

## Fondamenti di Informatica T2

### Introduzione a Eclipse

*Corso di Laurea in Ingegneria Informatica*

Anno accademico 2021/2022

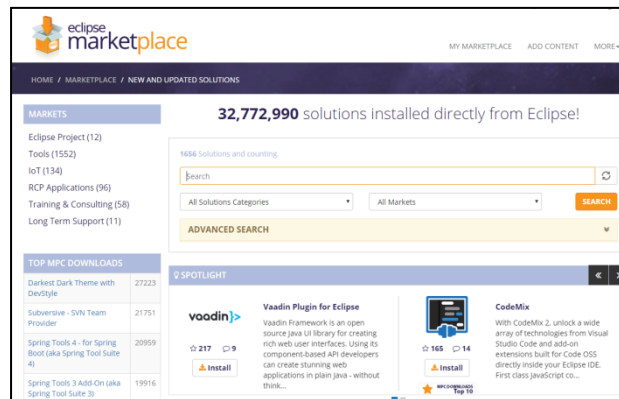
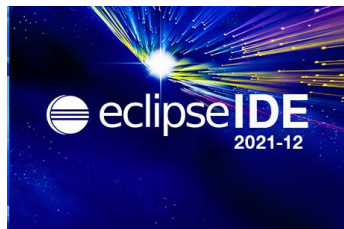
Prof. ROBERTA CALEGARI

Prof. AMBRA MOLESINI

*Dipartimento di Informatica – Scienza e Ingegneria (DISI)*

# ECLIPSE (1)

- Eclipse è un ambiente di sviluppo:
  - **opensource**: è scritto esso stesso in Java
  - **multiplatforma**: Windows, Mac, Linux,...
  - **multilinguaggio**: non serve solo per Java
  - **estendibile** tramite *plugin*
- ne esistono *migliaia* nel *MarketPlace*  
<http://marketplace.eclipse.org>



| MARKETS  |
|--|
| Eclipse Project (12)                               |
| <u>Tools (1552)</u>                                |
| Application Development Frameworks (188)           |
| Application Management (37)                        |
| Application Server (41)                            |
| BIRT (13)  |
| Build and Deploy (127)                             |
| Business Intelligence, Reporting and Charting (28) |
| Code Management (151)                              |
| Collaboration (47)                                 |
| Database (54)                                      |
| Database Development (41)                          |
| Database Persistence (23)                          |
| Documentation (80)                                 |
| Eclipse Kura (78)                                  |
| Eclipse SmartHome (54)                             |
| EclipseRT Target Platform Components (8)           |
| Editor (379)                                       |
| Entertainment (18)                                 |
| General Purpose Tools (131)                        |
| Graphics (41)                                      |
| IDE (395)  |
| Internet of Things (IoT) (33)                      |
| J2EE Development Platform (57)                     |
| J2ME (6)   |
| Languages (182)                                    |
| Linux Tools (18)                                   |
| Logging (26)                                       |
| Mobile and Device Development (58)                 |
| Modeling (112)                                     |
| Modeling Tools (184)                               |
| Mylyn Connectors (24)                              |
| Network (18)                                       |
| Other (83)   |
| Process (23)                                       |

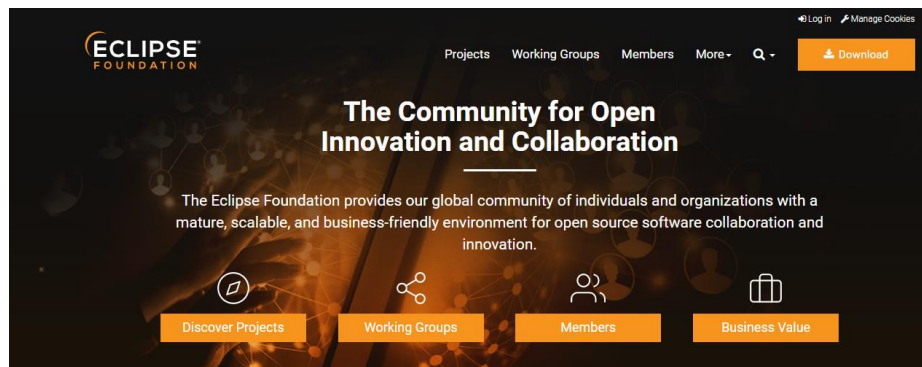


# ECLIPSE (2)

- Eclipse è esso stesso un'applicazione Java
  - parte utilizzando il JDK/JRE impostato nel path di sistema
- Possiede il suo **compilatore autonomo**
  - non utilizza quello del JDK
  - può compilare *anche per versioni precedenti* di Java
- Richiede però necessariamente un **JRE/JDK esterno** per far **girare le applicazioni**
  - il default è la versione di Java (JDK o JRE) con cui è partito
  - ma può usare anche altre versioni di Java presenti sul PC

# ECLIPSE (3)

- Attualmente viene rilasciata una nuova versione in simultanea con l'uscita delle nuove versioni di Java
  - in passato non era così
- Attualmente le versioni si chiamano *anno-mese*
  - ad esempio, 2021-12
  - in passato abbiamo avuto
    - astri, pianeti e lune varie:  
Callisto, Europa, Ganymede, Galileo, Helios, Indigo, Juno, Kepler, Luna, Mars
    - gas e altri nomi scientifici:  
Neon, Oxygen, Photon



# La Giusta Versione

- Eclipse è una piattaforma configurabile
  - arbitrariamente arricchibile con *plugin*
  - al download sono perciò proposte le *configurazioni di più largo uso*
- Voi volete la  
***Eclipse IDE for Java Developers***

## Eclipse IDE 2019-12 R Packages

### Eclipse IDE for Enterprise Java Developers

353 MB 359,883 DOWNLOADS



Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and more.

[Click here to file a bug against Eclipse Web Tools Platform.](#)

[Click here to file a bug against Eclipse Platform.](#)

[Click here to file a bug against Maven integration for web projects.](#)

### Eclipse IDE for Java Developers



201 MB 205,769 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

### Eclipse IDE for C/C++ Developers



237 MB 60,012 DOWNLOADS

An IDE for C/C++ developers with Mylyn integration.

### Eclipse IDE for Eclipse Committers

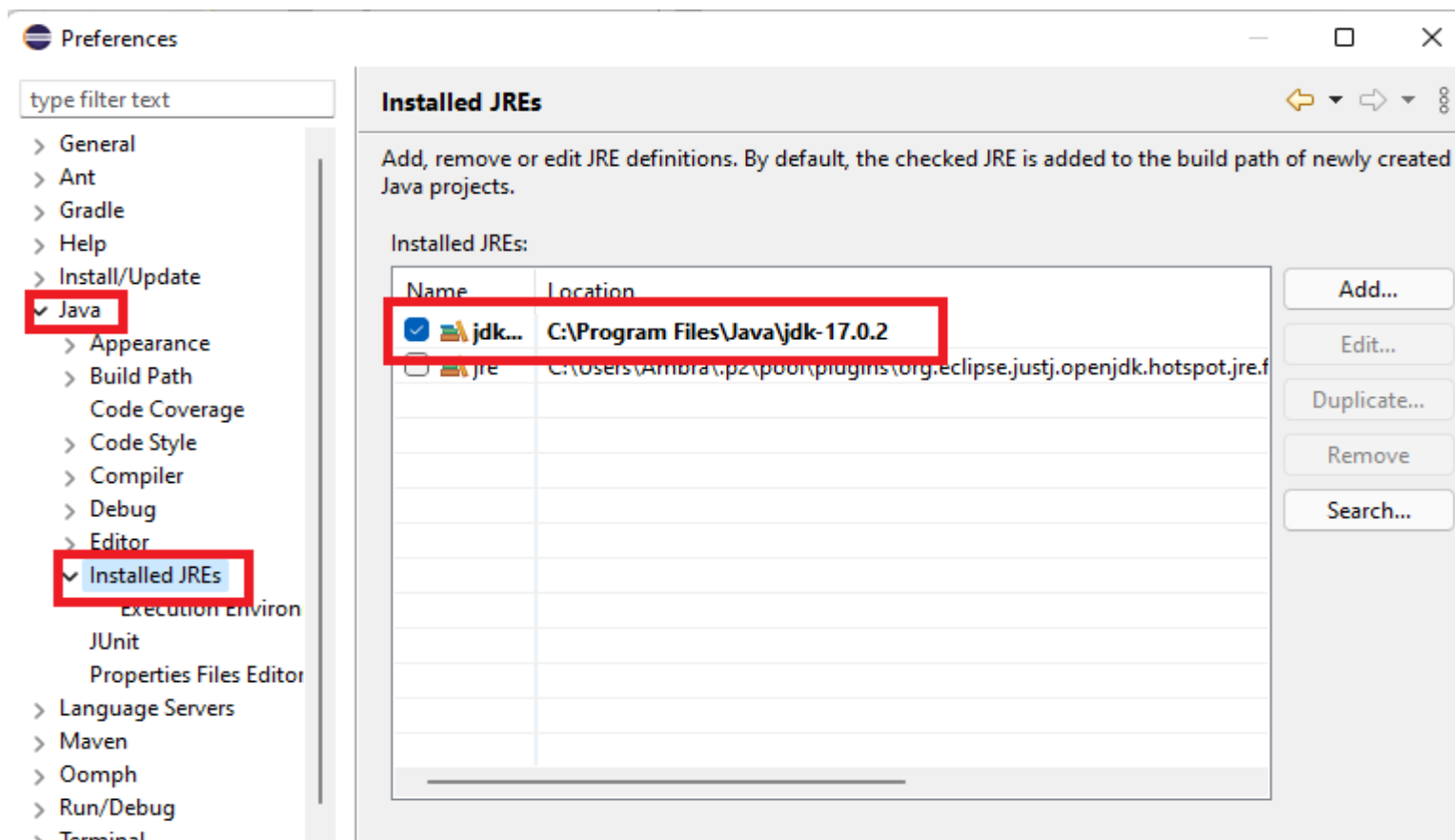


# Installazione (1)

- Easy way: scaricare l'installer ed installare  
*Eclipse IDE for Java Developers*
- Oppure, per chi ama il fai da te:
  1. Scaricare Eclipse IDE for Java Developers dal sito [www.eclipse.org](http://www.eclipse.org)
    - prestare attenzione alla piattaforma utilizzata:  
un JDK a 32 bit richiede Eclipse a 32 bit,  
un JDK a 64 bit richiede Eclipse a 64 bit,
  2. Decomprimere in una cartella a scelta
    - meglio evitare C:\Programmi (questioni di diritti..)
    - **decomprimere = installare:** Eclipse NON scrive nulla nel registro di Windows, non semina file nel vostro disco: è tutto contenuto nella cartella decompressa
    - **opportuno creare a mano un collegamento sul desktop e/o nel menù Start**

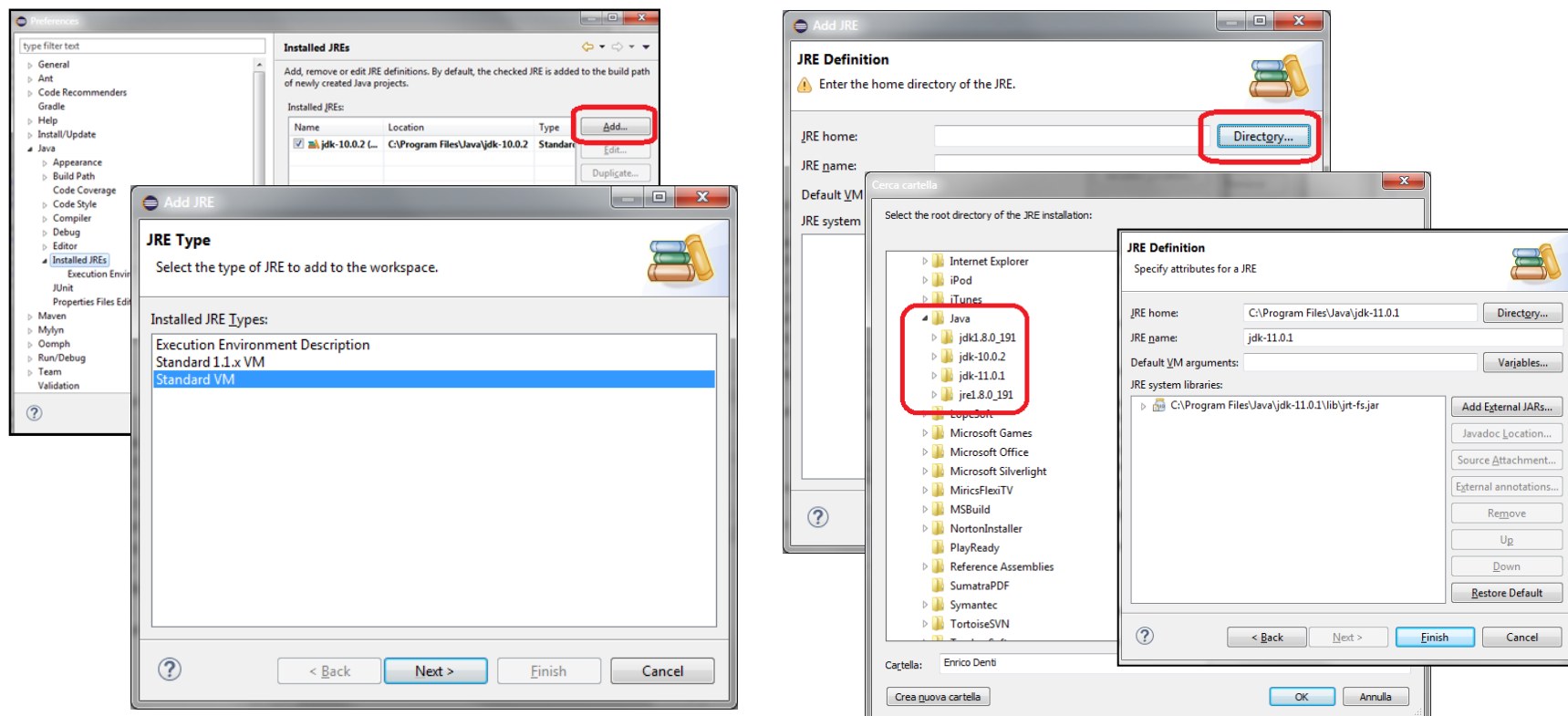
# Installazione (2)

- Per verificare a quale versioni di Java si è agganciato:  
> Window > Preferences > Java > Installed JREs



# Installazione (3)

- Se sul pc sono presenti altri JDK/JRE che si intende usare, si possono aggiungere:







# CONCETTI CHIAVE

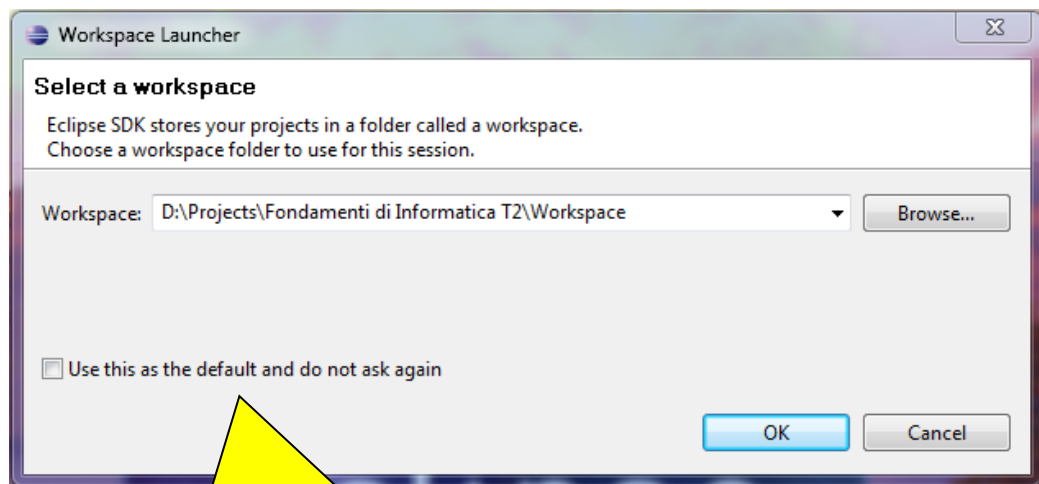
- Eclipse si basa su alcuni concetti chiave:
  - **workspace** lo spazio di lavoro
  - **progetto** l'entità concettuale su cui si lavora
- a loro volta strutturati in:
  - workspace → *prospettive, viste, editor, toolbar*
  - progetto → *file sorgenti, percorsi, cartelle, test, librerie e documentazione*
- Fin dall'avvio vi chiederà in che *workspace* lavorare
  - ne potete avere molti
  - organizzate con intelligenza il vostro lavoro per argomenti

# WORKSPACE

- Dal punto di vista LOGICO, un **workspace** è *l'ambiente di lavoro* in cui vengono creati e mantenuti i progetti
  - possiamo pensare al workspace come ad una raccolta di progetti
- Dal punto di vista FISICO, un **workspace** è una *cartella* in cui sono mantenute *le impostazioni e le caratteristiche* del workspace
  - ogni workspace ha le sue impostazioni *anche a livello Eclipse*
  - workspace diversi possono usare compilatori diversi, essere configurati in modo diverso, avere proprietà diverse, etc
  - i progetti inclusi logicamente nel workspace possono anche non risiedere in questa cartella: l'appartenenza di un progetto a un workspace è un fatto *LOGICO*, non fisico
  - perciò, un progetto può appartenere anche a più workspace

# ECLIPSE: PRIMO AVVIO (1)

Ad ogni avvio, Eclipse vi chiederà in che workspace volete lavorare:

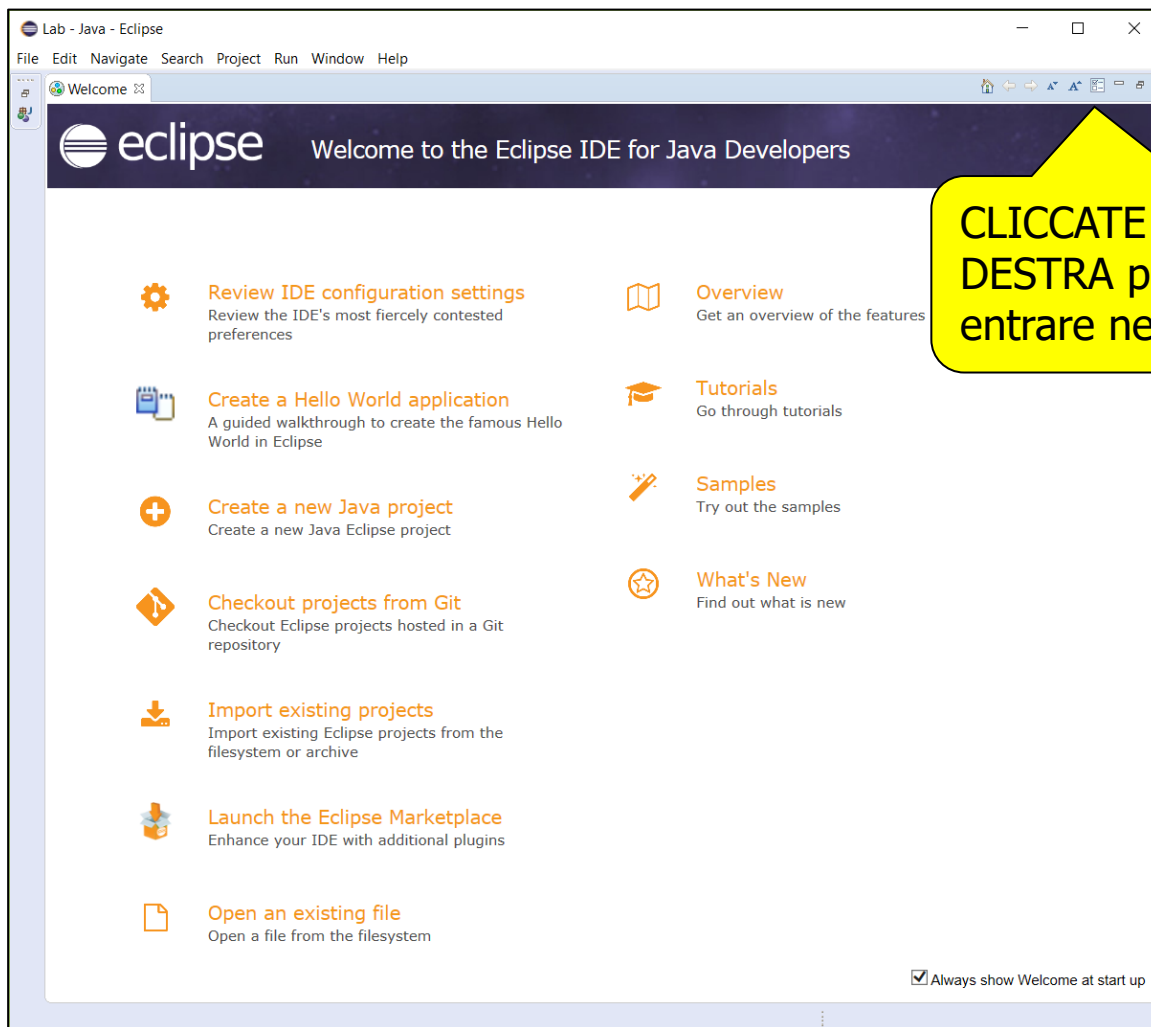


Impostare la cartella su cui salvare il workspace.

**SOLO SE PENSATE DI USARE UN SOLO WORKSPACE, impostare questa scelta come default.** Al prossimo avvio non verrà più mostrata questa schermata (sarà comunque possibile cambiare workspace da menù).



# ECLIPSE: PRIMO AVVIO (2)



CLICcate L'ICONA A DESTRA per andare oltre ed entrare nell'IDE.

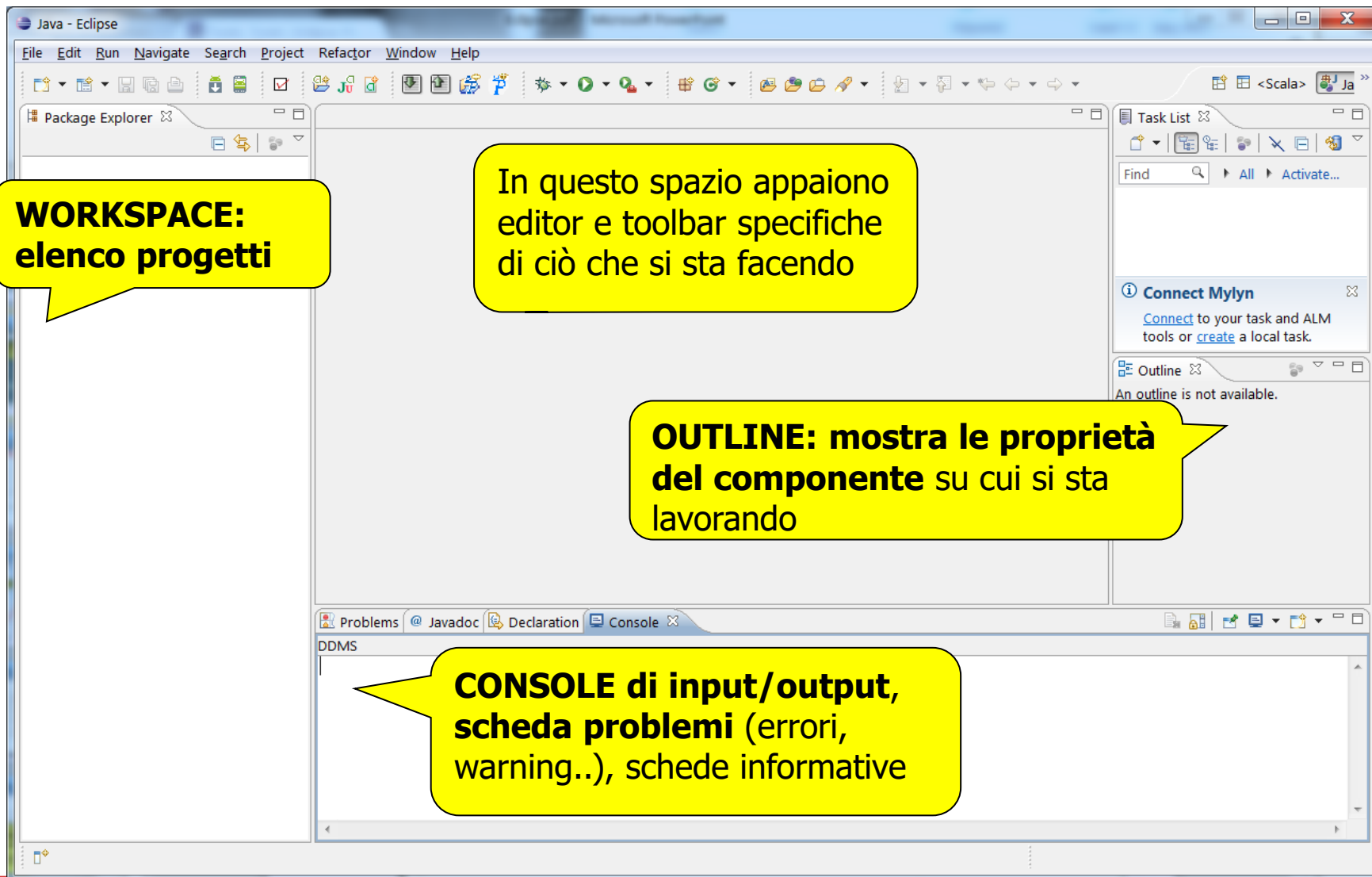
# ECLIPSE: PRIMO AVVIO (3)

**WORKSPACE:**  
elenco progetti

In questo spazio appaiono  
editor e toolbar specifiche  
di ciò che si sta facendo

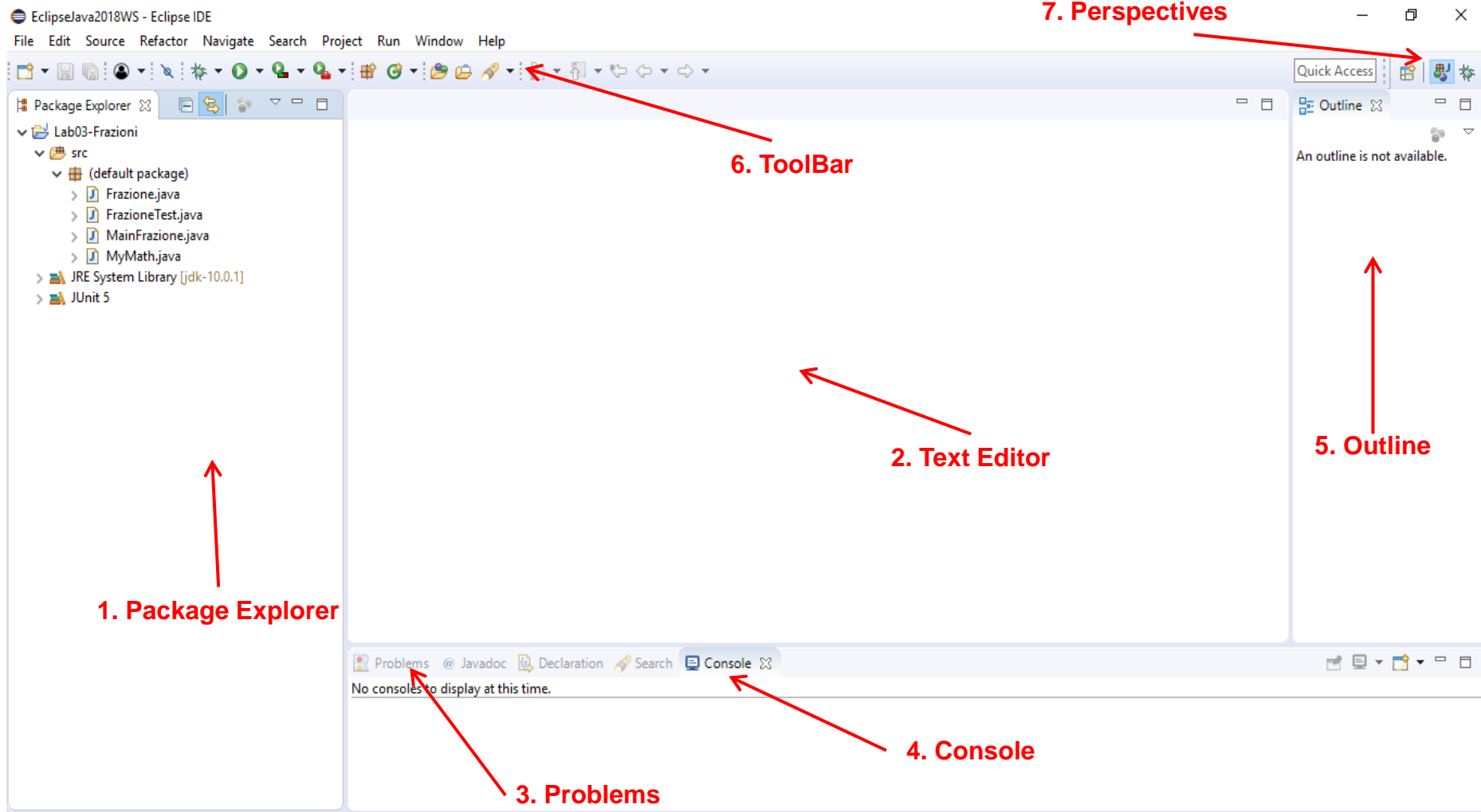
**OUTLINE:** mostra le proprietà  
del componente su cui si sta  
lavorando

**CONSOLE** di input/output,  
scheda problemi (errori,  
warning..), schede informative





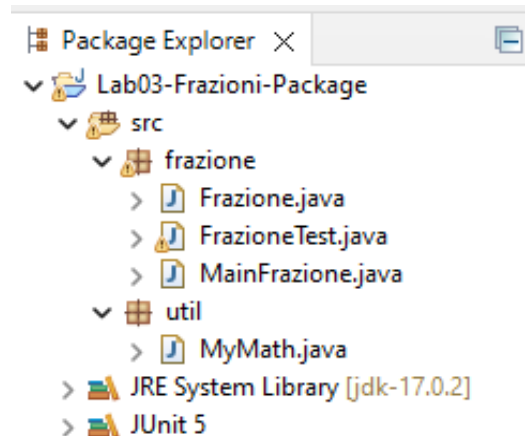
# AMBIENTE DI LAVORO (1)



# AMBIENTE DI LAVORO (2)

## 1. Package Explorer

- Permette di **visualizzare i progetti** all'interno del workspace (tipicamente, un progetto è organizzato in **package**)
- Visualizza le **dipendenze** ad altre **librerie** (ad esempio, dall'infrastruttura base: JRE System Library)
- **i sorgenti di un progetto sono inseriti nella cartella src**



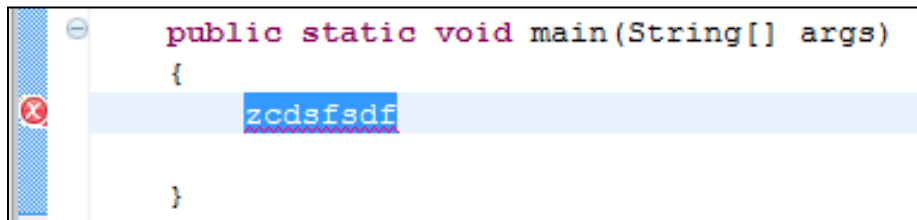
# AMBIENTE DI LAVORO (3)

## 2. Text Editor

- Permette di scrivere il codice sorgente
- evidenzia con colori differenti le parole chiave e le diverse sezioni del codice

```
/**  
 * @param args  
 */  
public static void main(String[] args)
```

- Eclipse ri-compila automaticamente i sorgenti a ogni modifica, evidenziando eventuali errori

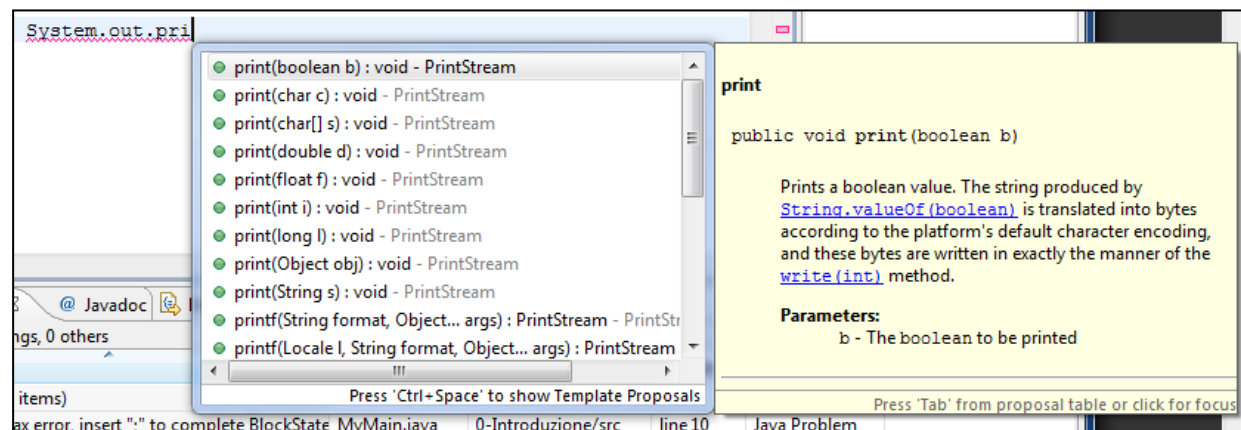


```
public static void main(String[] args)  
{  
    zcdsf sdf  
}
```



# AMBIENTE DI LAVORO (4)

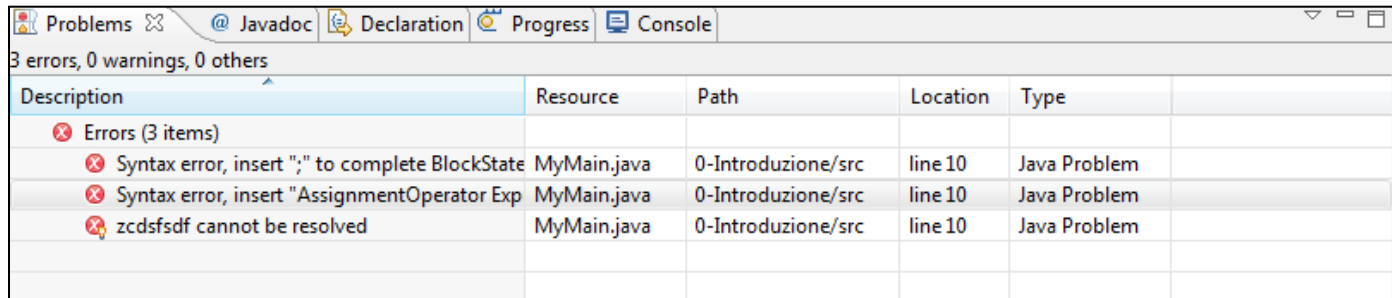
- In caso di errori/warning, cliccando sull'icona rossa/gialla sul lato sinistro del codice in errore/warning viene visualizzato un elenco di probabili cause e relative soluzioni (NON SEMPRE ESATTE...)
- È presente un supporto online che evidenzia i metodi di una classe  
Per forzare l'auto completamento → CTRL+SPACE  
*Non diventate dipendenti dall'autocompletamento! Siate autonomi!*
- L'help del metodo selezionato è mostrato nel riquadro



# AMBIENTE DI LAVORO (5)

## 3. Problems

- Visualizza eventuali **warning (GIALLI)** ed **errori (ROSSI)** di compilazione



The screenshot shows the Eclipse IDE's 'Problems' window. The title bar includes icons for Problems, Javadoc, Declaration, Progress, and Console. The main area shows '3 errors, 0 warnings, 0 others'. Below this is a table with columns: Description, Resource, Path, Location, and Type.

| Description                                     | Resource    | Path               | Location | Type         |
|---|-------------|--------------------|----------|--------------|
| Errors (3 items)                                |             |                    |          |              |
| Syntax error, insert ";" to complete BlockState | MyMain.java | 0-Introduzione/src | line 10  | Java Problem |
| Syntax error, insert "AssignmentOperator Exp    | MyMain.java | 0-Introduzione/src | line 10  | Java Problem |
| zcdfsdf cannot be resolved                      | MyMain.java | 0-Introduzione/src | line 10  | Java Problem |

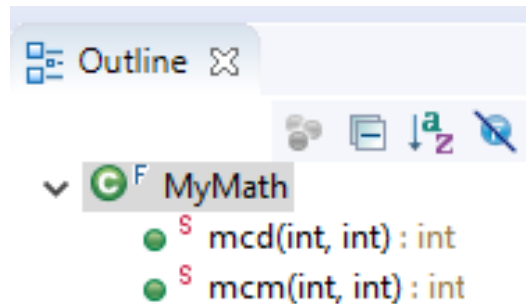
## 4. Console

- Incapsula la console dell'applicazione:  
standard output, standard input, standard error

# AMBIENTE DI LAVORO (6)

## 5. Outline

- Visualizza la struttura della **classe attualmente visualizzata** nell'editor, attraverso una struttura ad albero



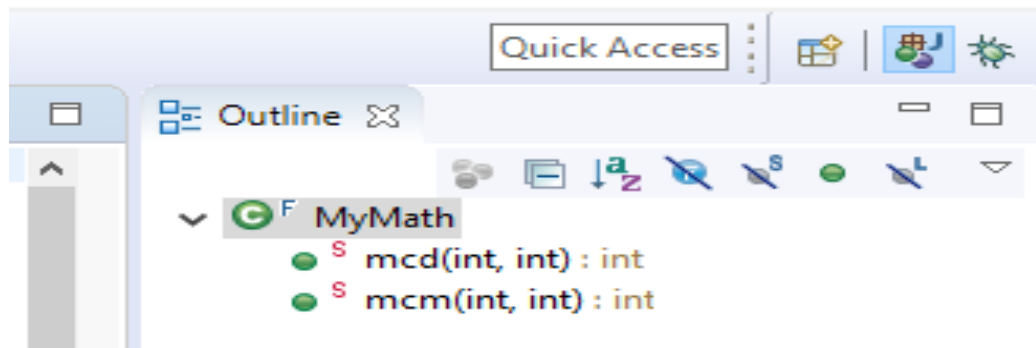
## 5. ToolBar

- È la classica barra con le operazioni di uso più comune (Apri, Salva, Compila, Esegui, Debug...) personalizzabile

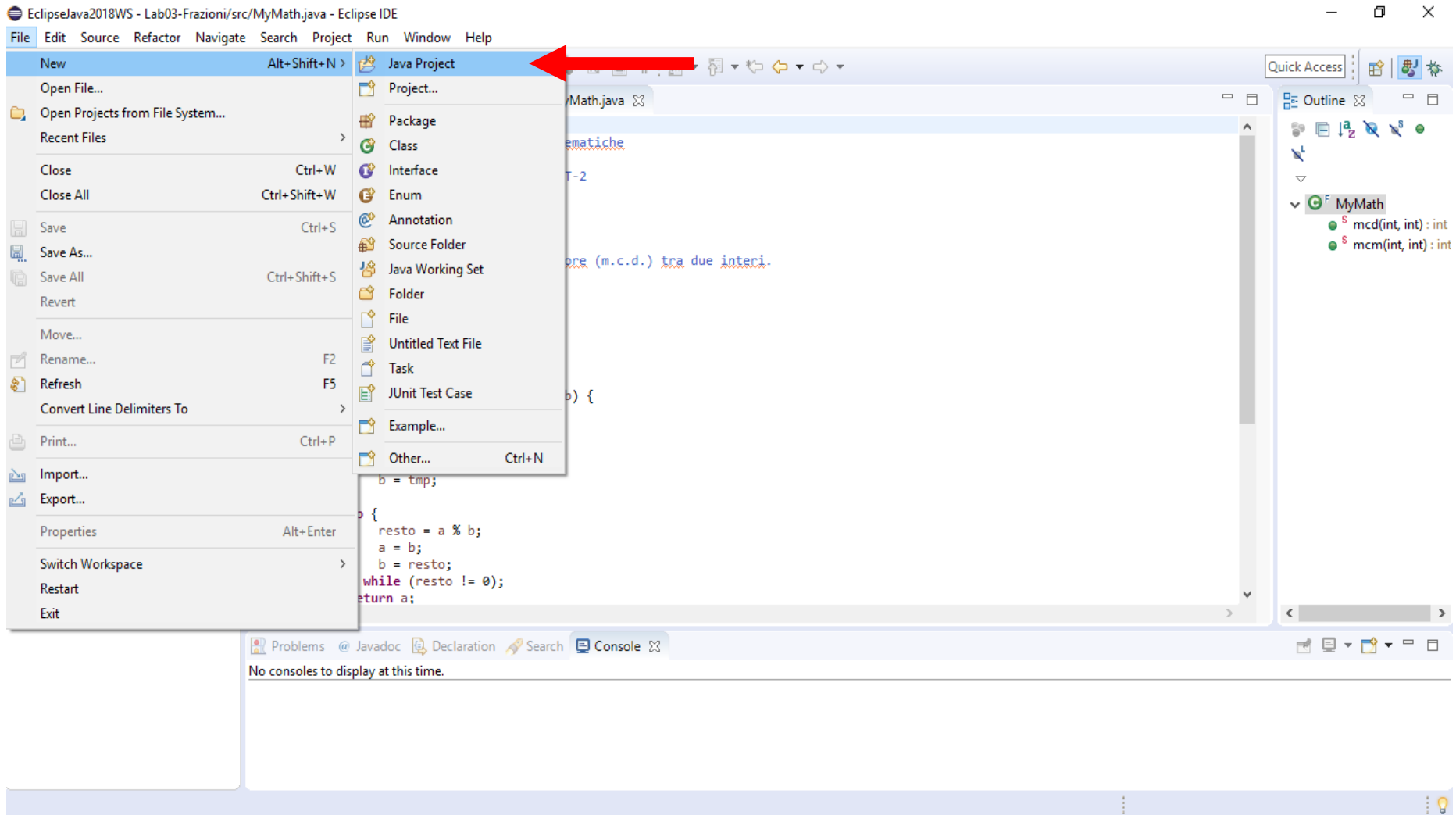
# AMBIENTE DI LAVORO (7)

## 7. Perspectives

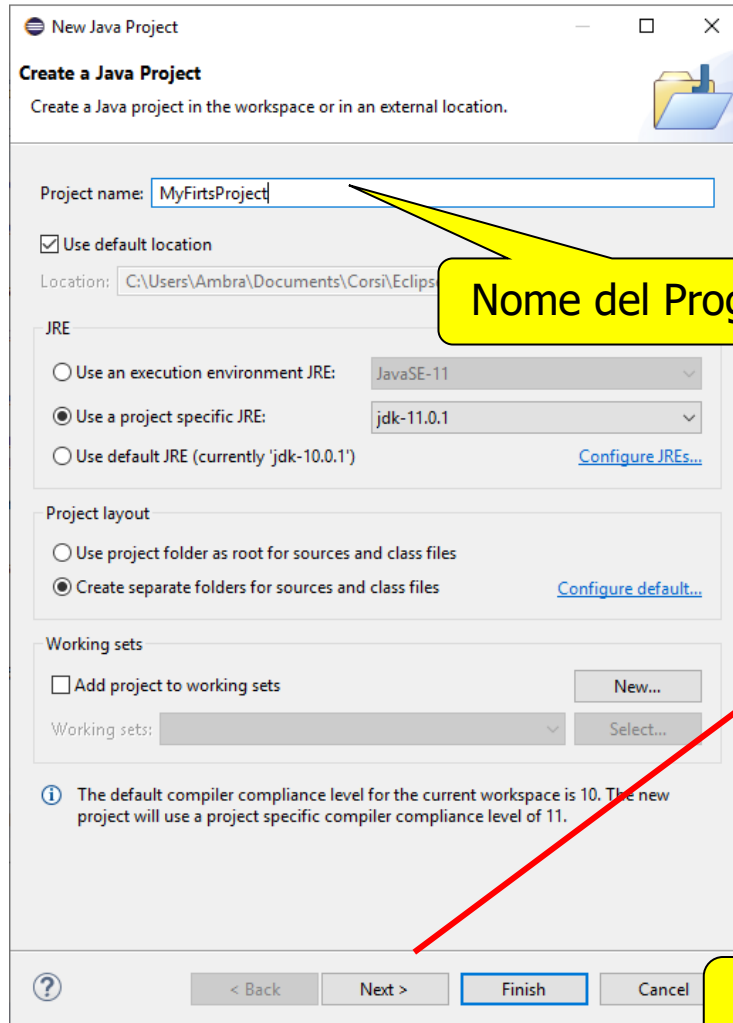
- Eclipse prevede più *prospettive di visualizzazione*
- Ogni prospettiva ha le sue viste e i suoi editor
  - quella mostrata finora è la prospettiva di programmazione
  - *ne esistono altre: ad esempio, quella di debug,* che mostra finestre e strumenti appositi per il debug
  - altre ancora dipendono dai plugin installati



# NUOVO PROGETTO (1)



# NUOVO PROGETTO (2)



**New Java Project**

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:

JRE

☐ Use an execution environment JRE:

☒ Use a project specific JRE:

☐ Use default JRE (currently 'jdk-10.0.1')

[Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

[Configure default...](#)

Working sets

☐ Add project to working sets

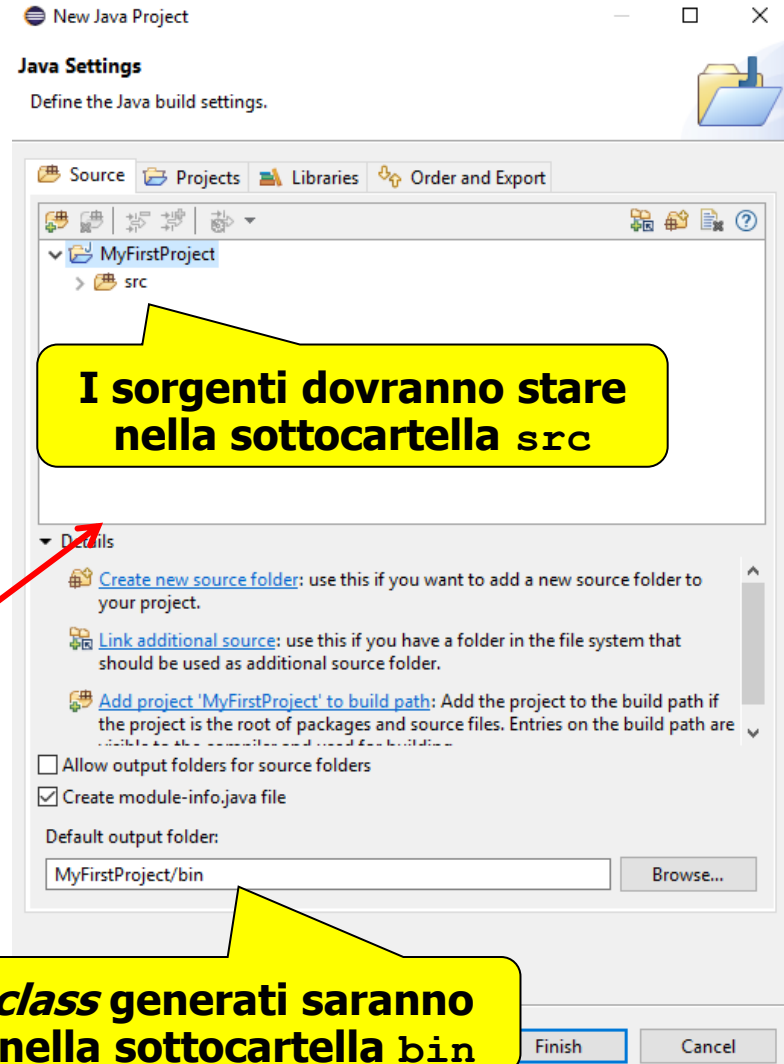
Working sets:

1 The default compiler compliance level for the current workspace is 10. The new project will use a project specific compiler compliance level of 11.

Nome del Progetto

I sorgenti dovranno stare nella sottocartella `src`

I file `.class` generati saranno messi nella sottocartella `bin`



**New Java Project**

Define the Java build settings.

Source Projects Libraries Order and Export

MyFirstProject

- src

Details

- [Create new source folder:](#) use this if you want to add a new source folder to your project.
- [Link additional source:](#) use this if you have a folder in the file system that should be used as additional source folder.
- [Add project 'MyFirstProject' to build path:](#) Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible in the **Project Explorer** and used for building.

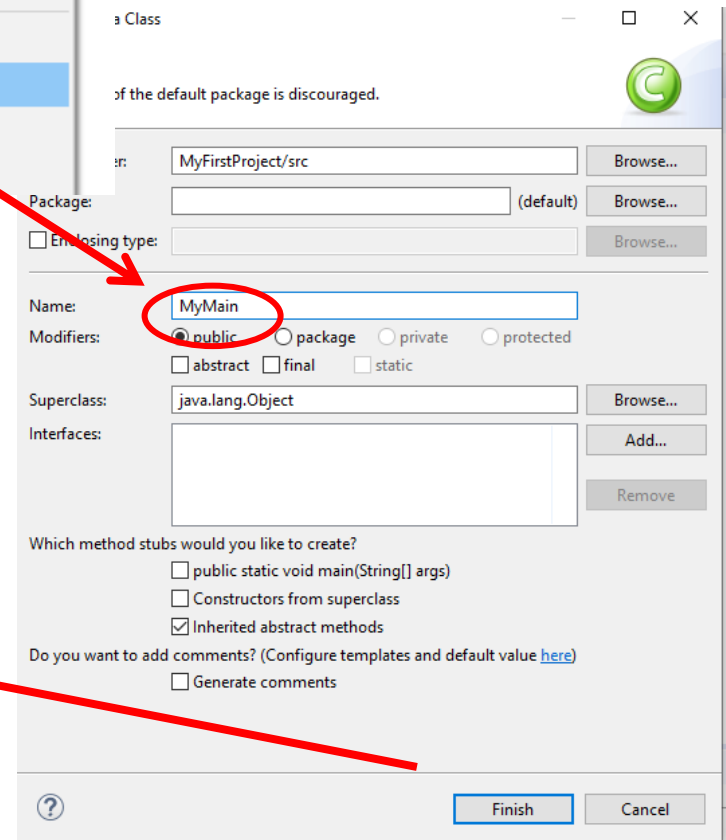
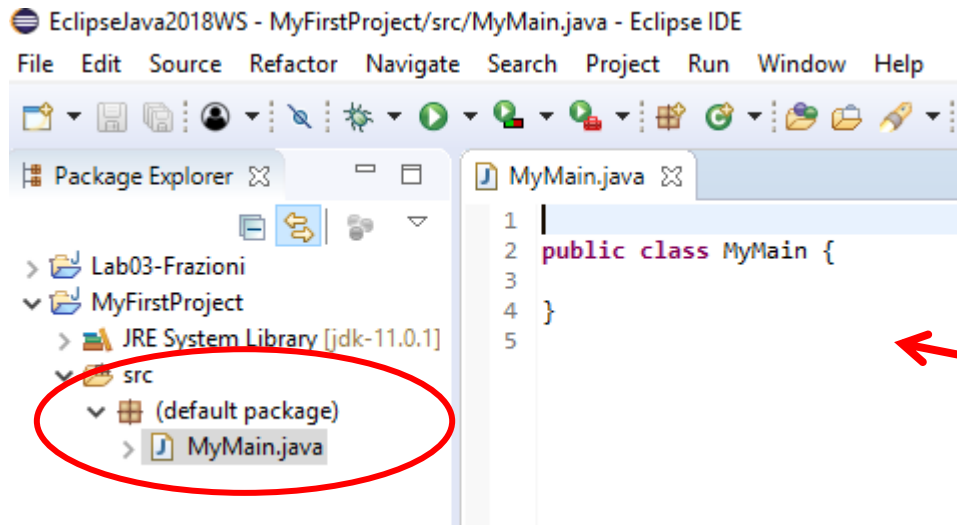
☐ Allow output folders for source folders

☒ Create module-info.java file

Default output folder:

# IL CASO BASE (senza package)

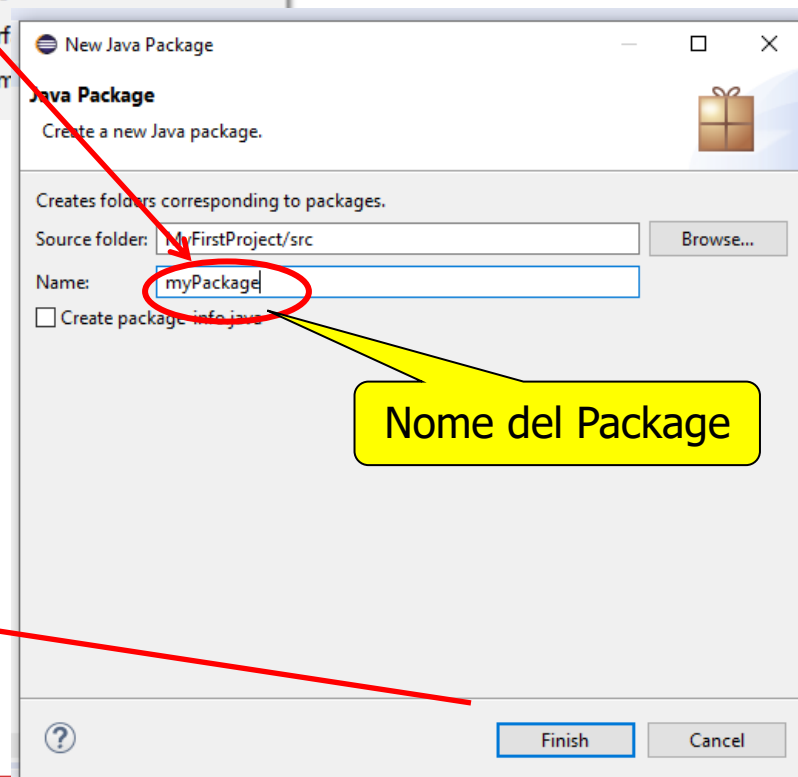
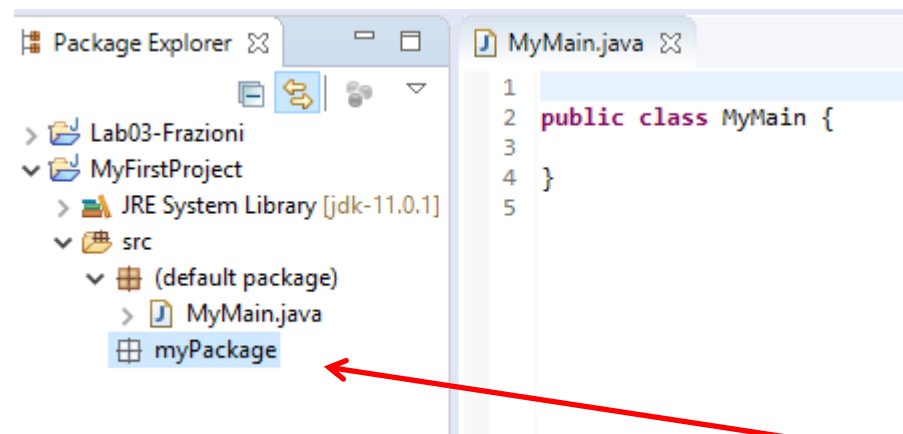
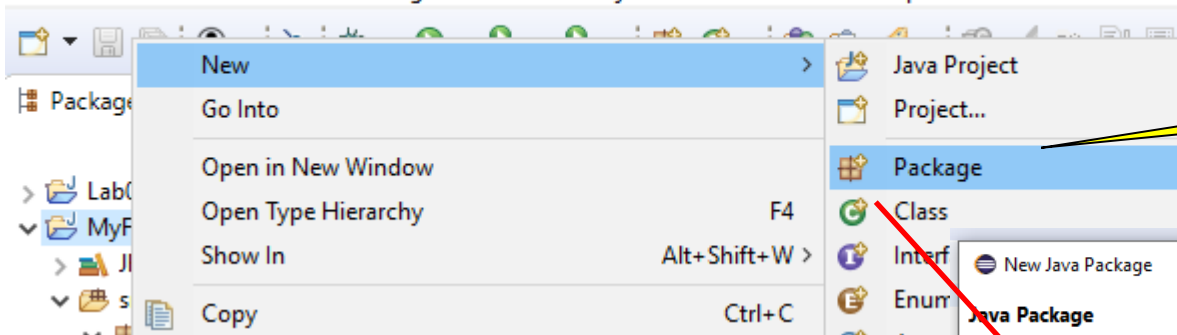
Click col tasto destro su **src**  
Scegliere **New > Class**



# IL CASO CON PACKAGE (1)

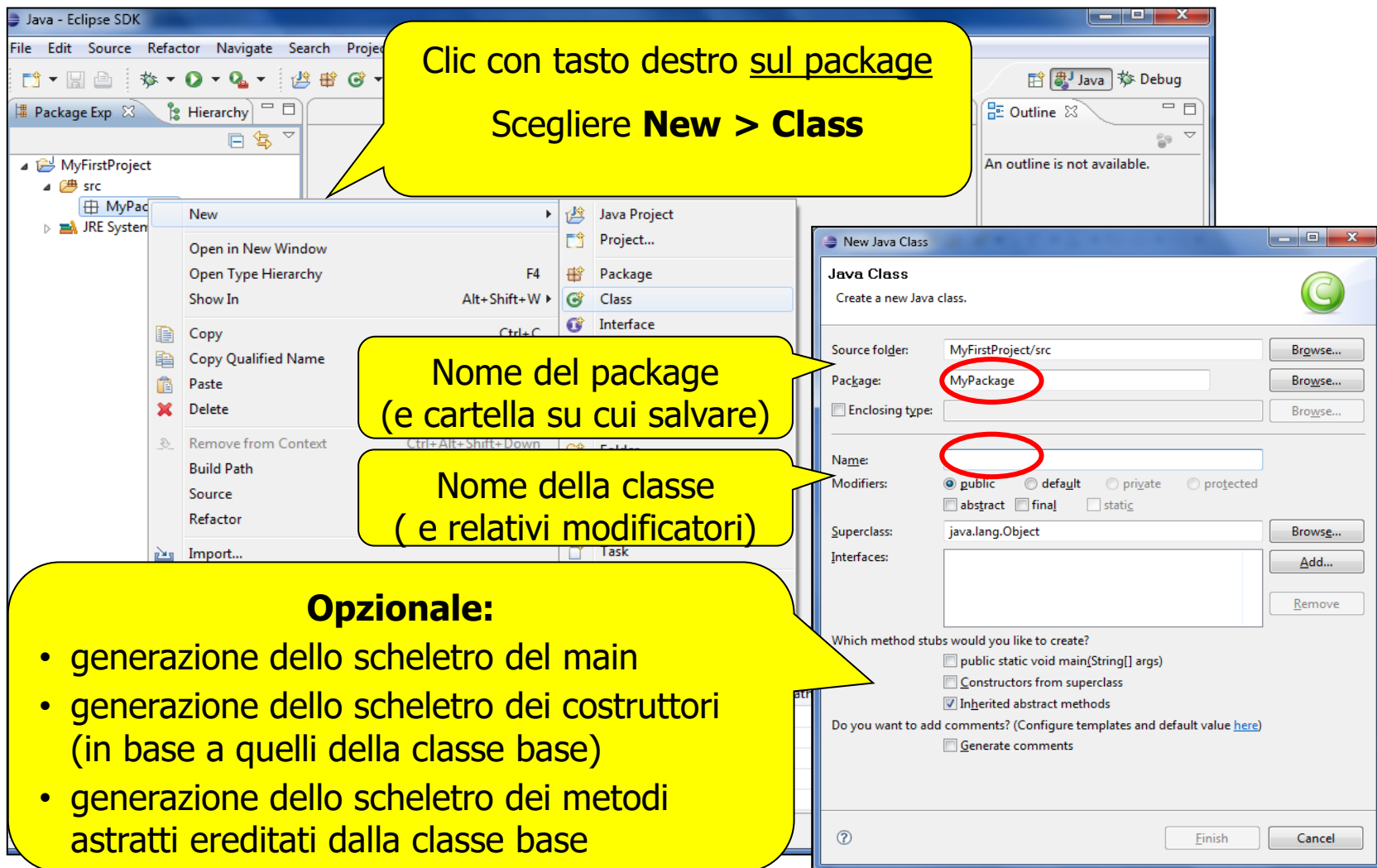
EclipseJava2018WS - MyFirstProject/src/MyMain.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help





# IL CASO CON PACKAGE (2)



Clic con tasto destro sul package  
Scegliere **New > Class**

Nome del package  
(e cartella su cui salvare)

Nome della classe  
(e relativi modificatori)

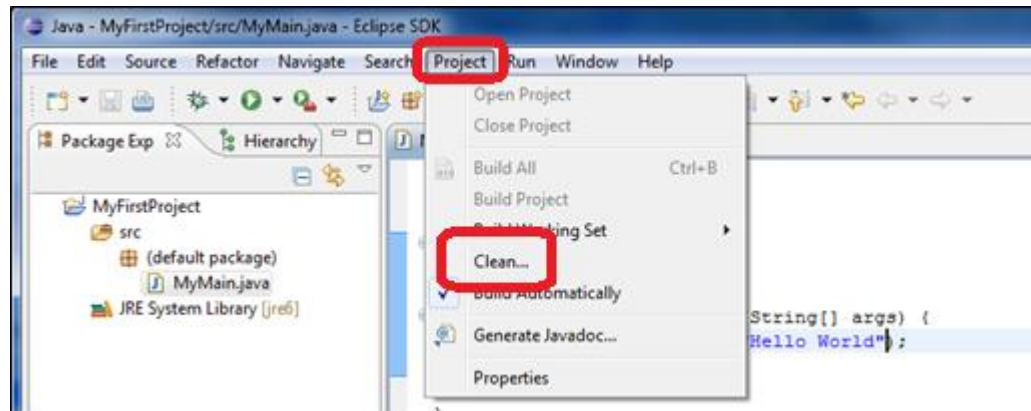
**Opzionale:**

- generazione dello scheletro del main
- generazione dello scheletro dei costruttori (in base a quelli della classe base)
- generazione dello scheletro dei metodi astratti ereditati dalla classe base

The screenshot shows the Eclipse IDE with the 'MyFirstProject' workspace. The 'src' package is selected, and the 'New' context menu is open. The 'Class' option is highlighted. The 'New Java Class' dialog is also shown, with the 'Package' field set to 'MyPackage' and the 'Name' field empty. The 'Modifiers' section shows 'public' selected. The 'Superclass' is set to 'java.lang.Object'. The 'Which method stubs would you like to create?' section has 'Inherited abstract methods' checked. The 'Do you want to add comments?' section has 'Generate comments' checked.

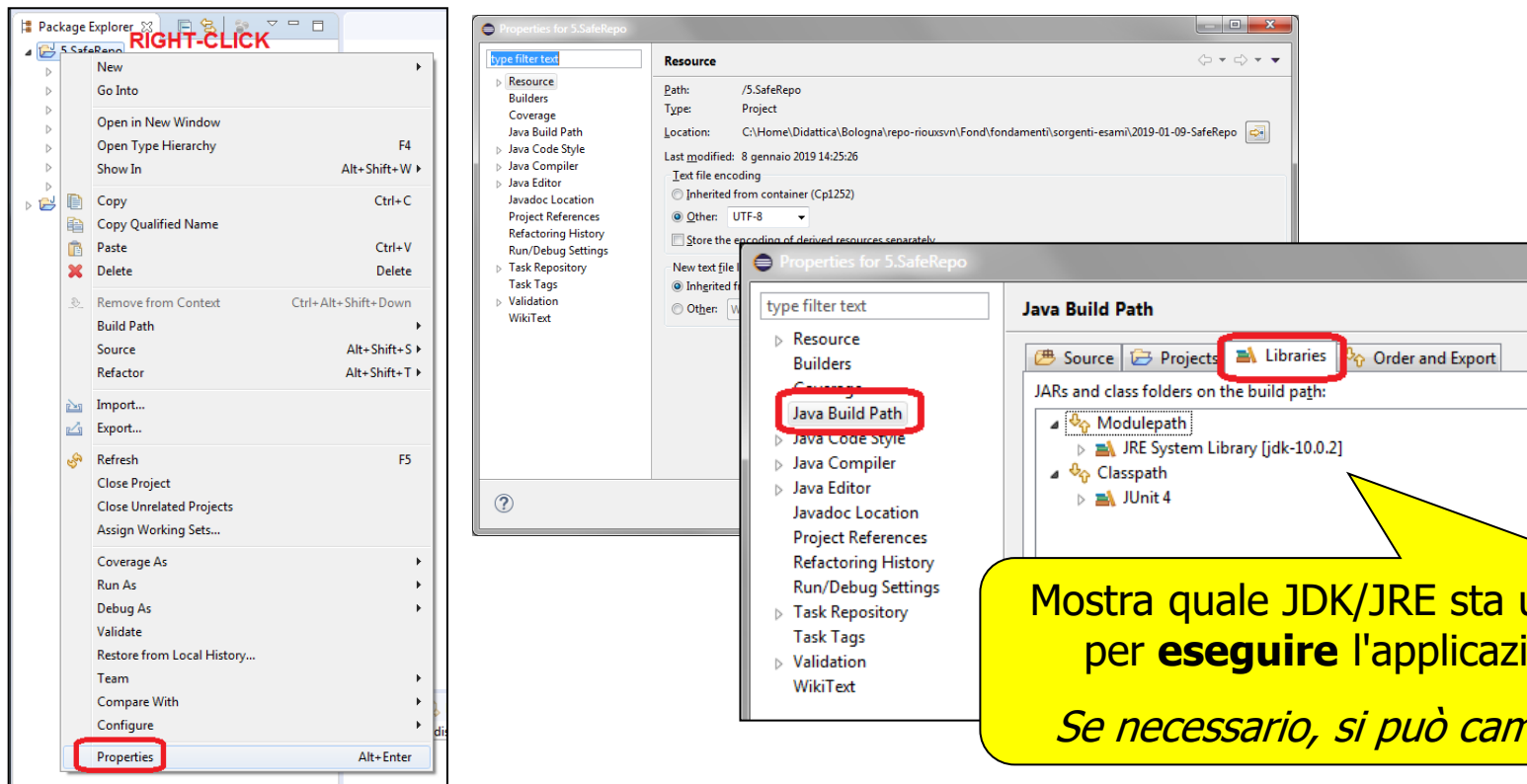
# COMPILAZIONE (1)

- Come già accennato, Eclipse effettua un *rebuild automatico* ad ogni salvataggio di un file sorgente
    - impostazione *Build Automatically* (attiva di default, ma disattivabile)
    - non è quindi necessario lanciare esplicitamente la compilazione
  - A volte, però, Eclipse non rileva modifiche "accavallatesi" in modo strano → restano errori incomprensibili
- SOLUZIONE:** il comando **Project > Clean**, che elimina tutti i file *.class* forzando una ricompilazione completa



# PROPRIETÀ DEL PROGETTO

- Talora può essere utile / necessario verificare le **proprietà del progetto**
  - clic con tasto destro sul nome del progetto > **Properties**
  - fra le tante che appaiono, interessa specialmente **Java Build Path**

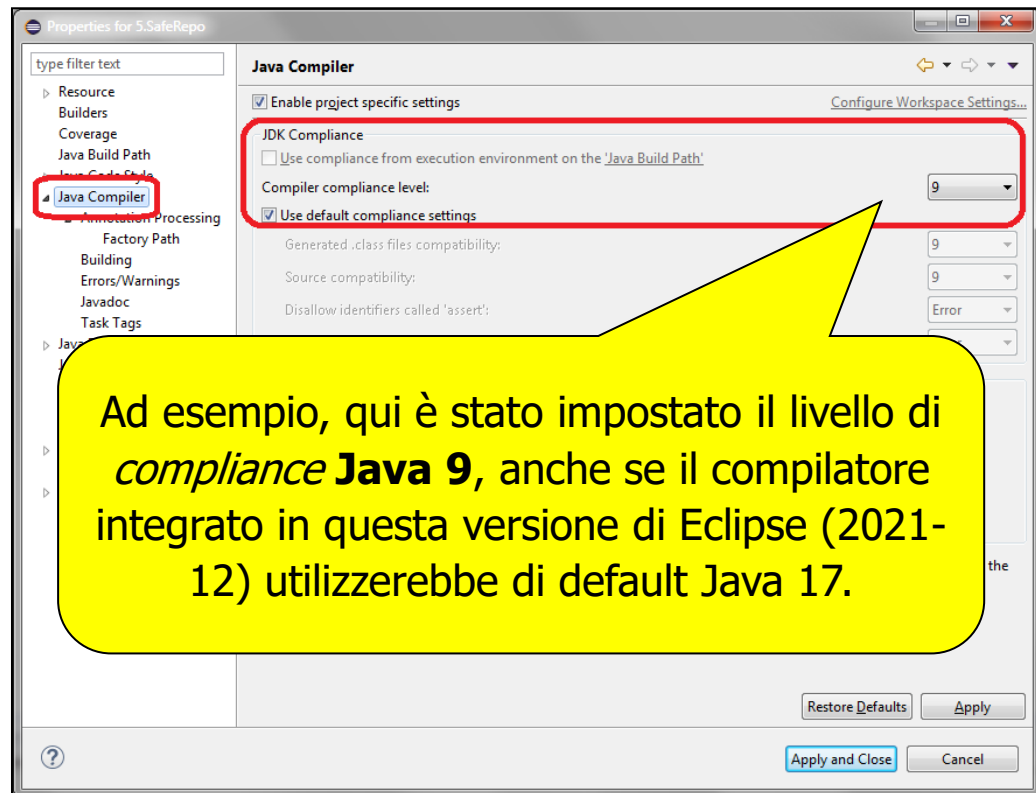
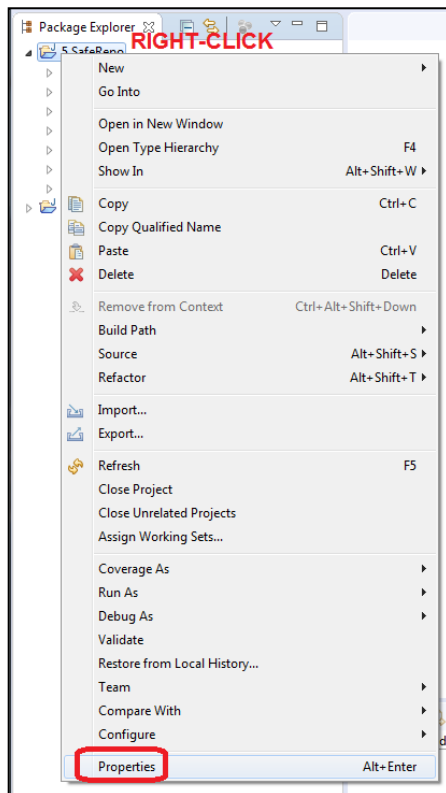


Mostra quale JDK/JRE sta usando  
per **eseguire** l'applicazione

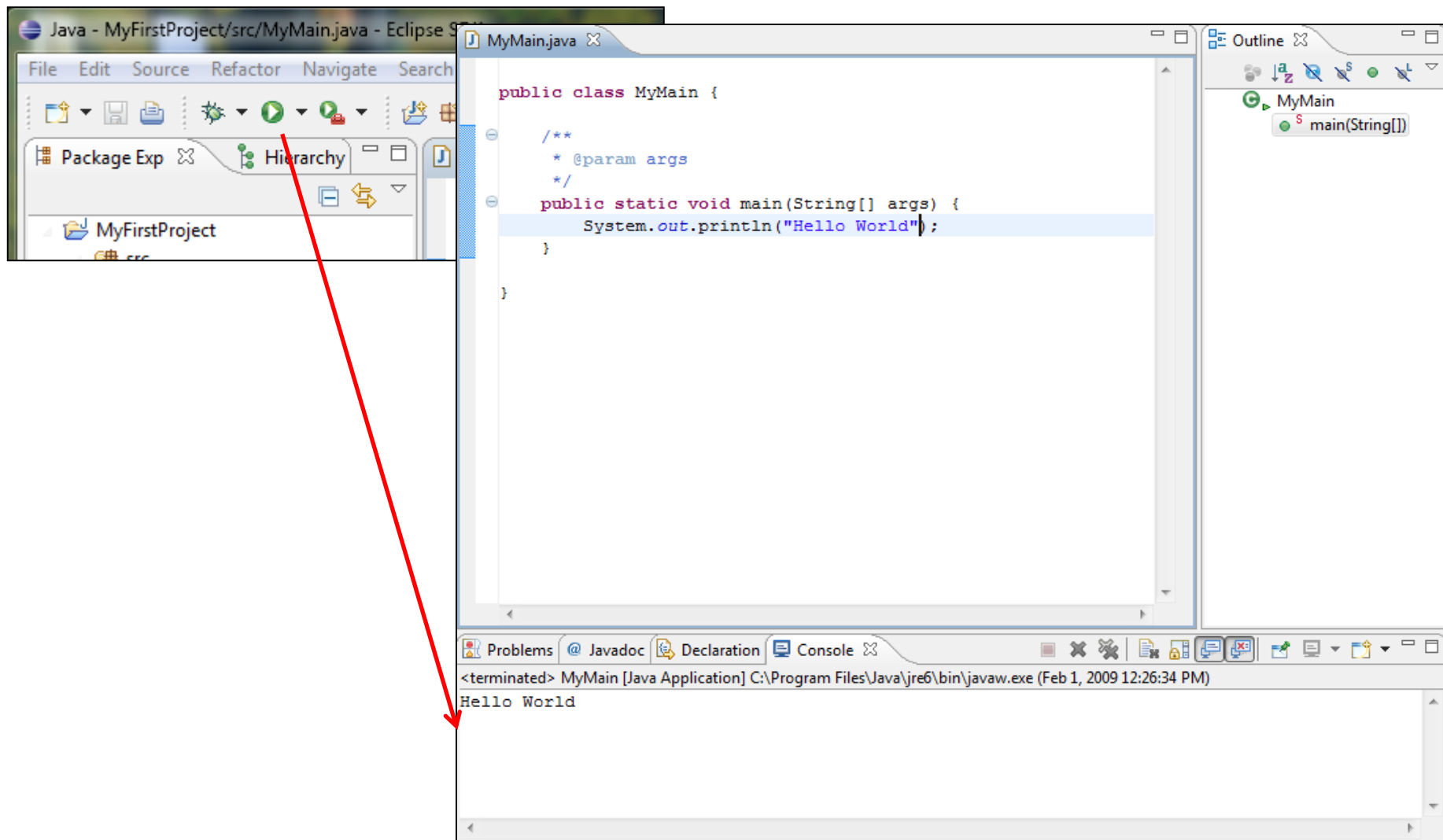
*Se necessario, si può cambiare*

# PROPRIETÀ DEL PROGETTO

- Se si desidera *compilare per versioni precedenti di Java*
  - ad esempio, compilare per Java 9 pur avendo Eclipse 2020-12, che è Java 15
  - occorre agire sulla proprietà *Java Compiler > JDK Compliance Level*

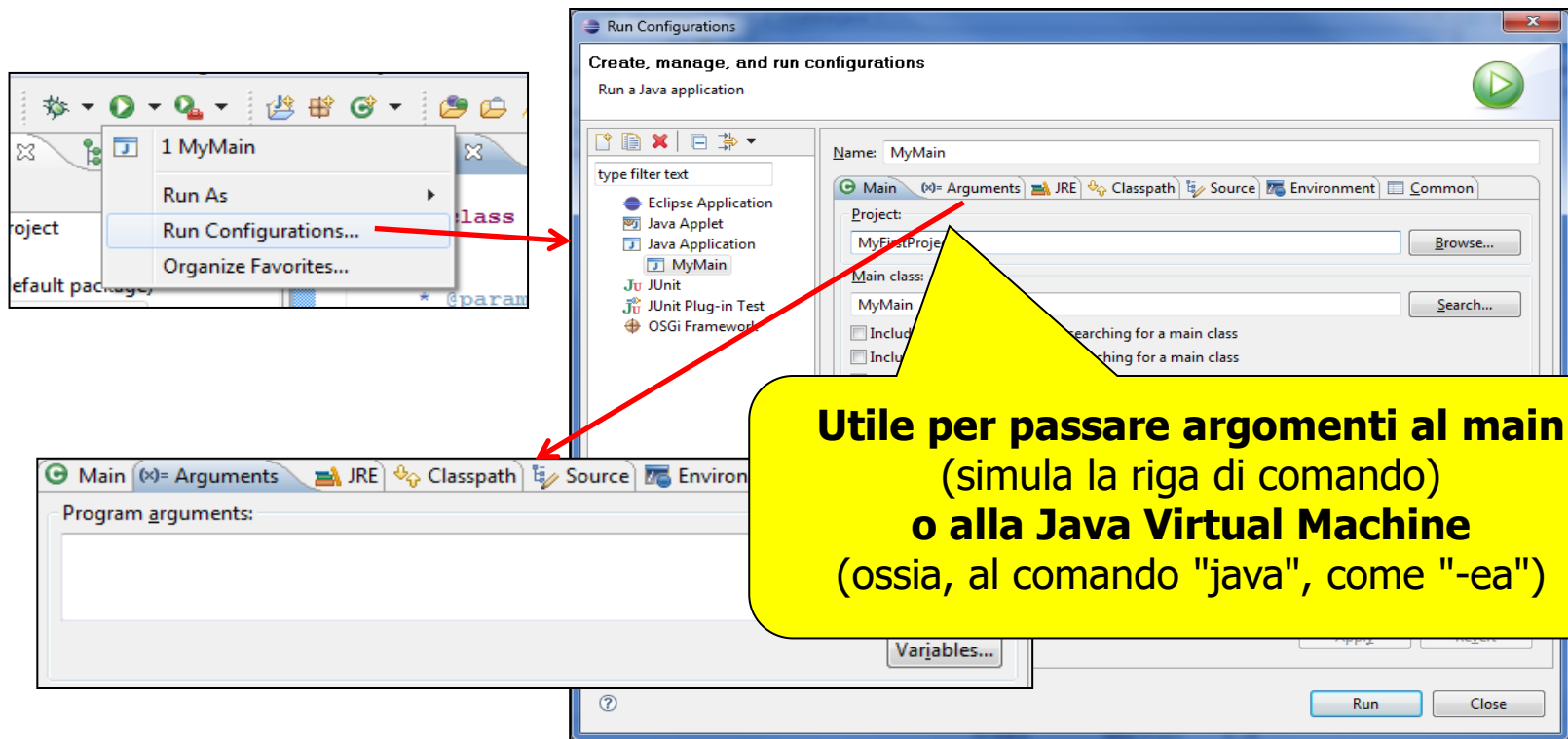


# ESECUZIONE (1)

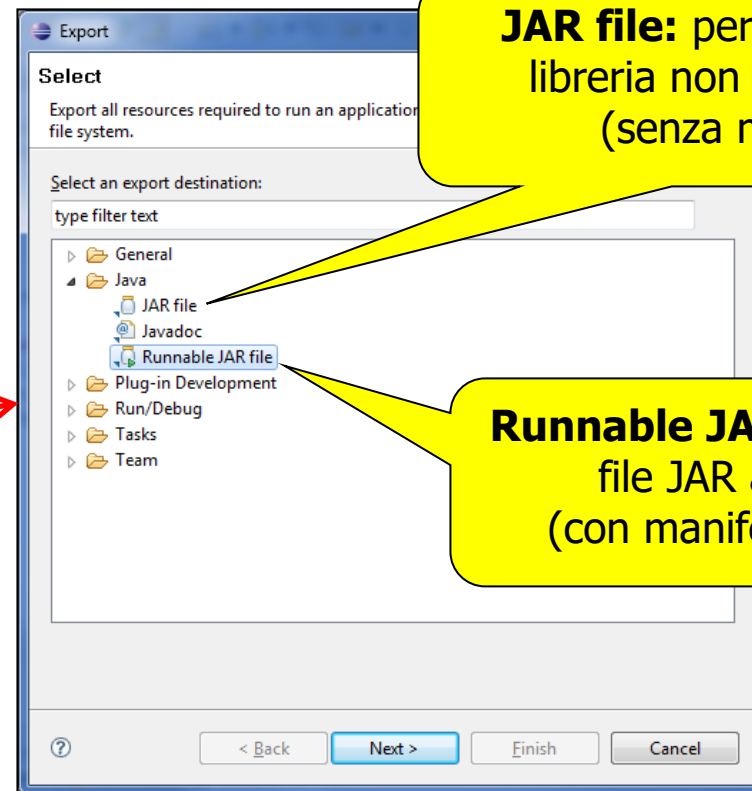
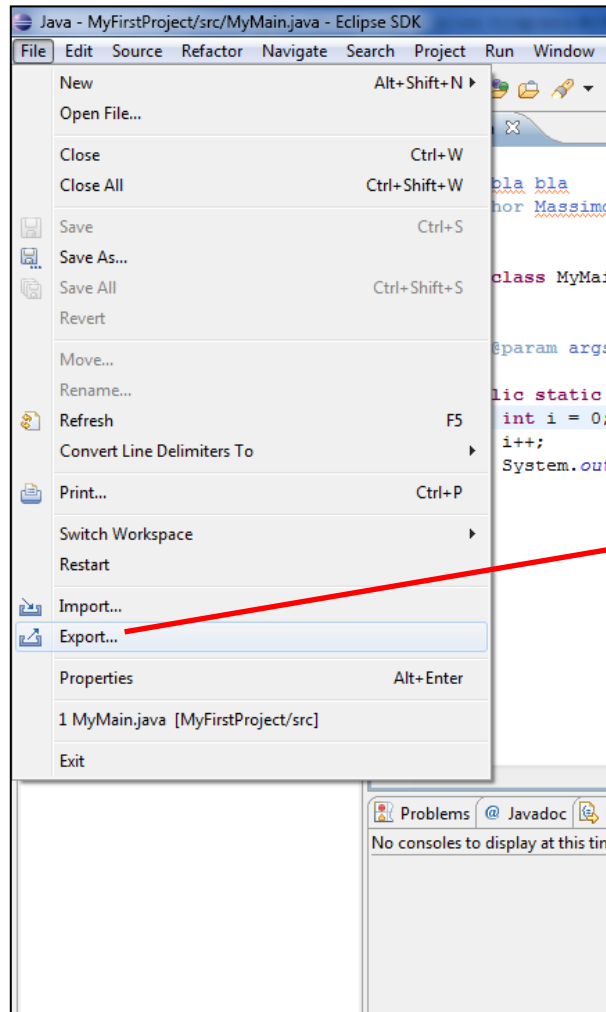


# ESECUZIONE (2)

- **Run Configurations** permette di impostare parametri di ingresso, runtime, classpath ecc.
- Si possono **definire più configurazioni** per uno stesso progetto



# CREAZIONE DI JAR (1)

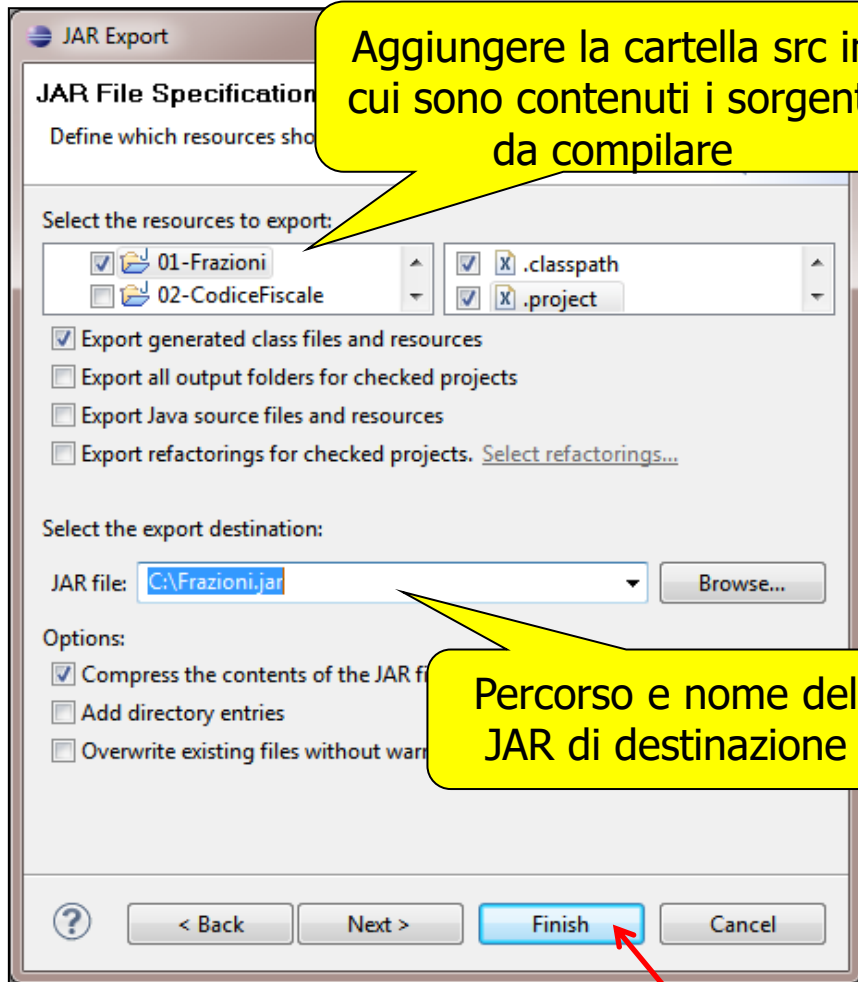


**JAR file:** per un JAR di libreria non avviabile (senza main)

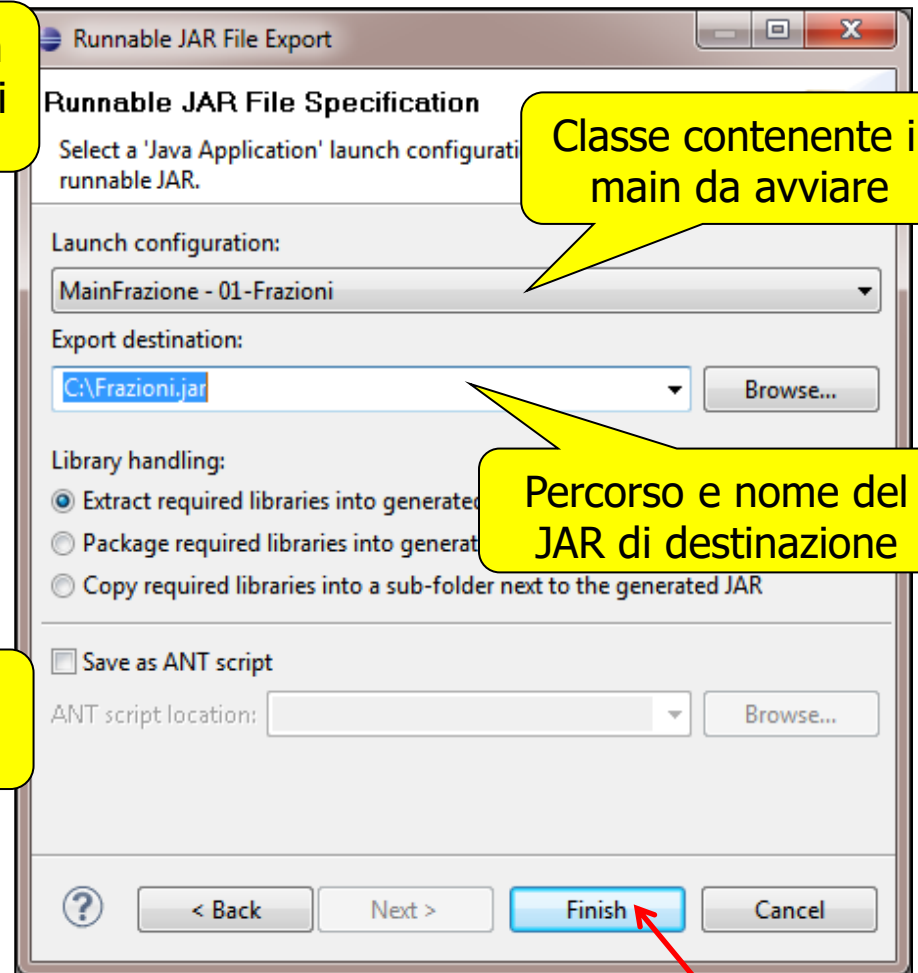
**Runnable JAR file:** per un file JAR avviabile (con manifest e main)

# CREAZIONE DI JAR (2)

## Libreria Jar



## Jar Eseguibile

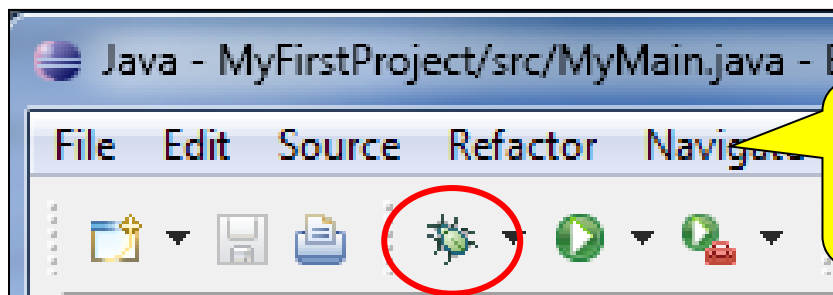




# DEBUG (1)

L'esecuzione dell'applicazione in *debug* permette di:

- eseguire **passo-a-passo**;
- controllare il valore delle **variabili** e vederle evolvere durante l'esecuzione;
- impostare **breakpoint**;
- visualizzare la catena delle chiamate a metodi.



NB: non si può debuggare se non c'è un *main* (o qualcosa del genere...)

Cliccando sull'icona "bug", scatta un **cambio di prospettiva**: dalla **"Java perspective"** alla **"Debug perspective"**.

# DEBUG (2)

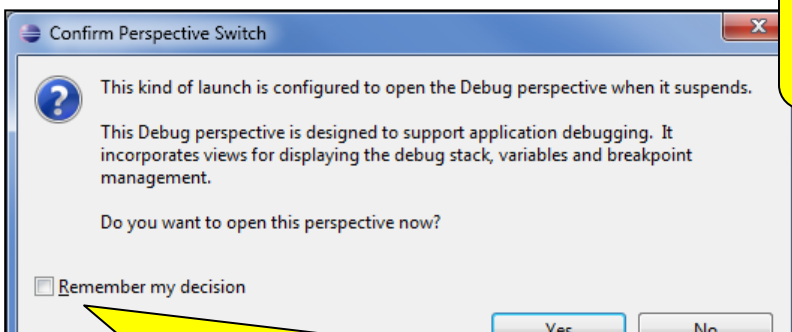
Nel codice sorgente è possibile impostare dei **breakpoint**.

Durante l'esecuzione il debug si fermerà in quel punto.

```
public class MyMain {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        int i = 0;  
        i++;  
        System.out.println(i);  
    }  
}
```

Impostazione/disattivazione del breakpoint tramite doppio click in questa zona

All'avvio del debug verrà richiesto se passare alla **perspective** di debug (...anche sì 😊)



Sì, ricordati. Quando «debuggo» mi serve la debug perspective.

**ATTENZIONE:** terminato il debug occorre **tornare manualmente** alla *Java perspective*, cliccando sull'apposito bottone.

# DEBUG (3): PERSPECTIVE

Menu di Esecuzione

The screenshot displays the Eclipse IDE in the Debug perspective. The top-left pane shows the 'Debug' console with the following content:

- MyMain [Java Application]
- MyMain at localhost:1323
- Thread [main] (Suspended (breakpoint at line 12 in MyMain))
- MyMain.main(String[]) line: 12
- C:\Program Files\Java\jre6\bin\javaw.exe (Feb 1, 2009 12:58:57 PM)

The top-right pane shows the 'Variables' and 'Breakpoints' tabs. The 'Variables' tab is active, displaying a table with the following data:

| Name | Value             |
|------|-------------------|
| args | String[0] (id=16) |

The bottom-left pane shows the 'MyMain.java' source code editor. The cursor is positioned at line 12, which is highlighted in blue. The code is as follows:

```
/**
 * Bla bla bla
 * @author Massimo
 */
public class MyMain {

    /**
     * @param args
     */
    public static void main(String[] args) {
        int i = 0;
    }
}
```

The bottom-right pane shows the 'Outline' view, displaying the class hierarchy:

- MyMain
- main(String[])






The bottom pane shows the 'Console' view, which is currently empty.

Red annotations are present on the image:

- A red arrow points to the 'Debug' menu icon in the top-left pane.
- The text 'Stack dei record di attivazione' is written in red over the 'Debug' console.
- The text 'Stato delle Variabili e dei Breakpoints' is written in red over the 'Variables' and 'Breakpoints' tabs.
- The text 'Posizione durante l'esecuzione sul codice sorgente' is written in red over the source code editor.
- The text 'Console' is written in red over the console view.

# DEBUG (4)

## Menu Di Esecuzione

-  → Continua l'esecuzione fino al breakpoint successivo
-  → Mette in pausa l'esecuzione
-  → Ferma l'esecuzione
-  → Entra all'interno del metodo (shortcut F5)
-  → Prosegue senza entrare nel metodo (shortcut F6)

# GENERAZIONE DELLA DOCUMENTAZIONE

Java - MyFirstProject/src/MyMain.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Open Project  
Close Project  
Build All Ctrl+B  
Build Project  
Build Working Set  
Clean...  
Build Automatically  
Generate Javadoc...

MyFirstProject  
src  
(default package)  
MyMain.java  
JRE System Library [jre6]

String[  
Hello W

Generate Javadoc

Javadoc Generation

Select types for Javadoc generation.

Javadoc command:  
C:\Program Files\Java\jdk1.6.0\_11\bin\javadoc.exe

Select types for which Javadoc will be generated:  
MyFirstProject

Public: Generate Javadoc for public classes and members.  
Use standard doclet  
Destination: C:\Users\Massimo\Desktop\Java\MyFirstProject\doc

Use custom doclet  
Doclet name:  
Doclet class path:

Generate Javadoc

Javadoc Generation

Configure Javadoc arguments for standard doclet.

Document title:

Basic Options  
Generate use page  
Generate hierarchy tree  
Generate navigator bar  
Generate index  
Separate index per letter

Document these tags  
@author  
@version  
@deprecated  
deprecated list

Select referenced archives and projects to which links should be generated:

Charsets.jar - http://java.sun.com/javase/6/docs/api/  
DNS.jar - http://java.sun.com/javase/6/docs/api/  
JCE.jar - http://java.sun.com/javase/6/docs/api/  
JSSE.jar - http://java.sun.com/javase/6/docs/api/

MyFirstProject  
src  
(default package)  
MyMain.java  
JRE System Library [jre6]  
doc  
class-use  
index-files  
resources  
allclasses-frame.html  
allclasses-noframe.html  
constant-values.html  
deprecated-list.html  
help-doc.html  
index.html  
MyMain.html  
overview-tree.html  
package-frame.html  
package-list  
package-summary.html  
package-tree.html  
package-use.html  
stylesheet.css

Impostare il percorso in cui trovare javadoc.exe [jdkDirectory]/bin

Directory in cui generare la documentazione. Default [projectDirectory]/doc

Finish