, e ripartire con una nuova somma.

2	2	4	5	3	9	3	1	5	0
13				12		3	6		

```
Inserisci un intero: 2
Inserisci un intero: 2
Inserisci un intero: 4
Inserisci un intero: 5
Inserisci un intero: 3
Inserisci un intero: 9
Inserisci un intero: 3
Inserisci un intero: 1
Inserisci un intero: 5
Inserisci un intero: 0
Massima somma trovata: 13
```

# Soluzione – SommaSequenza

```
import java.util.Scanner;
public class Esercitazione06 {
    public static void main(String[] args) {
        int cur_val = 0, old_val = 0, somma = 0, somma_max = 0;
        Scanner tastiera = new Scanner(System.in);
        do {
            System.out.print("Inserisci un intero: ");
            cur_val = tastiera.nextInt();

            if(cur_val < old_val) {
                if( somma > somma_max)
                    somma_max = somma;
                somma = cur_val;
            }
            else
                somma = somma + cur_val;

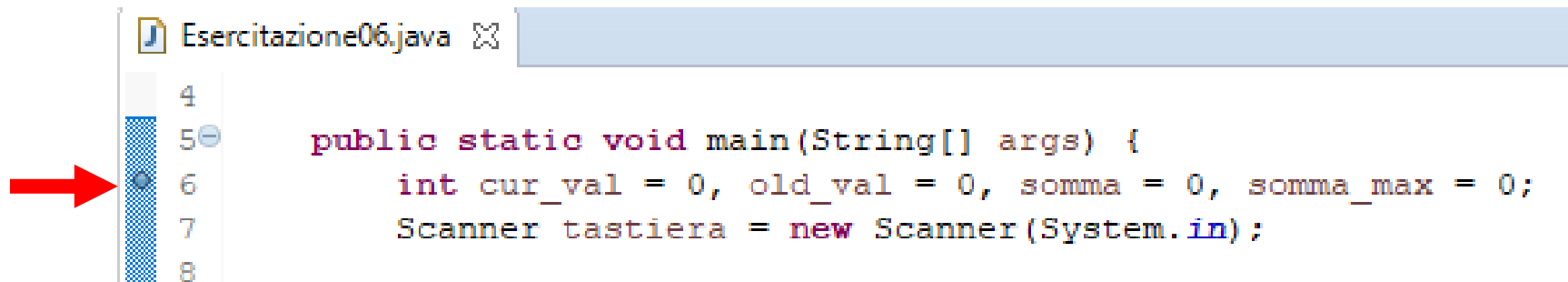
            old_val = cur_val;
        } while(cur_val != 0);

        System.out.println("Massima somma trovata: "+ somma_max);
    }
}
```

Perché questa  
parentesi?  
A chi viene riferito  
else?

# Eclipse: debugging

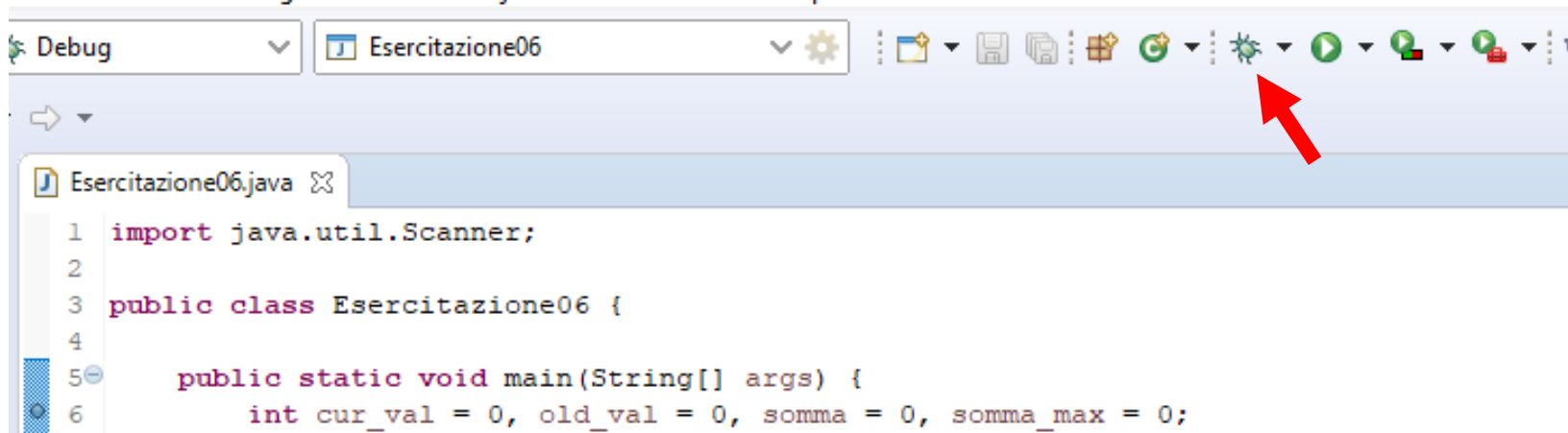
- Funzionalità estremamente utile offerta da Eclipse: consente di seguire passo passo **il flusso di esecuzione** del programma: così è molto più facile trovare errori.
- Proviamo a usare il debugger per l'esempio appena visto.
- Prima di tutto, dobbiamo scegliere da quale riga effettuare il debug e **collocare un breakpoint** in corrispondenza di essa: basta fare doppio click sull'area azzurra che corrisponde alla riga da cui si vuole partire.



```
Esercitazione06.java ✕  
4  
5  
6  
7  
8  
  
public static void main(String[] args) {  
    int cur_val = 0, old_val = 0, somma = 0, somma_max = 0;  
    Scanner tastiera = new Scanner(System.in);  
}
```

# Eclipse: debugging

- Funzionalità estremamente utile offerta da Eclipse.
- Consente di seguire passo passo l'esecuzione del programma: così è molto più facile trovare errori.
- Proviamo a usare il debugger per l'esempio appena visto.



# Ambiente di debugging

Comandi di debug (vedi prossima slide)

Prossima riga da eseguire

Coppie variabile/valore definite nell'ambiente corrente

Console per visualizzare l'output

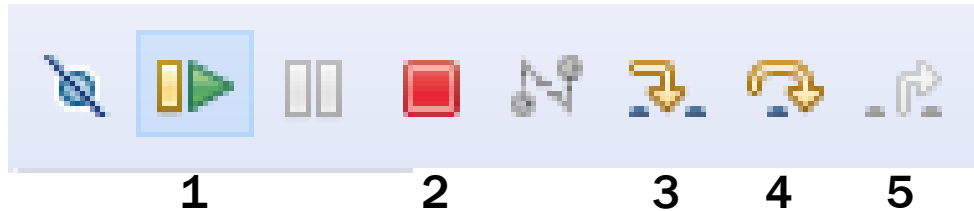
```

1 import java.util.Scanner;
2
3 public class Esercitazione06 {
4
5     public static void main(String[] args) {
6         int cur_val = 0, old_val = 0, somma = 0, somma_max = 0;
7         Scanner tastiera = new Scanner(System.in);
8
9         do {
10            //Lettura input
11            System.out.print("Inserisci un intero: ");
12            cur_val = tastiera.nextInt();
13
14            if(cur_val < old_val) {
15                if( somma > somma_max)
16                    somma_max = somma;
17                somma = cur_val;
18            } //Perché ci va questa parentesi? A chi sarebbe riferito l'else senza di essa?
19            else
  
```

Name	Value
no method return value	
args	String[0] (id=16)

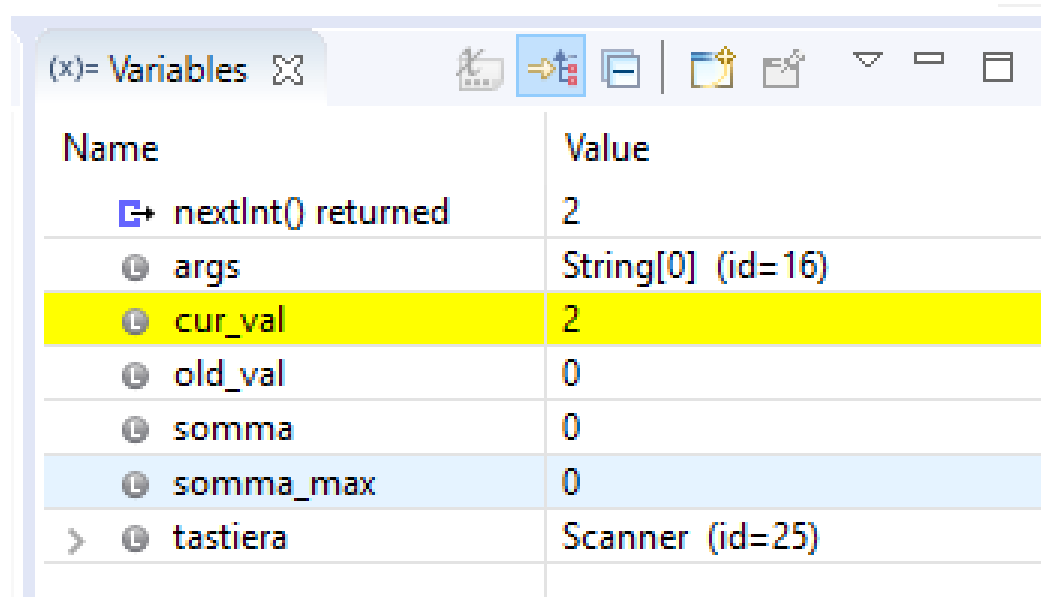
Console per visualizzare l'output

# Ambiente di debugging



1. "Play". Esegue direttamente tutti i passi fino al breakpoint successivo (se non c'è, fino alla fine).
2. "Stop". Termina l'esecuzione del programma immediatamente
3. "Step into". Passa ad eseguire la riga successiva; se la riga corrente prevede l'invocazione di un metodo, prosegue il debug sul codice del metodo.
4. "Step over". Passa ad eseguire la riga successiva.
5. "Step return". Riprende l'esecuzione dalla riga successiva a quella che ha portato all'invocazione del metodo usato correntemente.

# Eclipse: debugging



Name	Value
nextInt() returned	2
args	String[0] (id=16)
cur_val	2
old_val	0
somma	0
somma_max	0
tastiera	Scanner (id=25)

- Quando una variabile assume un nuovo valore, si evidenzia la riga corrispondente nella vista "Variabili".
- Molto utile per controllare l'esecuzione di cicli: si può vedere l'evoluzione di indici e accumulatori.



## Esercizio 2 – Formula (1/2)

- Si realizzi un programma che legga un intero N da tastiera, e stampi a video il risultato della seguente sommatoria:

$$\sum_{i=0}^N \left[ (-1)^i \frac{4}{2 * i + 1} \right]$$

Cosa ottengo con questa formula?

- Una volta calcolato e stampato il valore a video, il programma deve chiedere un nuovo numero all'utente e ripetere il calcolo.
- Il programma deve terminare solo qualora l'utente inserisca un valore negativo.

## Esercizio 2 – Formula (2/2)

$$\sum_{i=0}^N \left[ (-1)^i \frac{4}{2 * i + 1} \right]$$

Cosa ottengo con questa formula?

Provare a vedere cosa accade aumentando progressivamente il valore di N (10, 100, ..., 10000)

- dal punto di vista del risultato (come cambia?)
- dal punto di vista del tempo impiegato (risponde sempre subito?)

Qual è il valore massimo che potete inserire? Perché?

## Esercizio 3 – Fattoriali

- Scrivere un programma che permetta di:
  - Chiedere all'utente quanti numeri vuole inserire
  - Leggere i numeri inseriti dall'utente e calcolare la somma dei fattoriali
- Esempio: l'utente vuole inserire 3 numeri: 4, 3, 6
  - Il programma deve calcolare  $4! + 3! + 6! = 750$
- Seguire l'esecuzione del programma con l'uso del debugger di Eclipse.

## Esercizio 4 – Sequenze0e1

- Realizzare un programma che prende in input una sequenza di caratteri 0 e 1 e conta la lunghezza della più lunga sotto-sequenza di 0 di fila.
- L'inserimento della sequenza termina quando si inserisce un carattere diverso da 0 e 1. A quel punto, si stampa a video il valore trovato.
- Seguire l'esecuzione del programma con l'uso del debugger di Eclipse.

## Esercizio 5 – ValoriPositivi

Si scriva un programma che legga da utente una sequenza di al massimo 10 valori positivi (si scartino i negativi). L'utente può terminare prima inserendo lo 0.

Il programma deve stampare:

1. il numero di valori positivi letti
2. il numero di valori scartati
3. la somma di tutti i valori positivi
4. L'elemento maggiore della sequenza dei positivi
5. L'elemento minore della sequenza dei positivi

Seguire l'esecuzione del programma con l'uso del debugger di Eclipse.