

UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica

Laboratorio di Programmazione ad Oggetti

(Object Oriented Programming)

Dott. Ric. Juri di Rocco
juri.dirocco@univaq.it

**Dott. Ric. Riccardo
Rubei**
riccardo.rubei@univaq.it

Cos'è Eclipse

- › Eclipse è una community Open Source
- › Più di 300 progetti dedicati a vari aspetti dello sviluppo software
- › I progetti Eclipse supportano diverse aree:
 - Un ambiente di sviluppo per applicazioni Java e Android
 - Un ambiente di sviluppo per tecniche di modeling
 - Un ambiente di sviluppo J2EE
 -

Cos'è Eclipse

› Eclipse è una community Open Source

› Pi

› Ip

PROJECT NAME	WEBSITE	CODE	DOWNLOAD
> 4DIAC - Framework for Distributed Industrial Automation and Control			
> Acceleo			
> Accessibility Tools Framework			
> Aether			
> Agent Modeling Platform			
> Ajax Tools Framework (ATF)			
> AJDT - AspectJ Development Tools Project			
> Amalgamation			
> AMW - Atlas Model Weaver			
> Andmore - Eclipse Android Tooling			
> APP4MC			

ware

Eclipse

- › Il progetto risale al 2001 quando il primo codice venne sviluppato da IBM
- › Nel 2004 è divenuto Eclipse Foundation
- › Ma perché il nome Eclipse?

Eclipse

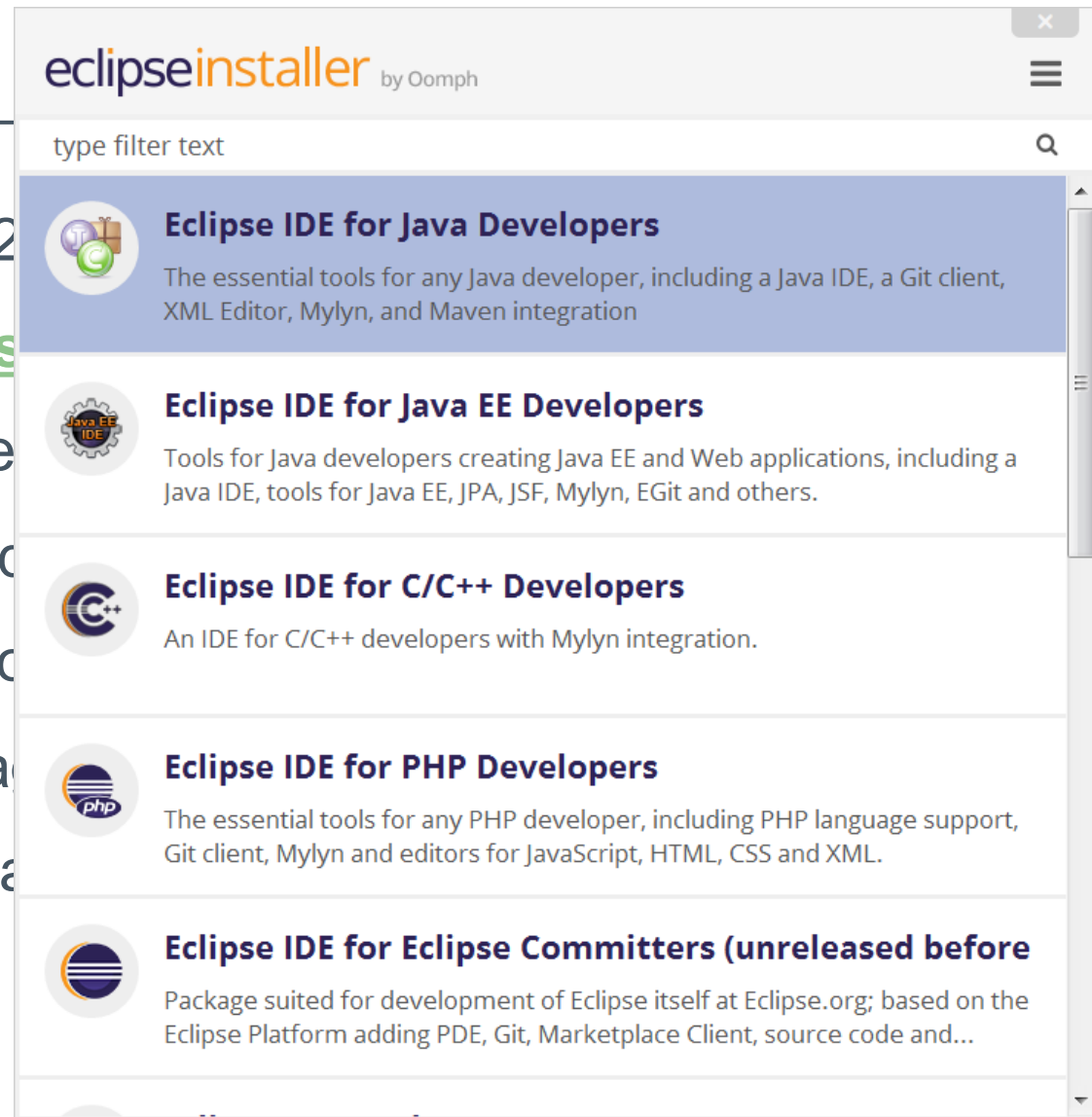
- › Il progetto risale al 2001 quando il primo codice venne sviluppato da IBM
- › Nel 2004 è divenuto Eclipse Foundation
- › Ma perché il nome Eclipse?
- › L'idea di IBM era quella di eclissare Microsoft
- › Con l'acquisizione di Sun Microsystems da parte di Oracle il conflitto è finalmente cessato
- › Oracle è attualmente uno delle 5 più grandi compagnie che contribuiscono allo sviluppo del progetto Eclipse

Eclipse

- › Il 23 novembre 2022 è stata rilasciata la versione **4.26**
- › In questo link <https://download.eclipse.org/eclipse/downloads/> si può trovare la versione più recente di Eclipse SDK
- › L'IDE è composta da diversi componenti
- › Il sito Eclipse.org fornisce un installer che permette di installare versioni già "packaged" (ad esempio Eclipse Modeling Tool, Eclipse IDE for java developers, Eclipse for Enterprise Java Developers ecc.)

Eclipse

- › Il 23 novembre 2022
- › In questo link <https://www.eclipse.org/ideads/> si può trovare la versione più recente dell'Eclipse IDE.
- › L'IDE è composta da diverse versioni già "packaged" (Eclipse IDE for Java Developers ecc.)



Getting started

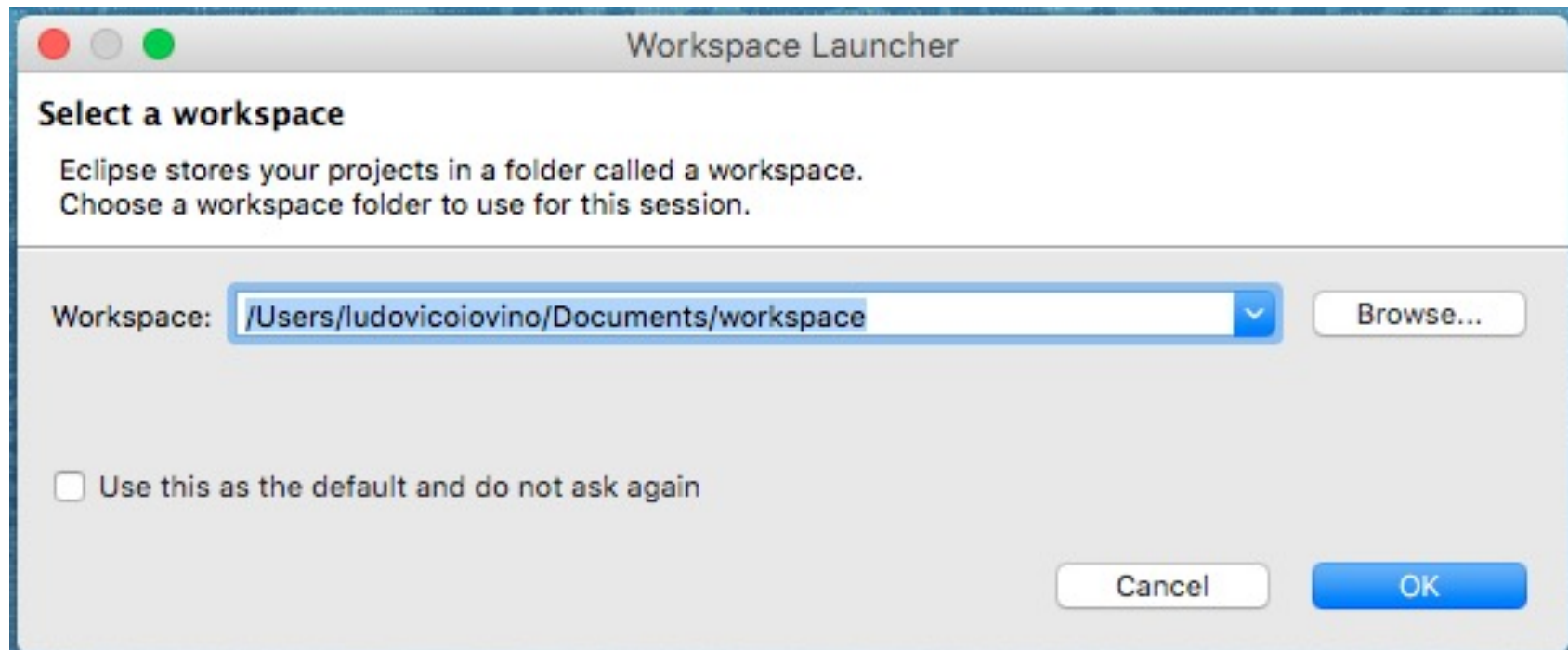
› In base al sistema operativo:

- Per avviare Eclipse, doppio click su eclipse.exe (Microsoft Windows)
- Oppure file eclipse nella cartella dove è stato spaccettato (Linux/Mac)



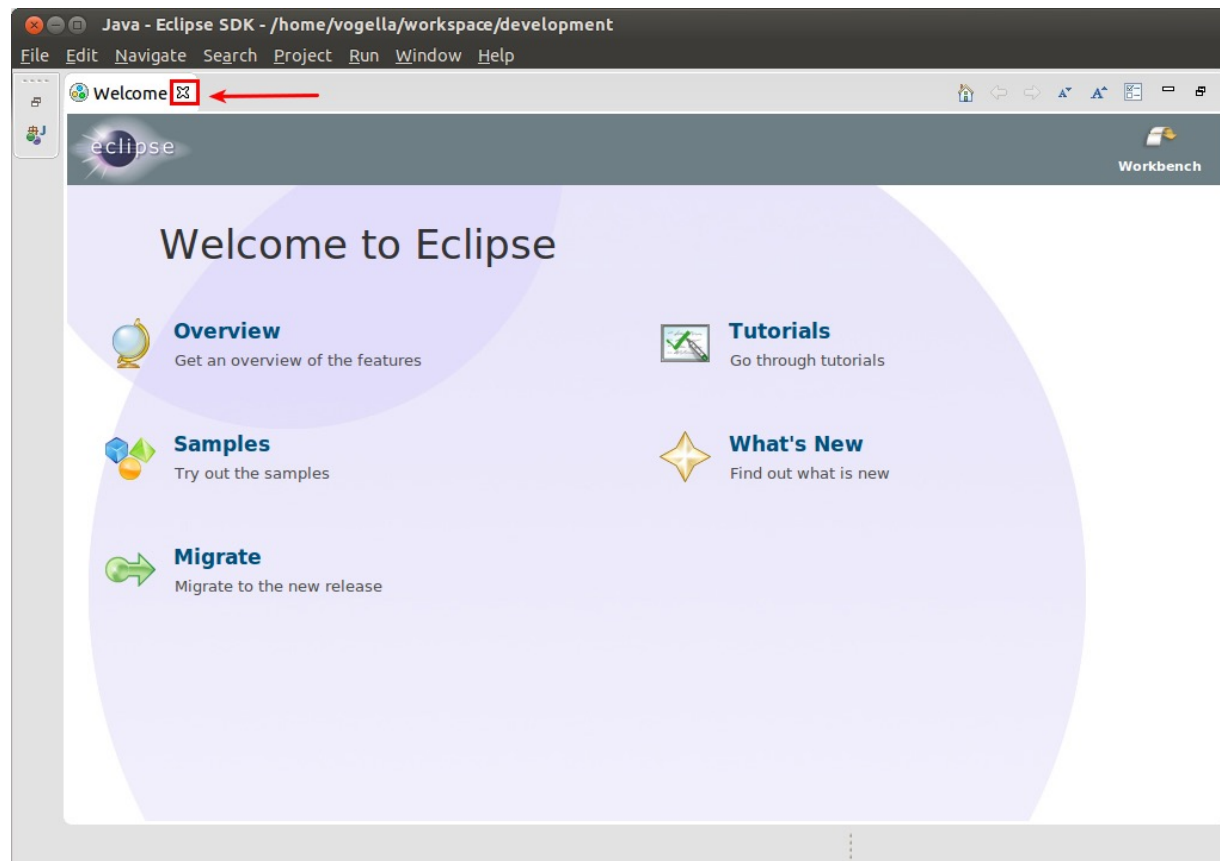
Eclipse Workspace

- › Eclipse vi chiederà di creare un workspace
- › Potete selezionare una cartella anche fuori da quella di default dove Eclipse è installato



Eclipse Workspace

- › Una volta scelto il workspace Eclipse si avvierà e vi mostrerà questa schermata di benvenuto

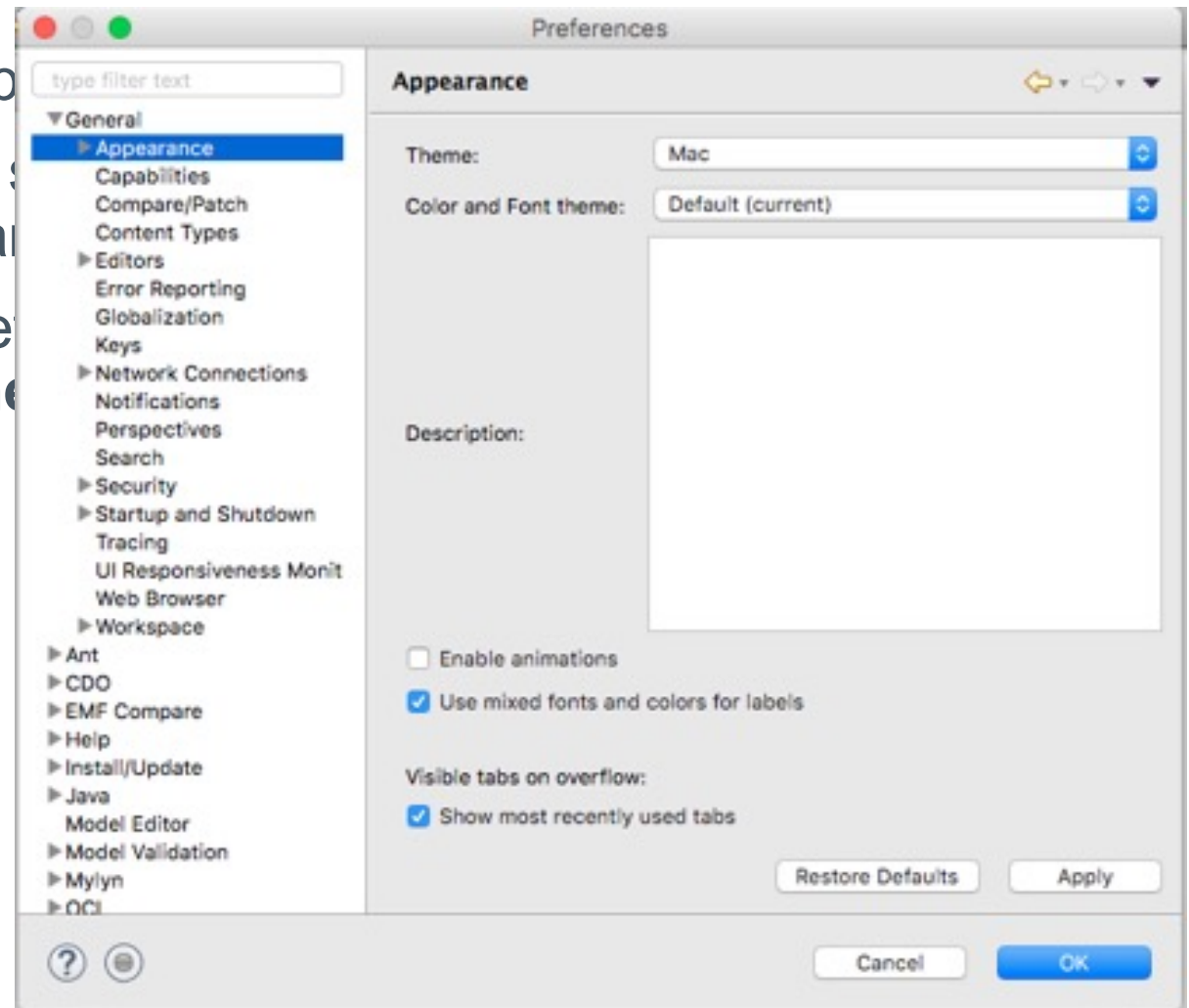


Aspetto

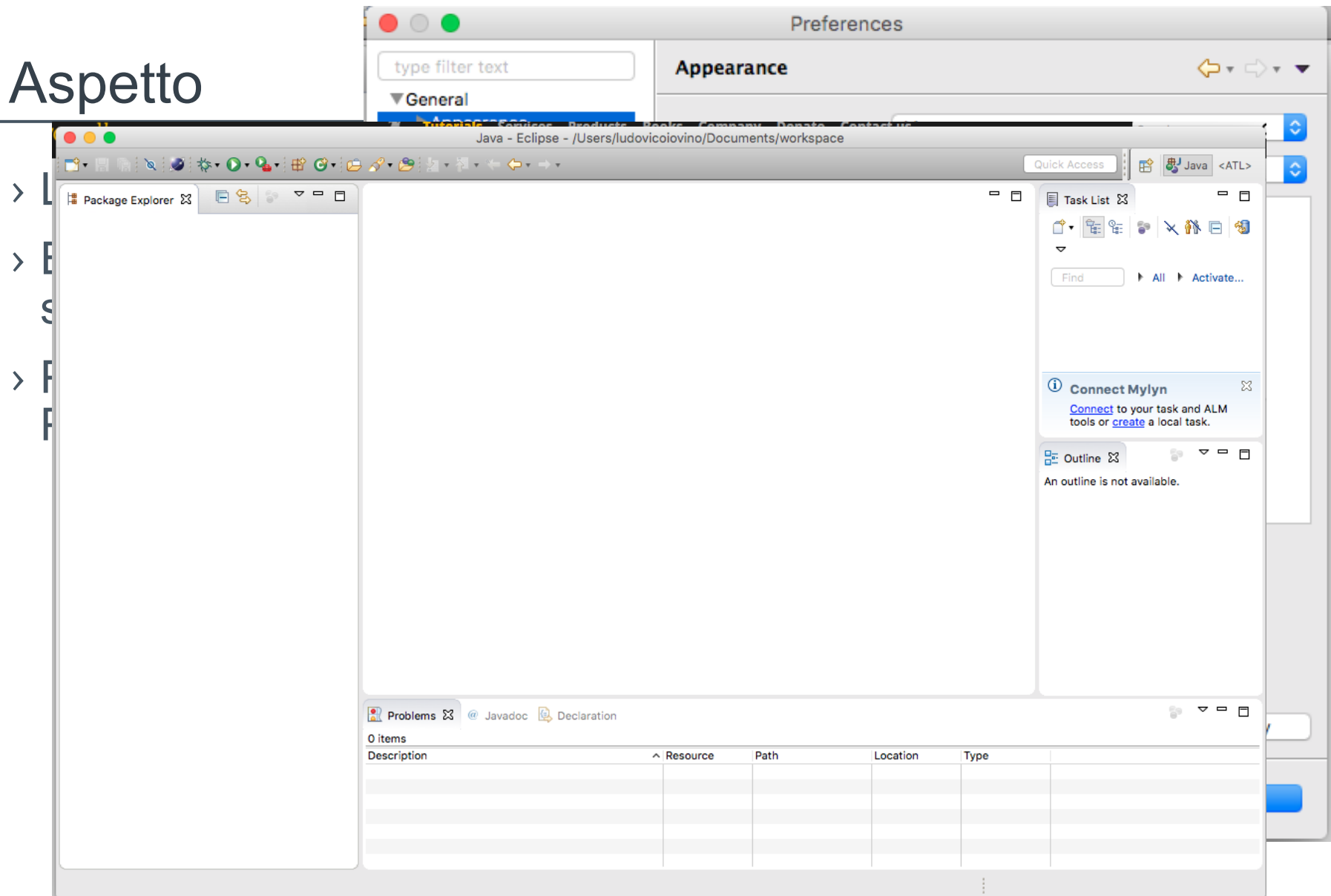
- › L'aspetto di Eclipse può essere configurato
- › Eclipse fornisce una serie di temi predefiniti, ma se ne possono scaricare altri dal marketplace
- › Per modificare l'aspetto, selezionare dal menu **Window-> Preferences -> General -> Appearance**

Aspetto

- › L'aspetto di Eclipse p
- › Eclipse fornisce una s
scaricare altri dal ma
- › Per modificare l'aspe
Preferences -> Gene

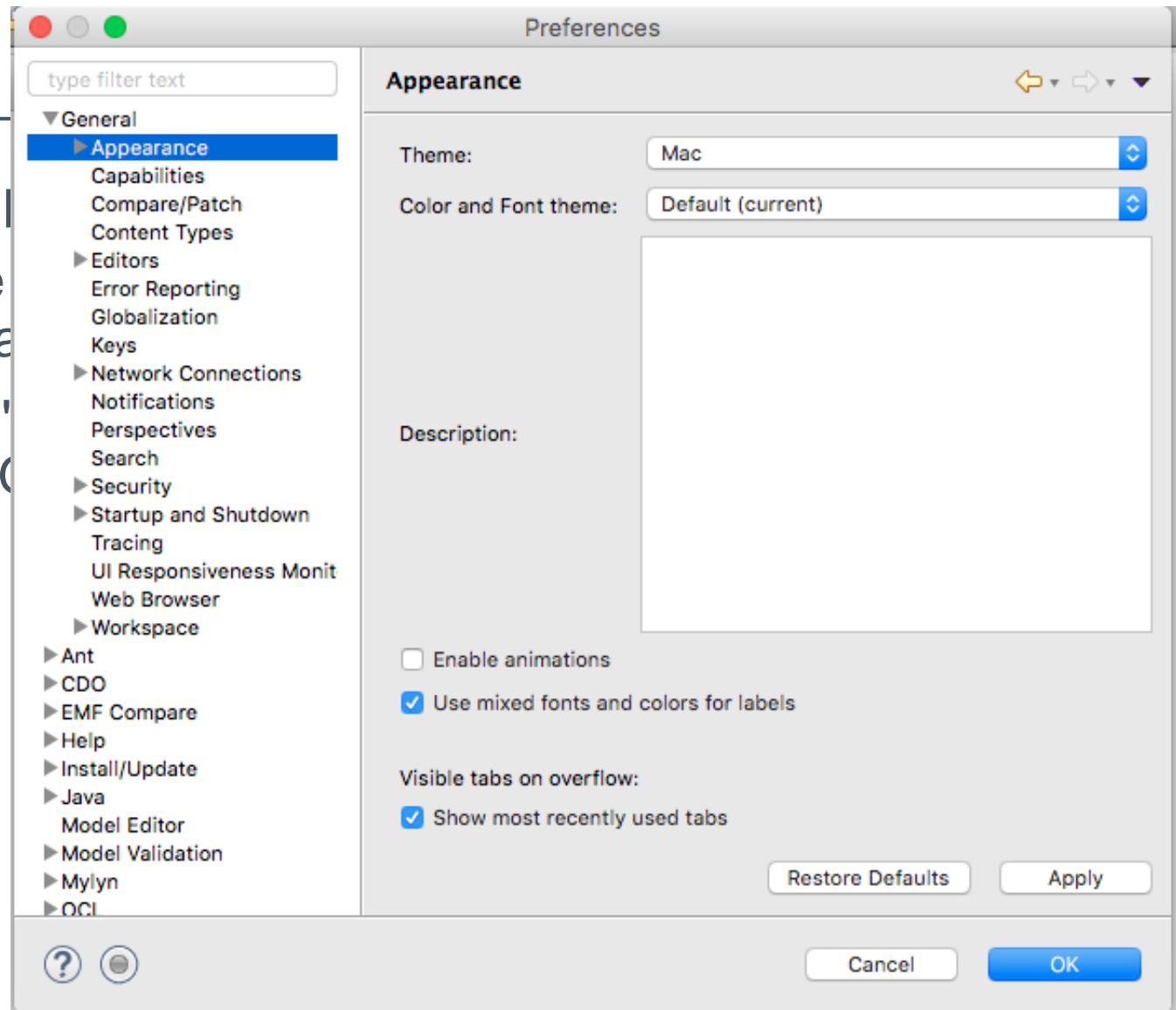


Aspetto

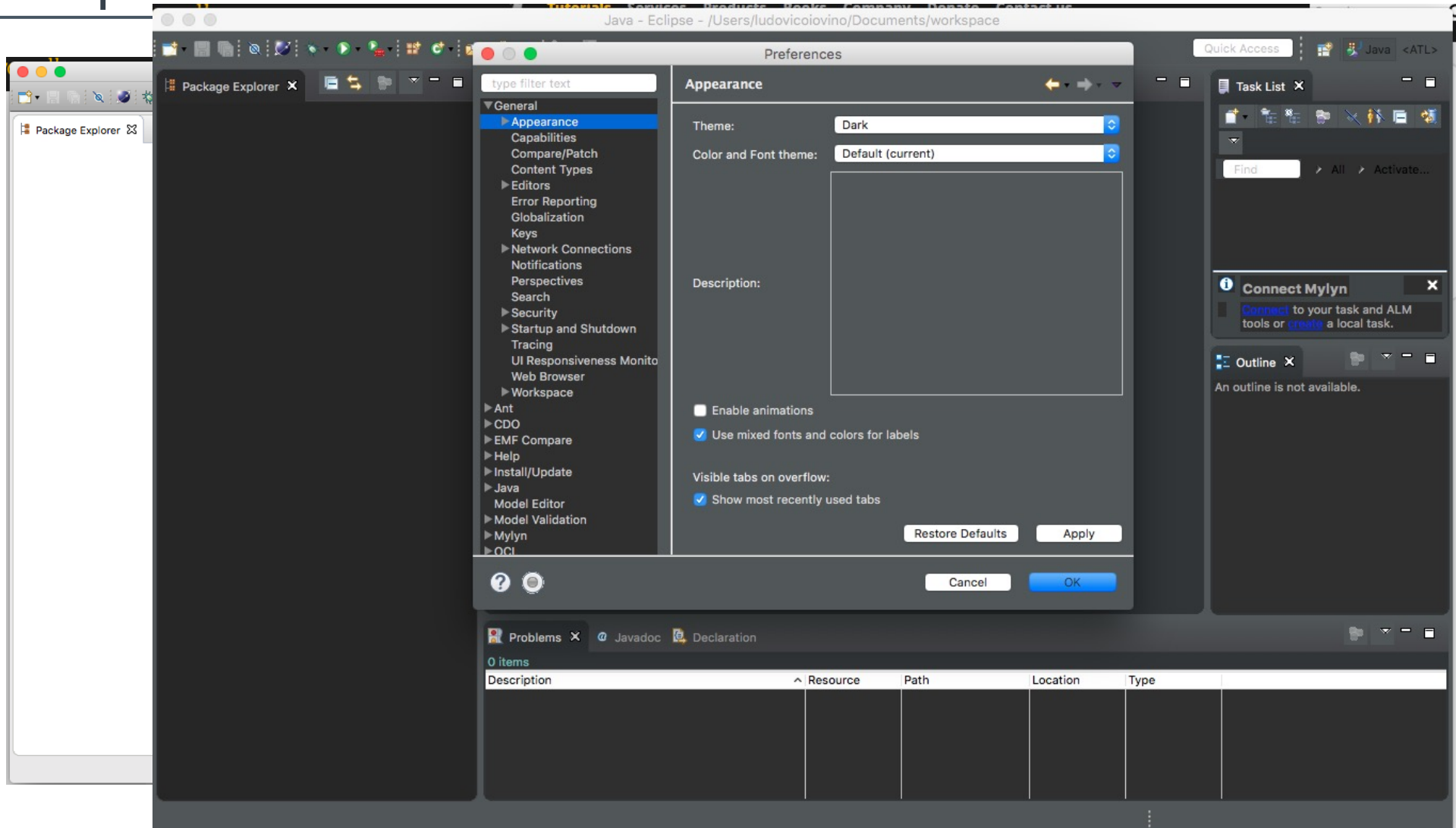


Aspetto

- › L'aspetto di Eclipse
- › Eclipse fornisce il modo per scaricare altri dati
- › Per modificare l'aspetto di Eclipse, andare in Preferences -> Appearance



Aspetto



Eclipse Workspace

- › L'Eclipse workspace è il percorso e location sul file system dove Eclipse salva le sue configurazioni, preferenze e altre risorse
- › Lo scopo di un workspace è di raggruppare un insieme di progetti simili con lo scopo di creare un'applicazione
- › In genere si creano diversi workspaces se si ha la necessità di differenziare i settings dei vari progetti, o dividerli fisicamente in cartelle diverse
- › Si può passare da un workspace all'altro andando su **File->Switch Workspace -> Others**

Eclipse Workspace

- › Se vi trovate in un workspace A e decidete di passare su un workspace B
 - Eclipse si chiuderà e si riaprirà
 - Qualunque progetto associato al workspace A (quelli visibili nel Project Explorer) non saranno più visibili
 - I progetti associati al workspace B saranno visibili
- › Quindi, un progetto per poter essere "aperto" all'interno di Eclipse **deve** essere associate ad un workspace

Eclipse Workspace

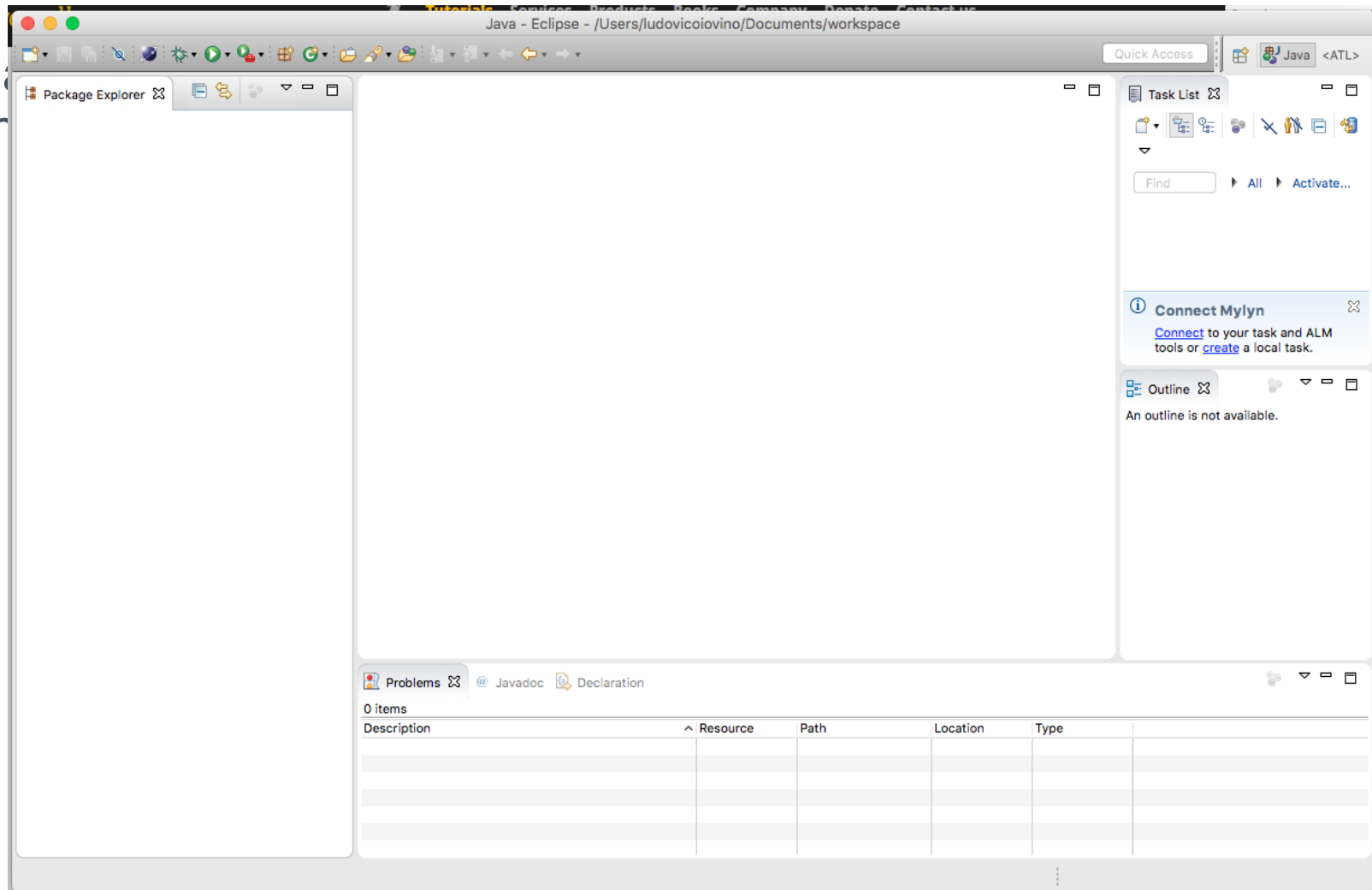
- › Il workspace contiene diverse risorse ad esempio:
 - Progetti
 - Files
 - Cartelle
- › Il workspace possiede una struttura gerarchica
- › I progetti sono al livello più alto della gerarchia al cui interno possiamo trovare file e cartelle
- › Gli utenti usano le funzionalità fornite dalle views, editor e wizard per creare e modificare le risorse nel workspace

Eclipse Workspace

- › Una volta chiusa la schermata di benvenuto, dovrete vedere una schermata simile a questa

Eclipse Workspace

> Un
sch



Eclipse Projects

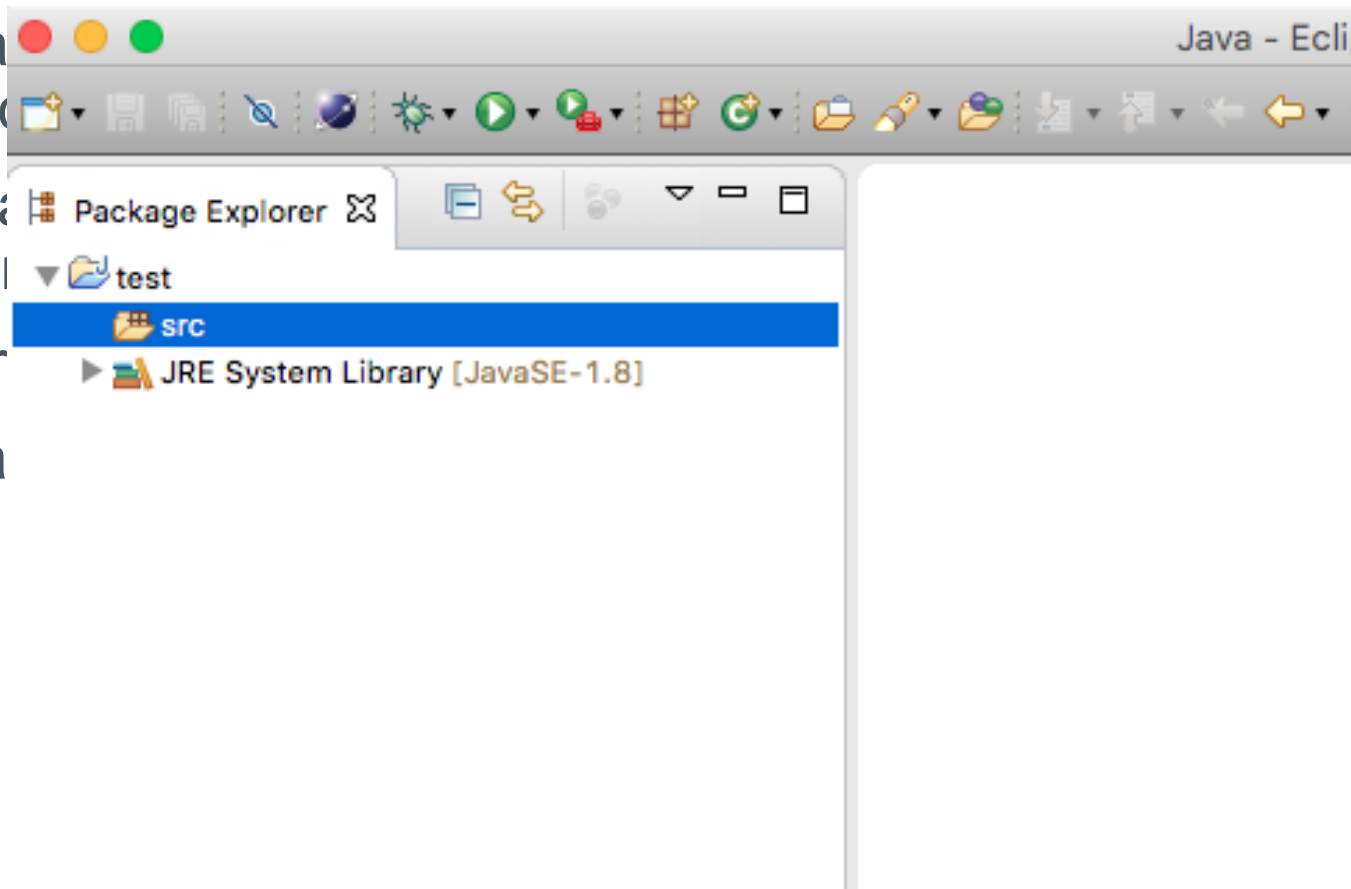
- › Un progetto Eclipse contiene codice sorgente, file di configurazione e binari relativi ad un certo task e li raggruppa in unità buildabili e riutilizzabili
- › Un progetto Eclipse può avere una **natura** assegnata che serve a descriverlo
 - Per esempio Java **nature** definisce un progetto Java
 - I progetti posso avere diverse nature per poter esprimere diversi aspetti
- › Le natures di un progetto sono definite nel file **.project** contenuto nella cartella del progetto
- › Progetti Eclipse non possono contenere altri progetti al loro interno

Eclipse Parts

- › Le "parts" sono interfacce utente che permettono di navigare e modificare le informazioni
- › Una part può avere un menù a tendina, un menu contestuale e una toolbar
- › Le parts sono generalmente divise in **viste** e **editors**
- › Un classico esempio di vista in Eclipse è il Package Explorer

Eclipse Parts

- › Le "pa
modific
 - › Una pa
toolba
 - › Le par
 - › Un cla
- are e
iale e una
rer

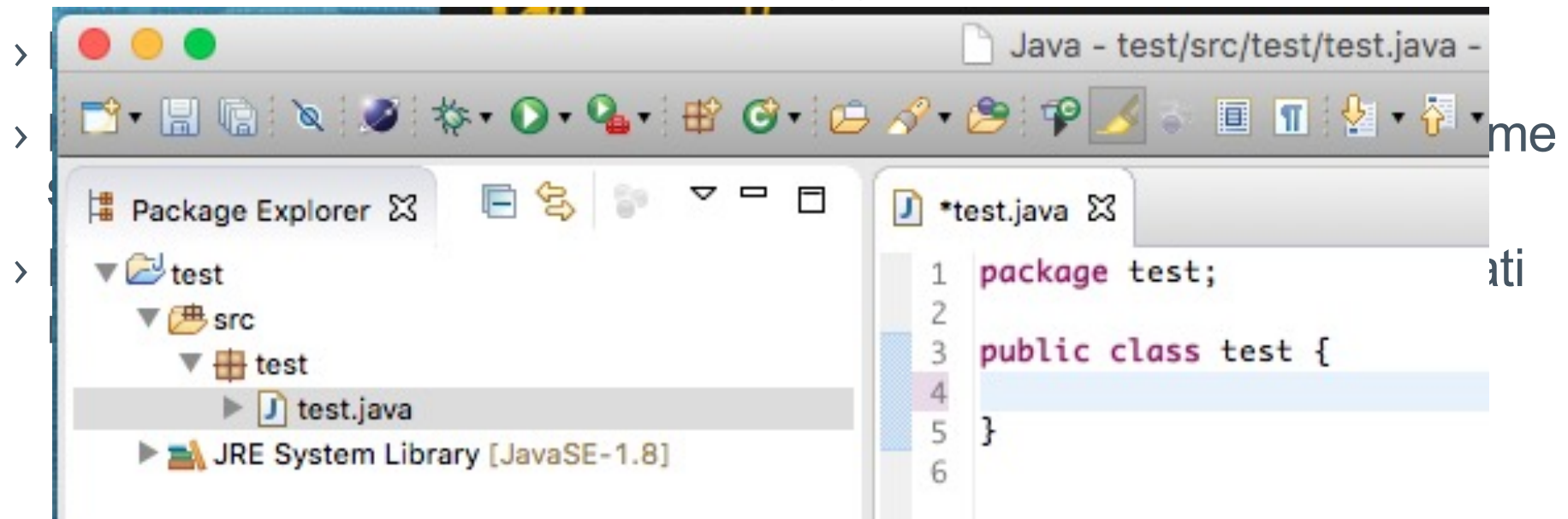


Eclipse Parts

- › Gli editor sono generalmente usati per modificare un singolo elemento, per esempio un file o un oggetto
- › Per rendere effettivi i cambiamenti fatti nell'editor, l'utente deve esplicitamente effettuare il salvataggio
- › L'editor Java si utilizza per modificare il codice di file java
- › Le modifiche al codice sono applicate una volta che l'utente preme sul pulsante salva
- › Nell'editor apparirà un asterisco accanto a quei file che sono stati modificati ma non ancora salvati

Eclipse Parts

- › Gli editor sono generalmente usati per modificare un singolo elemento, per esempio un file o un oggetto
- › Per rendere effettivi i cambiamenti fatti nell'editor, l'utente deve esplicitamente effettuare il salvataggio

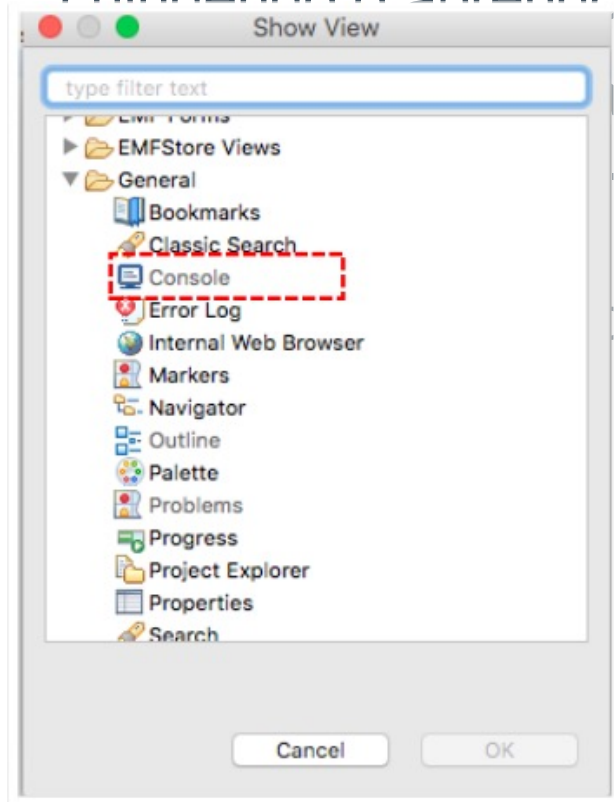


Eclipse viste

- › Si può modificare il layout e il contenuto di una perspective chiudendo o aprendo parts o spostandoli direttamente
- › Per aprire una nuova part in una perspective, andare su **Window - > Show View -> Other...**
- › Nella prossima slide possiamo vedere come la finestra di dialogo ci permetta di cercare parts

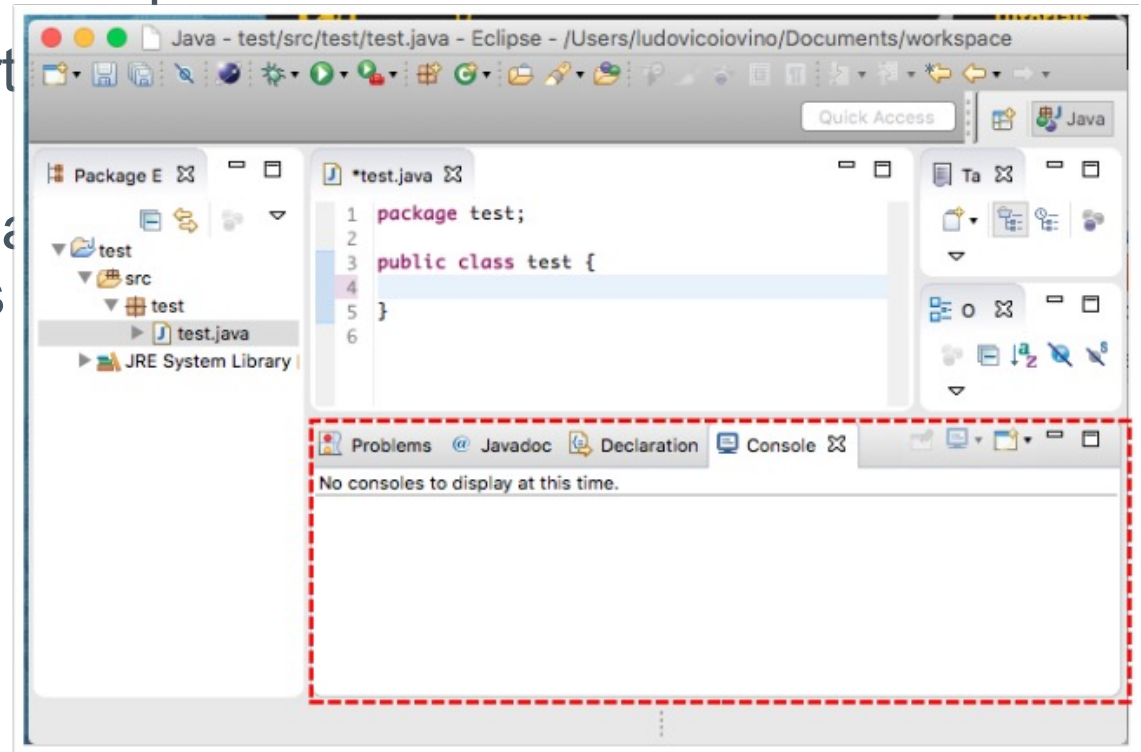
Eclipse viste

- › Si può modificare il layout e il contenuto di una perspective chiudendo o aprendo parts o spostandoli direttamente



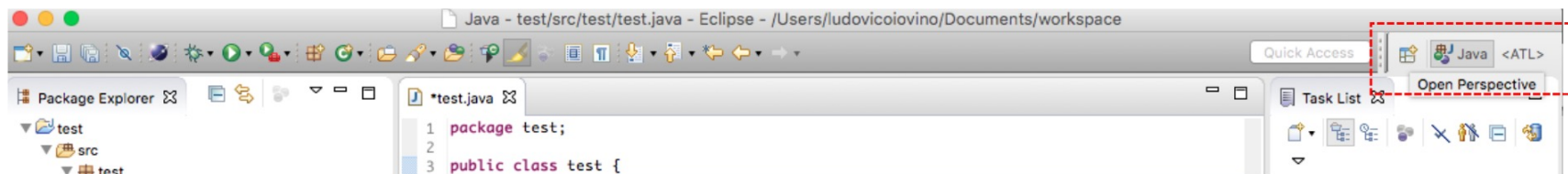
part

posia
parts



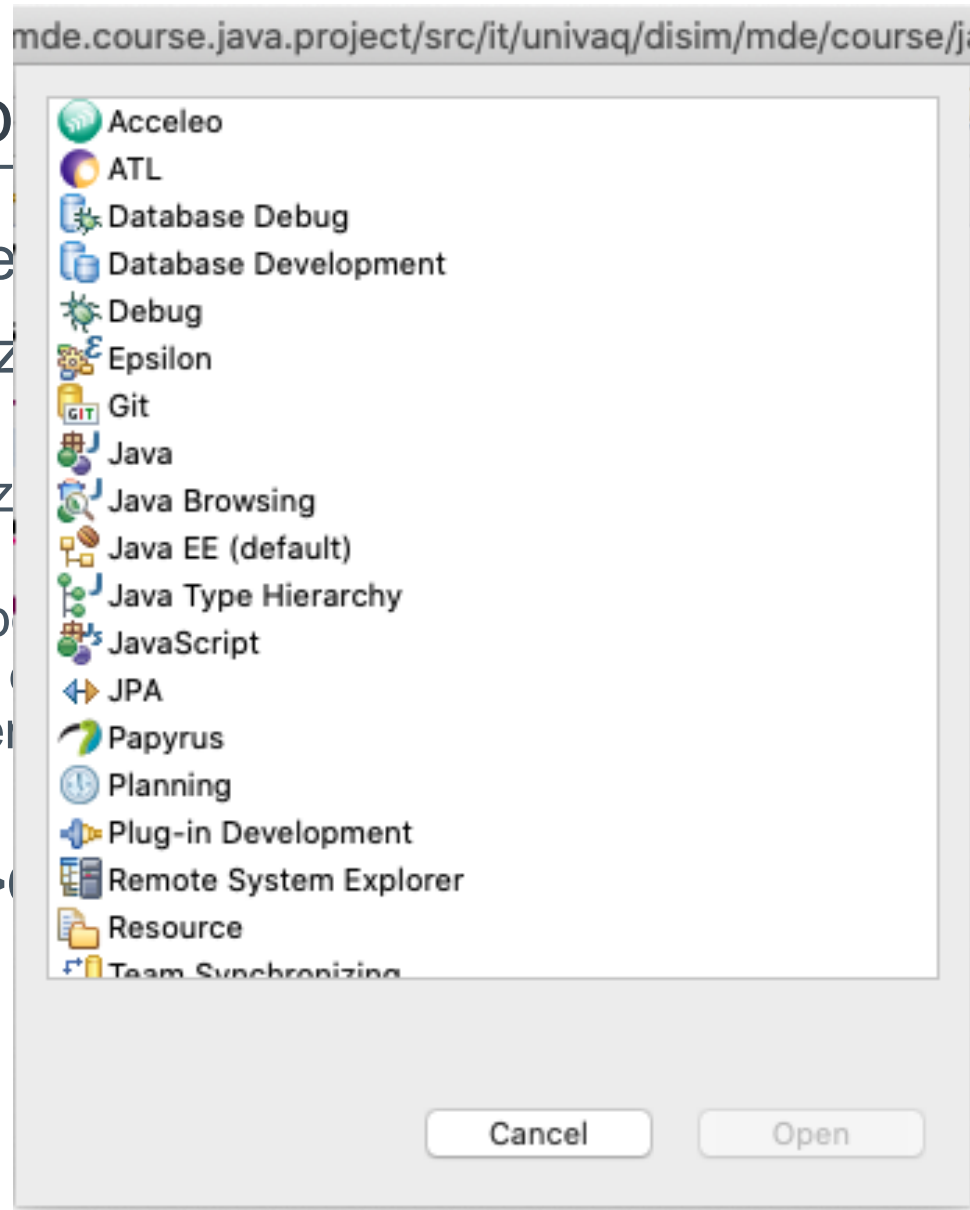
Eclipse Perspective

- › Una perspective è un container opzionale di parts
- › Può essere utilizzata per immagazzinare e organizzare diverse parts
 - Eclipse le utilizza per organizzare il tutto il layout più consono per un particolare task (debugging, sviluppo, revisioni, ecc.)
 - Gli editor già aperti vengono condivisi tra la varie perspective, se per esempio c'è un editor aperto su una certa classe nella perspective Java, e si passa alla perspective debug, l'editor rimane comunque aperto
- › Per passare tra una perspective all'altra andare su **Windows->perspective->Open Perspective->Other...**



Eclipse Persp

- › Una perspective
- › Può essere utilizzato in diverse parts
 - Eclipse le utilizza per un particolare task
 - Gli editor già aperti, per esempio c'è un editor Java, se si passa alla perspective Java, e se è aperto
- › Per passare tra perspective->perspective->



are diverse

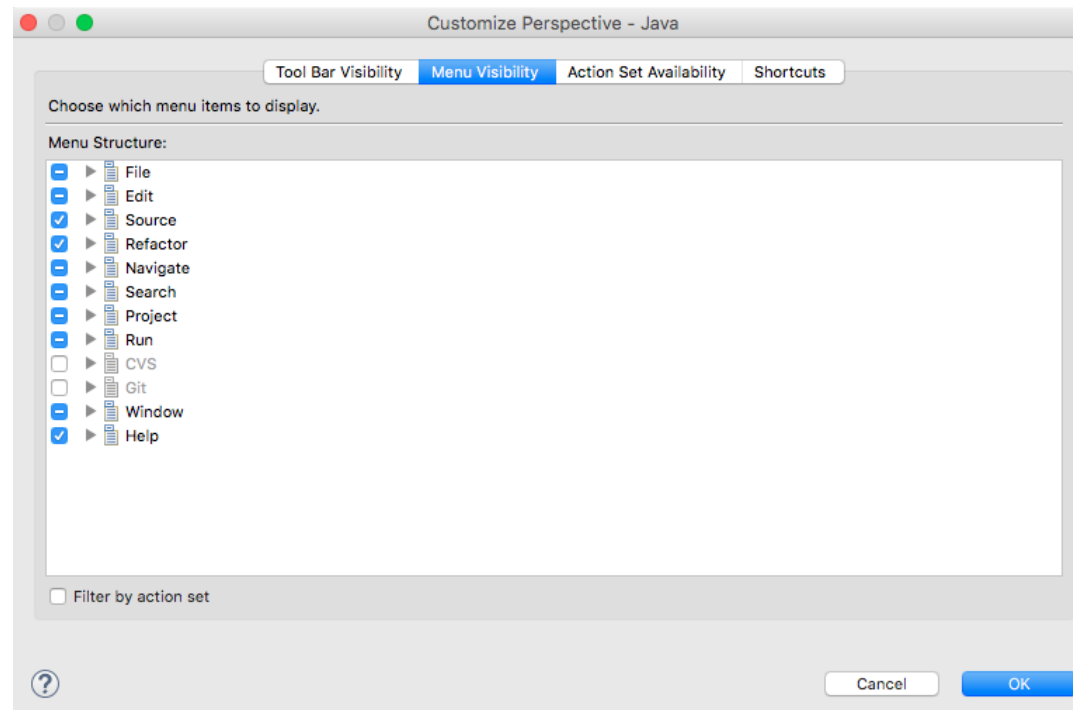
ono per un

tive, se per
erspective Java, e
e aperto

Windows-

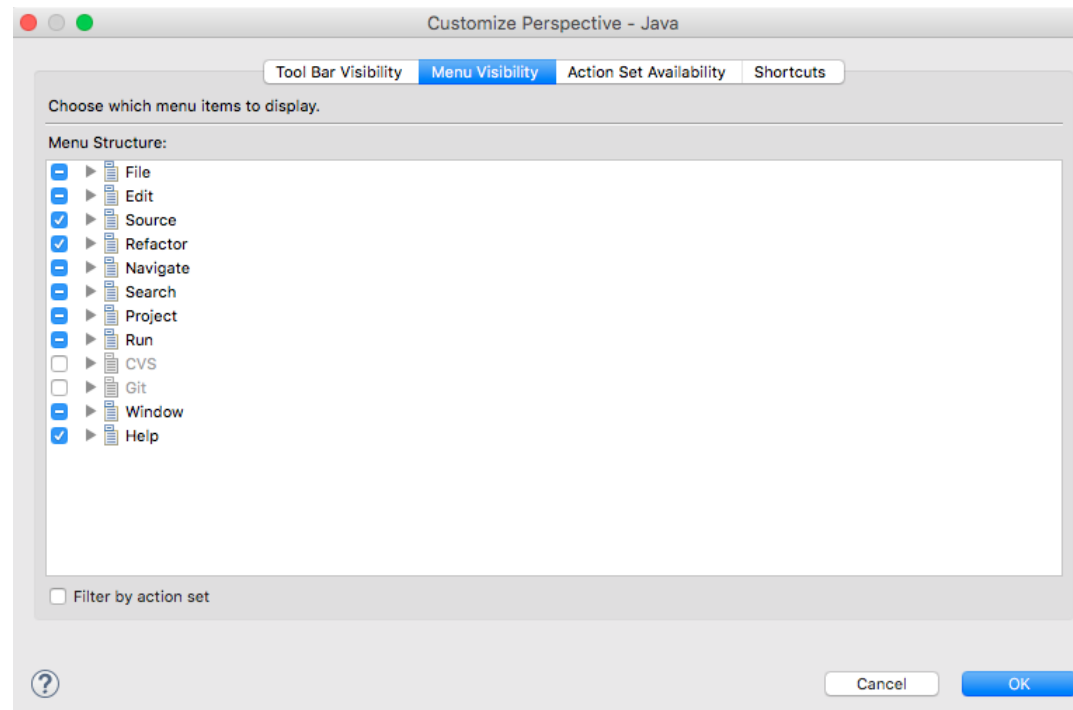
Eclipse

- › Si possono salvare le impostazioni delle perspective andando in **Window -> Save perspective as...**
- › Il menu **Window -> Customize Perspective...** Permette di modificare la perspective scelta



Eclipse

- › Si possono salvare le impostazioni delle perspective andando in **Window -> Save perspective as...**
- › Il menu **Window -> Customize Perspective...** Permette di modificare la perspective scelta



Eclipse Programmi Java

- › Creare un progetto Java minimale usando Eclipse
- › La tradizione vuole che il primo programma si limiti a scrivere sulla console la classica frase "Ciao Mondo"
- › Modificheremo leggermente questa tradizione e scriveremo qualcosa di diverso

Eclipse Creazione Progetto

- › Andare su **File -> New -> Java Project** dal menu, e inserire un nome, ad esempio: `it.gssi.eclipse.ide.first`
- › Premere su finish

Eclipse Package e Classi Java

- › Selezionare la cartella src, tasto destro su di essa e selezionare **New -> Package**
- › Tasto destro sul package e selezionare **New -> Class** e chiamarla MyFirstJavaClass e impostare le opzioni nel seguente modo

Eclipse Package e Classi Java

- > Se
Ne
- > Ta
My

The screenshot shows the 'New Java Class' dialog box in Eclipse. The 'Name' field is filled with 'MyFirstJavaClass'. The 'Source folder' is 'it.gssi.eclipse.ide.first/src' and the 'Package' is 'it.gssi.eclipse.ide.first'. The 'Enclosing type' is empty. The 'Modifiers' section has 'public' selected. The 'Superclass' is 'java.lang.Object'. The 'Interfaces' section is empty. The 'Which method stubs would you like to create?' section has 'public static void main(String[] args)' and 'Inherited abstract methods' checked. The 'Do you want to add comments?' section has 'Generate comments' unchecked. A red dashed box highlights the 'Name', 'Modifiers', 'Superclass', 'Interfaces', and 'Which method stubs would you like to create?' sections.

destro su di essa e selezionare

onare New -> Class e chiamarla
opz

The screenshot shows the 'MyFirstJavaClass.java' file in Eclipse. The code is as follows:

```
1 package it.gssi.eclipse.ide.first;
2
3 public class MyFirstJavaClass {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         System.out.println("Hello GSSI !");
8     }
9
10 }
11
```

A red dashed box highlights the line `System.out.println("Hello GSSI !");`.

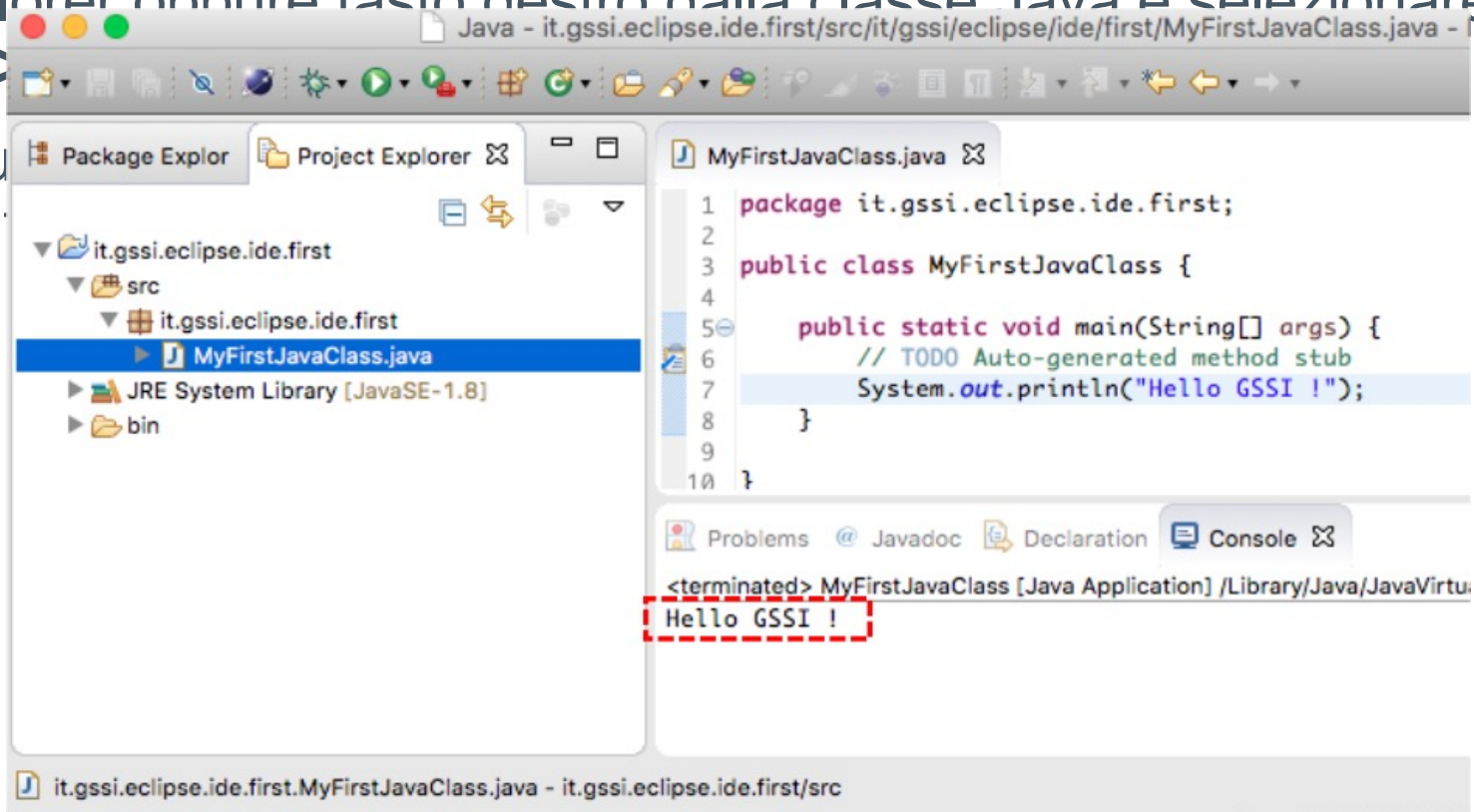
Eclipse Eseguire un programma Java

- › Premendo col tasto destro sulla vostra classe Java dal Package Explorer oppure tasto destro dalla classe Java e selezionare **Run-as -> Java Application**
- › A questo punto Eclipse eseguirà il vostro programma Java. Il risultato sarà mostrato nella view della consolle

Eclipse Eseguire un programma Java

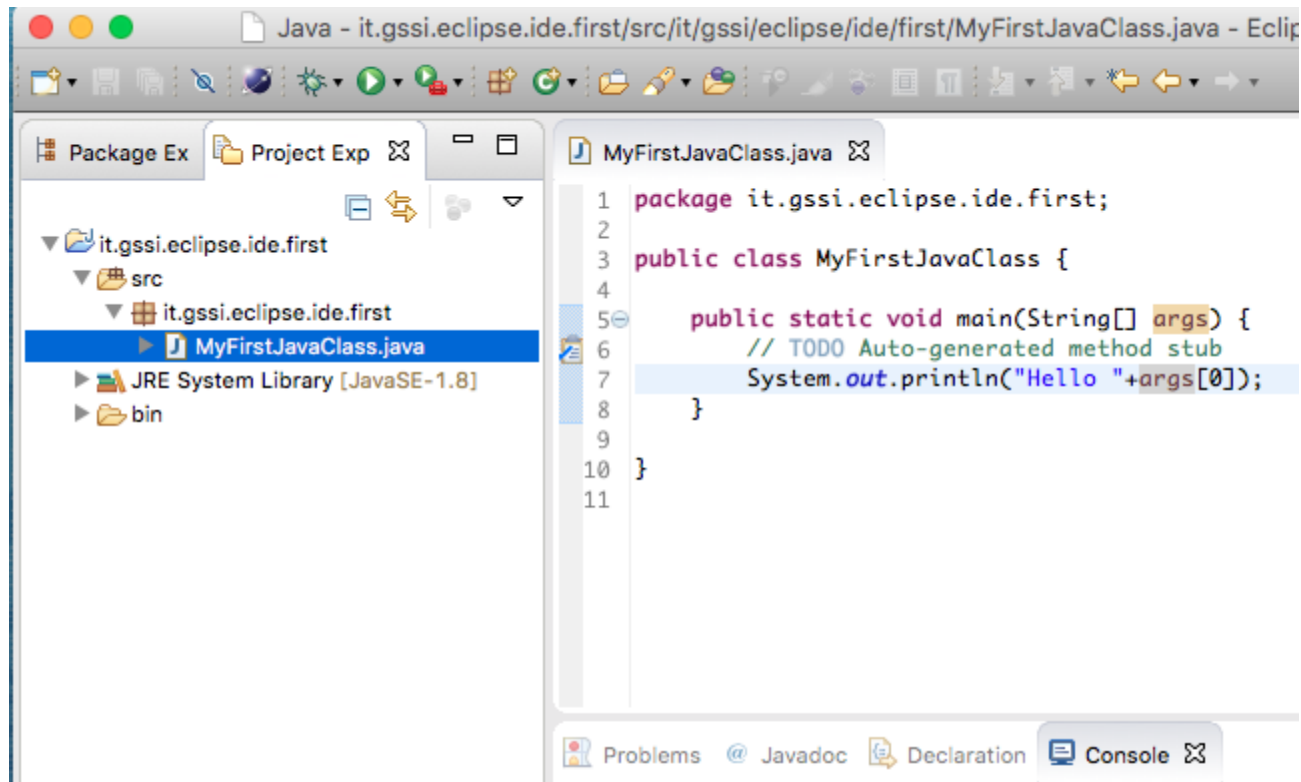
- › Premendo col tasto destro sulla vostra classe Java dal Package Explorer oppure tasto destro dalla classe Java e selezionare **Run-As -> Java Application**

- › A questo punto il risultato sarà:



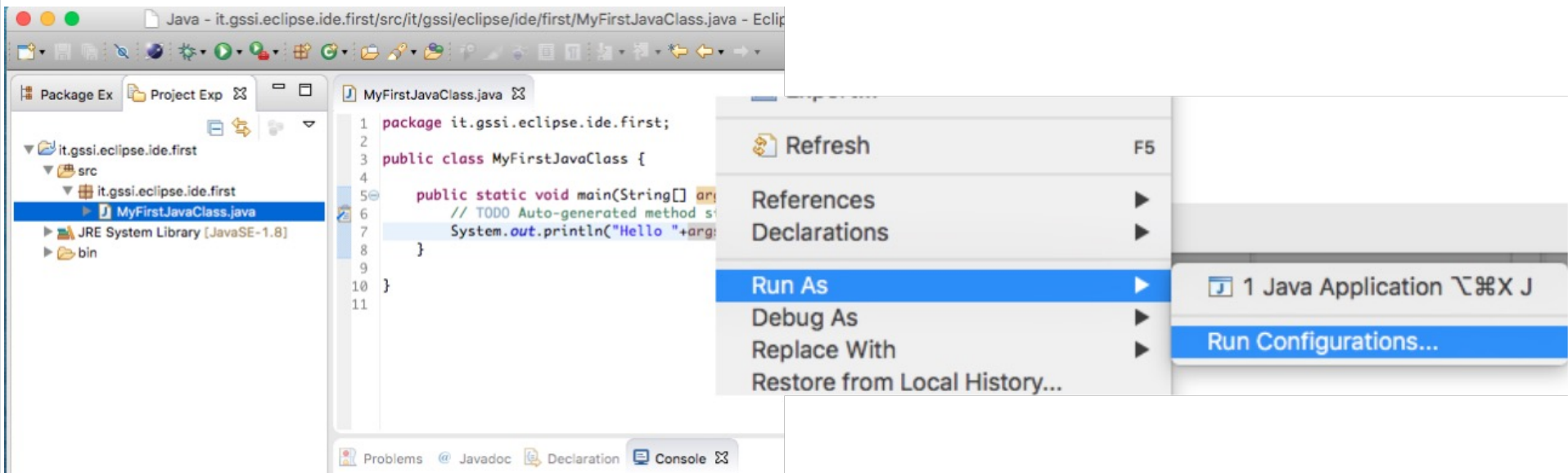
Eclipse Eseguire un programma Java

- › Modificare la stringa con qualcosa simile a questo:



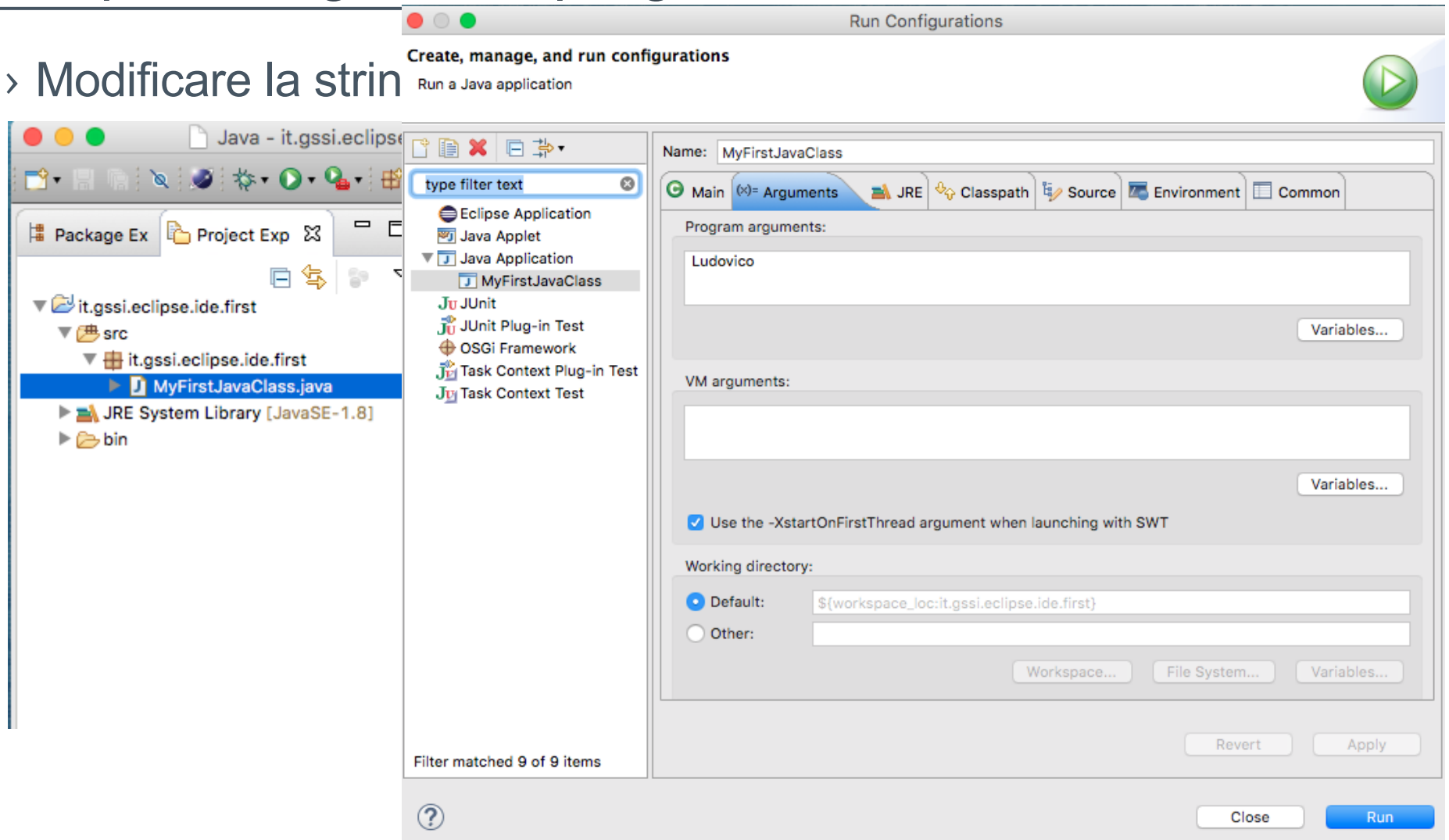
Eclipse Eseguire un programma Java

› Modificare la stringa con qualcosa simile a questo:



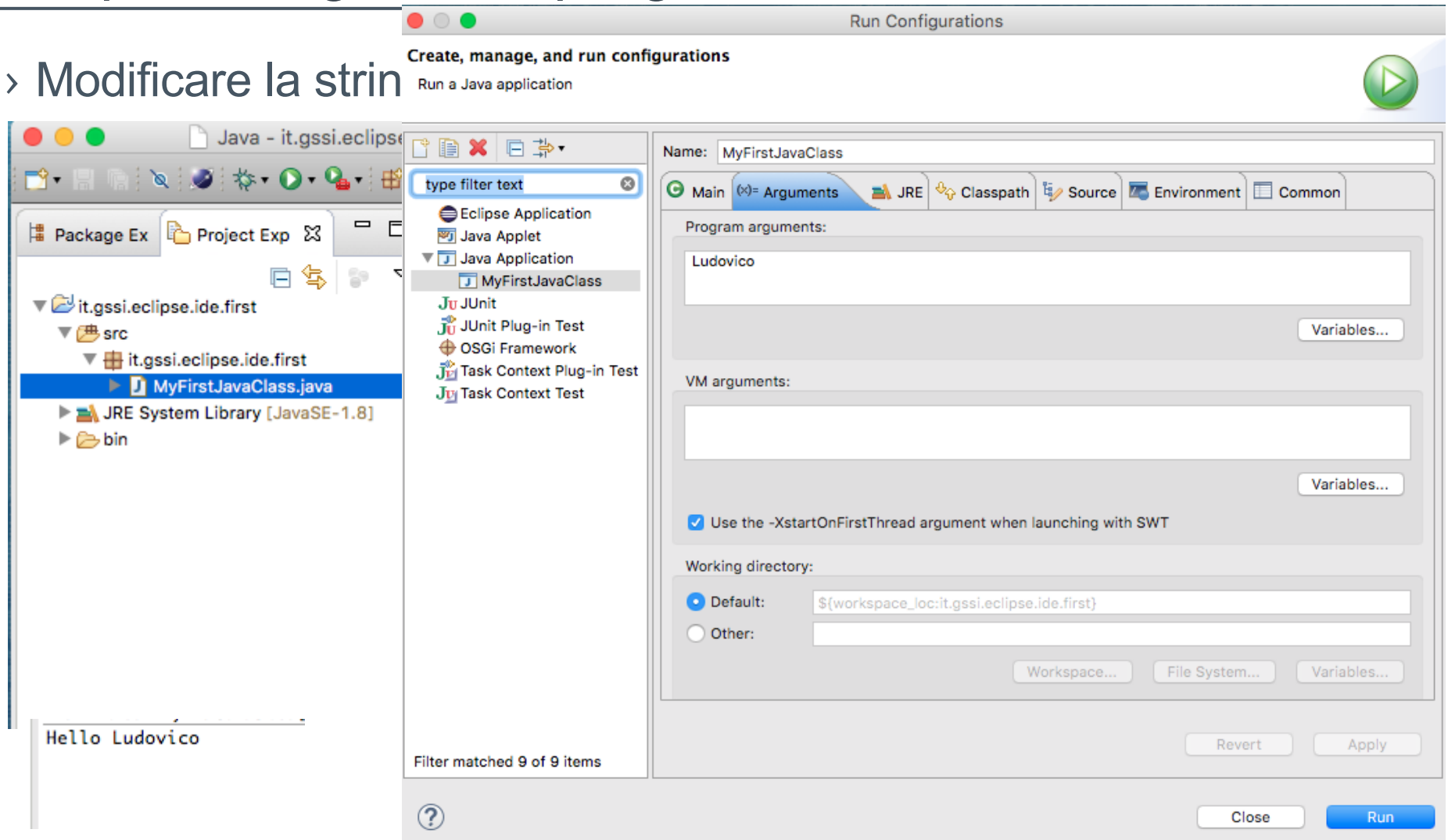
Eclipse Eseguire un programma Java

› Modificare la stringa



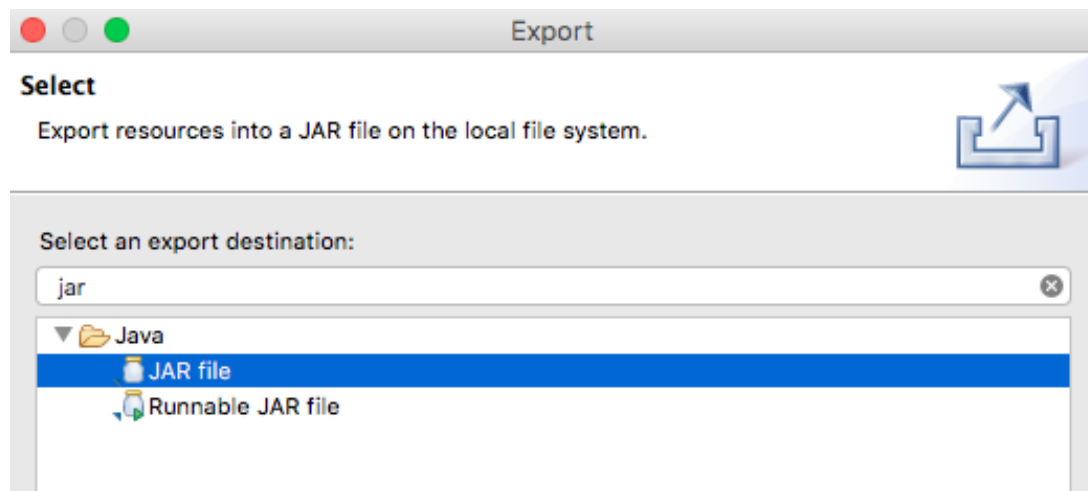
Eclipse Eseguire un programma Java

› Modificare la stringa



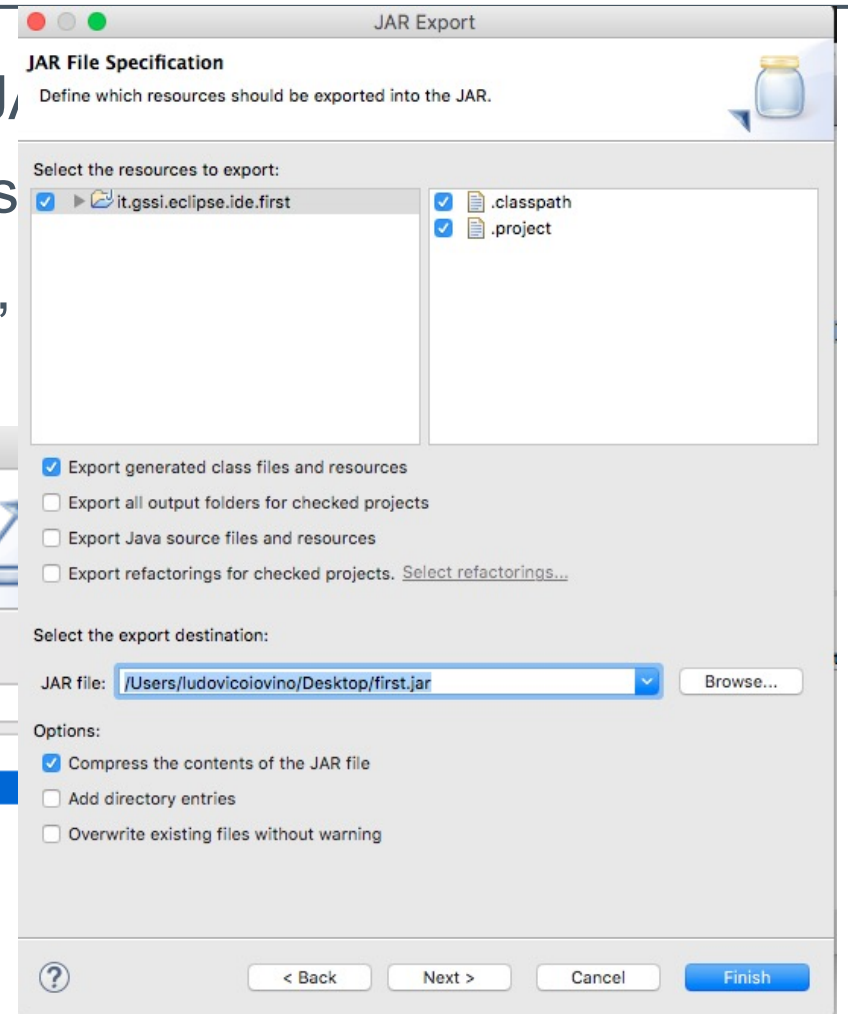
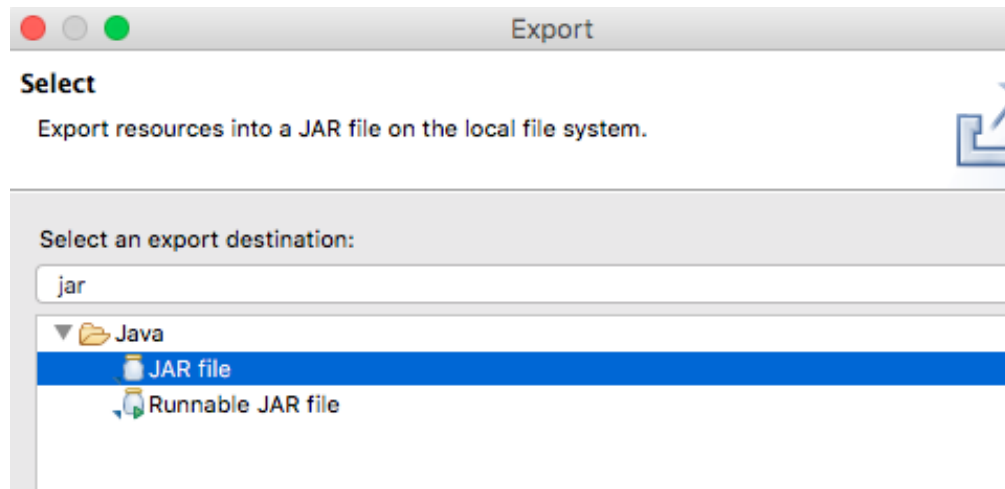
Eseguire un programma Java fuori Eclipse

- › Esportare il programma con un file JAR
- › Un file JAR è un file utilizzato per distribuire le applicazioni JAVA
- › Selezionare il progetto da esportare, tasto destro e selezione Export



Eseguire un programma Java fuori Eclipse

- › Esportare il programma con un file JAR
- › Un file JAR è un file utilizzato per distribuire un programma Java
- › Selezionare il progetto da esportare, Export



Eseguire un programma Java fuori Eclipse

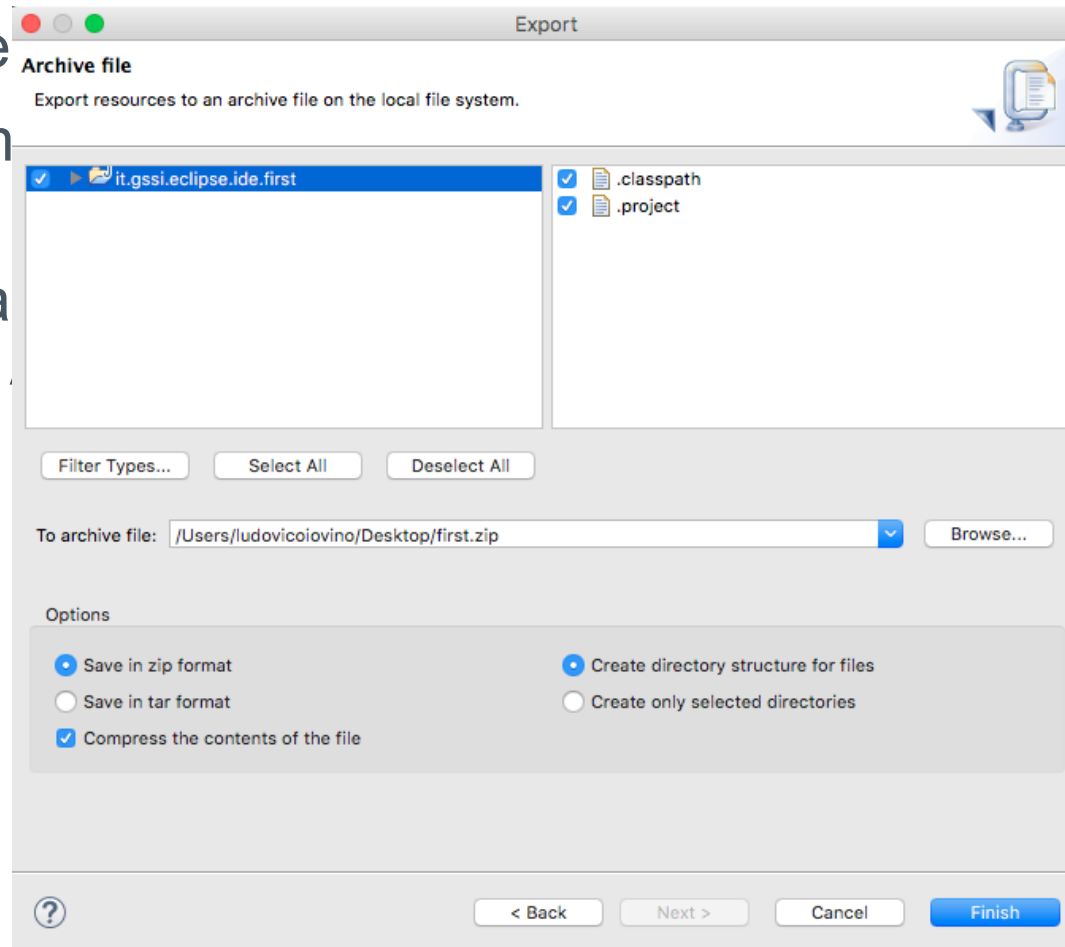
- › Esportare il programma con un file JAR
- › Un file JAR è un file utilizzato per distribuire le applicazioni JAVA
- › Selezionare il progetto da esportare, tasto destro e selezione Export
- › Aprire la console, navigate fino alla cartella di esportazione e digitare:
- › *java -classpath first.java.jar it.gssi.ide.first.MyFirstClassLudovico*

Esportare e Importare Progetti

- › In Eclipse è possibile esportare ed importare progetti
- › In questo modo è possibile condividere progetti esistenti con altre persone
- › Per esportare un progetto Eclipse, selezionare **File -> Export -> General -> Archive File** e selezionare i progetti che si desidera esportare

Esportare e Importare Progetti

- › In Eclipse è
 - › In questo m
 - › Per esporta
- General ->**
esportare



tenti con altre

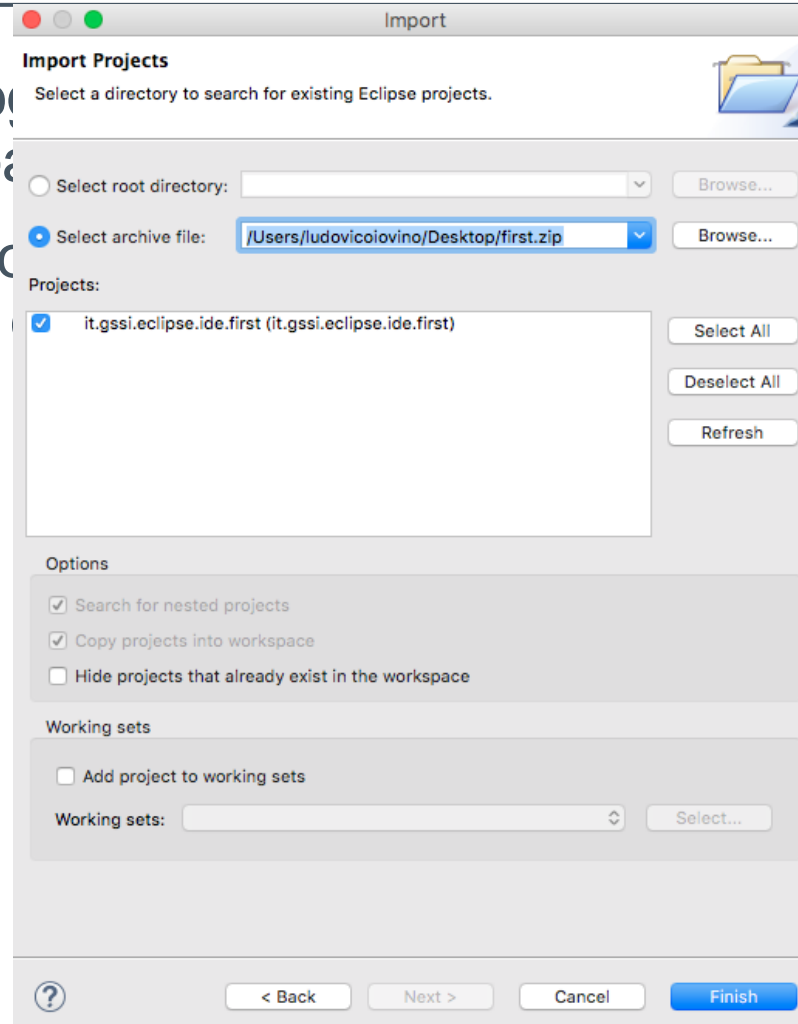
-> **Export ->**
si desidera

Esportare e Importare Progetti in Eclipse

- › Per Importare un progetto, selezionare File -> Import -> Existing Projects into Workspace
- › Possono essere importati progetti anche da file come ad esempio zip o direttamente la cartelle dove il progetto è stato aperto

Esportare e Importare Progetti in Eclipse

- › Per Importare un progetto in Eclipse
Projects into Workspace
- › Possono essere importati da un file zip o direttamente la



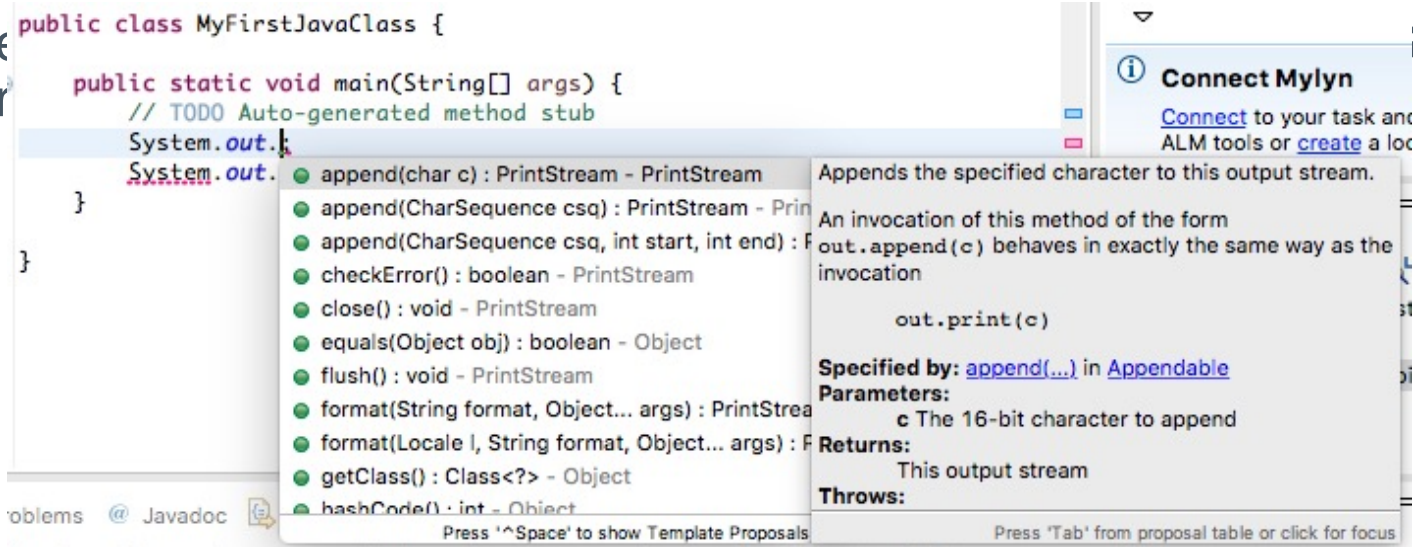
Import -> Existing

ne ad esempio
o aperto

Quick Fix e Assistenza

- › Eclipse fornisce un ottimo meccanismo di assistenza durante lo sviluppo che permette di completare il codice in maniera automatizzata in base al contesto
- › Può essere utilizzata premendo la combinazione di tasti **CTRL + SPAZIO**

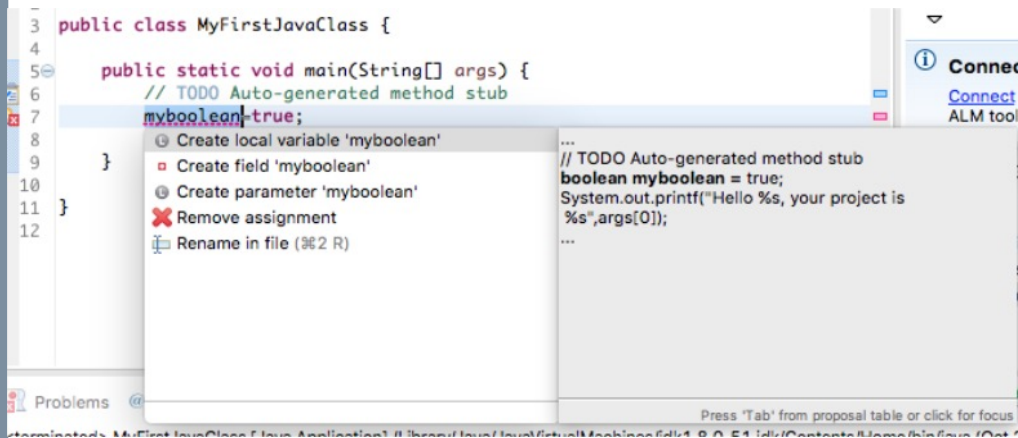
- Per esempio, la stringa



SPAZIO,

Quick Fix e Assistenza

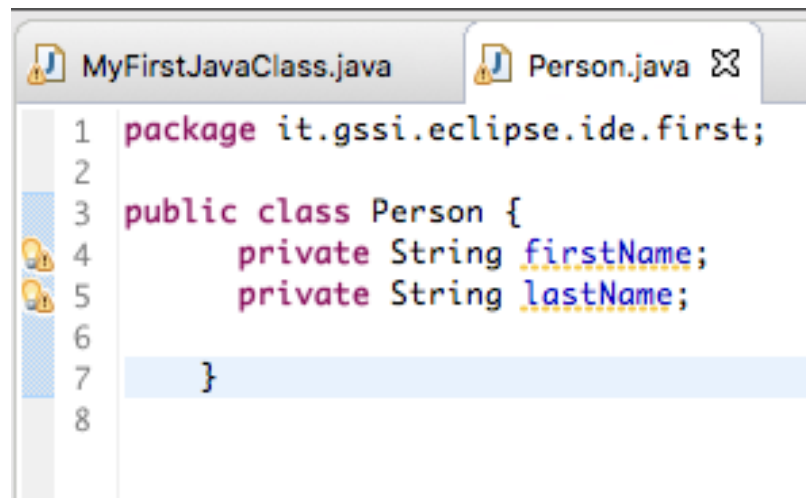
- › Ogni volta che Eclipse identifica un problema, sottolinea la riga problematica nell'editor
- › Selezionando tale riga e premendo la combinazione **CTRL+1** (CMD+1) un menu di dialogo apparirà per mostrare le soluzioni più immediate al problema. Questa funzionalità è chiamata Quick Fix



```
1 public class MyFirstJavaClass {
2
3     public static void main(String[] args) {
4         // TODO Auto-generated method stub
5         myboolean = true;
6         System.out.printf("Hello %s, your project is %s",args[0]);
7     }
8 }
9
10
11
12
```

Generazione automatizzata del codice

- › Eclipse fornisce diversi meccanismi per generare automaticamente il codice per facilitare gli sviluppatori
- › Lo scopo è di velocizzare il processo di sviluppo
 - Per esempio in Eclipse si può fare l'override dei metodi dalle superclassi e generare il metodo toString()
 - Può anche generare i getter e i setter per gli attributi definiti nella classe Java.



The screenshot shows the Eclipse IDE with two tabs: 'MyFirstJavaClass.java' and 'Person.java'. The 'Person.java' tab is active, displaying the following code:

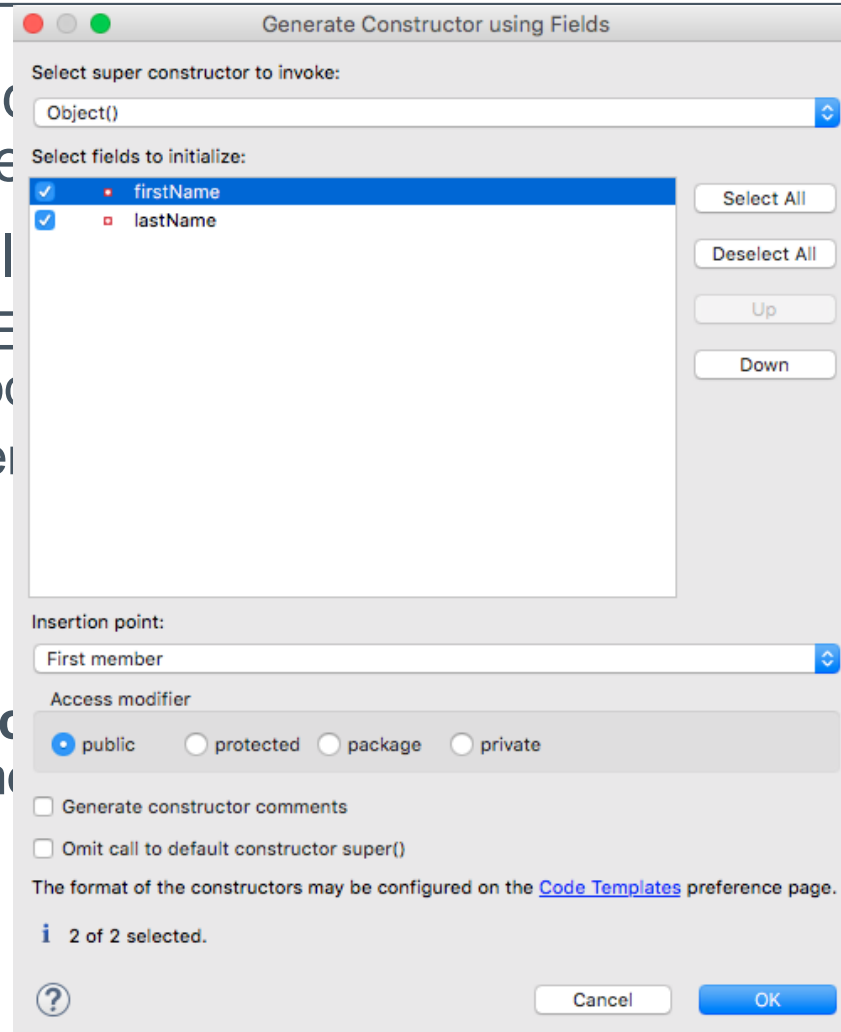
```
1 package it.gssi.eclipse.ide.first;
2
3 public class Person {
4     private String firstName;
5     private String lastName;
6
7 }
8
```

Generazione automatizzata del codice

- › Eclipse fornisce diversi meccanismi per generare automaticamente il codice per facilitare gli sviluppatori
- › Lo scopo è di velocizzare il processo di sviluppo
 - Per esempio in Eclipse si può fare l'override dei metodi dalle superclassi e generare il metodo toString()
 - Può anche generare i getter e i setter per gli attributi definiti nella classe Java.
 - Andare su **Source->Generate Constructor using Fields...**, selezionarli entrambi e premere su OK

Generazione automatizzata del codice

- › Eclipse fornisce un wizard per generare automaticamente i costruttori.
- › Lo scopo è di velocizzare la scrittura del codice.
 - Per esempio in Eclipse si può generare il metodo `main()` dalla superclasse `Runnable`.
 - Può anche generare i costruttori per le classi definite nella classe.
- Andare su **Source** -> **Generate Constructors** e premere **OK**.



e
costruttori

di dalle superclassi e
definiti nella classe

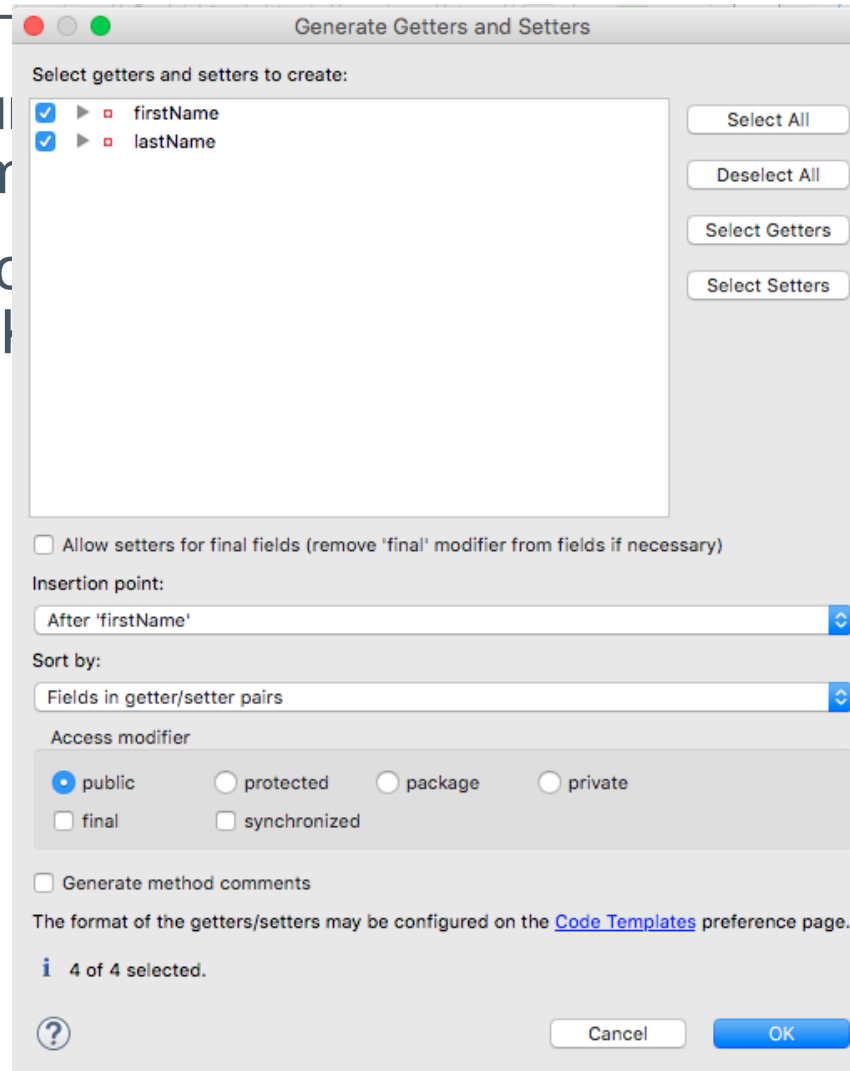
fields..., selezionarli

Generazione automatizzata del codice

- › Selezionare Source->Generate Getter and Setter, selezionare tutti i campi e poi premere OK
- › Selezionare Surce->Generate toString()..., selezionare tutti i campi e poi premere OK

Generazione automatizzata del codice

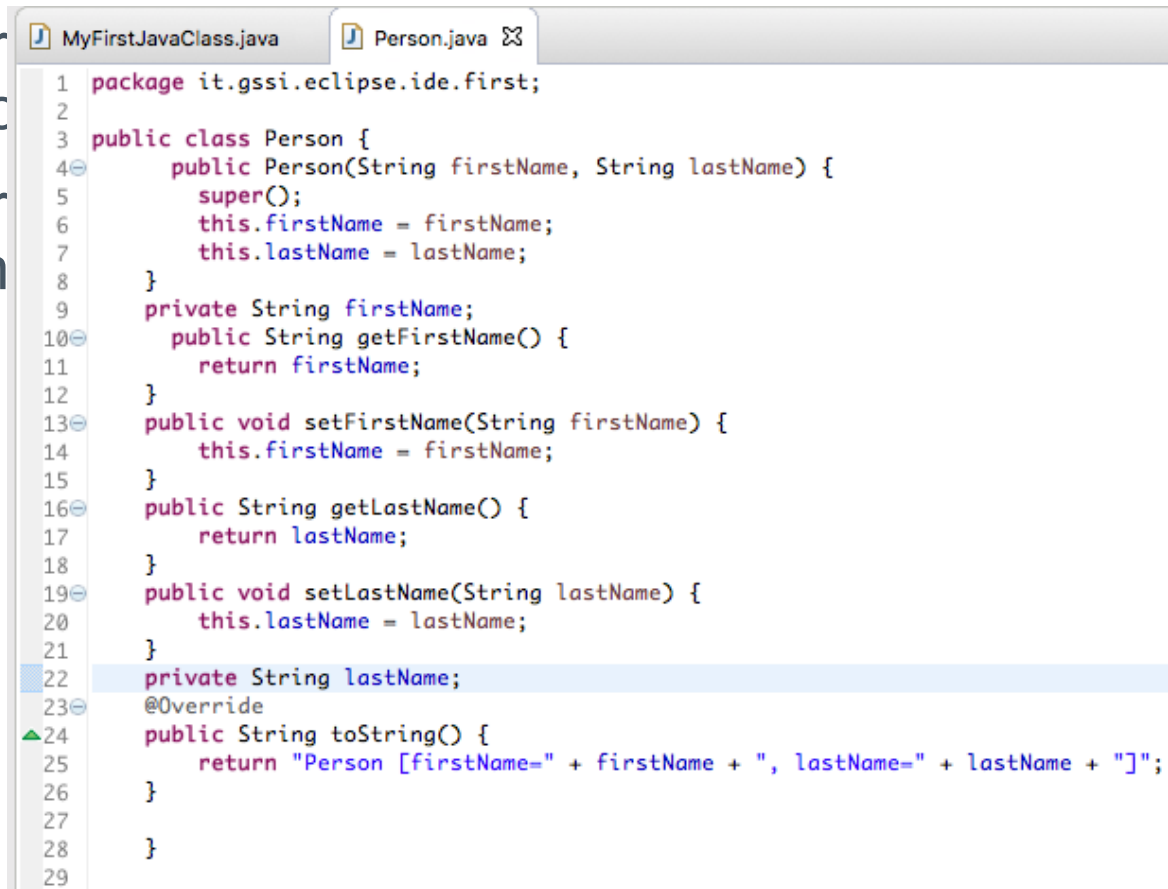
- › Selezionare Sorgenti e poi premere OK
- › Selezionare Sorgenti e poi premere OK



; selezionare tutti i
onare tutti i campi

Generazione automatizzata del codice

- › Selezionare tutti i campi e poi premere
- › Selezionare tutti i campi e poi premere



```
1 package it.gssi.eclipse.ide.first;
2
3 public class Person {
4     public Person(String firstName, String lastName) {
5         super();
6         this.firstName = firstName;
7         this.lastName = lastName;
8     }
9     private String firstName;
10    public String getFirstName() {
11        return firstName;
12    }
13    public void setFirstName(String firstName) {
14        this.firstName = firstName;
15    }
16    public String getLastName() {
17        return lastName;
18    }
19    public void setLastName(String lastName) {
20        this.lastName = lastName;
21    }
22    private String lastName;
23    @Override
24    public String toString() {
25        return "Person [firstName=" + firstName + ", lastName=" + lastName + "]";
26    }
27
28 }
29
```

Selezionare tutti i

tutti i campi

Esercizio: generazione codice e assistenza

- › Selezionare Source->Generate Constructor using Fields..., per generare un costruttore usando tutti i campi
- › Selezionare Source->Generate Getter and Setter per creare i getter e i setter per tutti i campi
- › Il risultato dovrebbe assomigliare al seguente:

Esercizio:

- › Selezionare il package e generare un progetto
- › Selezionare il package e generare i sorgenti
- › Il risultato è un progetto con i sorgenti

```
1 package it.gssi.eclipse.ide.todo;
2 import java.util.Date;
3 public class Todo {
4
5     public Todo(long id, String summary, String description, boolean done, Date dueDate) {
6         super();
7         this.id = id;
8         this.summary = summary;
9         this.description = description;
10        this.done = done;
11        this.dueDate = dueDate;
12    }
13
14    private long id;
15
16    public long getId() {
17        return id;
18    }
19
20    public void setId(long id) {
21        this.id = id;
22    }
23
24    public String getSummary() {
25        return summary;
26    }
27
28    public void setSummary(String summary) {
29        this.summary = summary;
30    }
31
32    public String getDescription() {
33        return description;
34    }
35
36    public void setDescription(String description) {
37        this.description = description;
38    }
39
40    public boolean isDone() {
41        return done;
42    }
43
44    public void setDone(boolean done) {
45        this.done = done;
46    }
47
48    public Date getDueDate() {
49        return dueDate;
50    }
51
52    public void setDueDate(Date dueDate) {
53        this.dueDate = dueDate;
54    }
55
56    private String summary = "";
57    private String description = "";
58    private boolean done = false;
59    private Date dueDate;
```

a

ids..., per

· creare i

Esercizio: generazione codice e assistenza

- › Creare una nuova classe chiamata **TodoProvider**
- › Creare il seguente metodo statico all'interno della classe **TodoProvider**

```
package it.gssi.eclipse.ide.todo;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class TodoProvider {
    private static int current = 0;

    // example data, change if you like
    public static List<Todo> createInitialModel() {
        ArrayList<Todo> list = new ArrayList<Todo>();
        list.add(createTodo("SWT", "Learn Widgets"));
        list.add(createTodo("JFace", "Especially Viewers!"));
        list.add(createTodo("DI", "@Inject looks interesting"));
        list.add(createTodo("OSGi", "Services"));
        list.add(createTodo("Compatibility Layer", "Run Eclipse 3.x"));
        return list;
    }

    private static Todo createTodo(String summary, String description) {
        return new Todo(current++, summary, description, false, new Date());
    }
}
```

Esercizio: generazione codice e assistenza

- › Scrivere una classe di test chiamata `TodoProviderTest` al cui interno vi è il metodo statico `void main(String[] args)`

```
package it.gssi.eclipse.ide.todo;

import java.util.List;

import javax.management.RuntimeErrorException;

public class TodoProviderTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List<Todo> model=TodoProvider.createInitialModel();
        if(model.size()!=5){
            throw new RuntimeException("size should be 5");
        }else{
            System.out.println("Correct");
        }
    }

}
```

Esercizio: generazione codice e assistenza

- › Eseguire con e senza la riga commentata nel TODO creator

```
// example data, change if you like
public static List<Todo> createInitialModel() {
    ArrayList<Todo> list = new ArrayList<Todo>();
    list.add(createTodo("SWT", "Learn Widgets"));
    list.add(createTodo("JFace", "Especially Viewers!"));
    list.add(createTodo("DI", "@Inject looks interesting"));
    list.add(createTodo("OSGi", "Services"));
    //list.add(createTodo("Compatibility Layer", "Run Eclipse 3.x"));
    return list;
}
```

Console

```
<terminated> TodoProviderTest [Java Application]
Correct
```

```
<terminated> TodoProviderTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_51.jdk/Contents/Home/bin/java (Nov 2, 2015, 12
Exception in thread "main" java.lang.RuntimeException: size should be 5
    at it.gssi.eclipse.ide.todo.TODOProviderTest.main(TODOProviderTest.java:13)
```

Gestione dei Task

- › In prossimità di righe di codice che richiedono attenzione si può posizionare un segnalino accessibile dalla Task View
- › Possono essere usati `//TODO`, `//FIXME` oppure `//XXX`
- › Facendo doppio click sul task si può andare direttamente in corrispondenza del codice
- › La view può essere aperta andando in **Windows -> Show View -> Taks**

Gestione dei Task

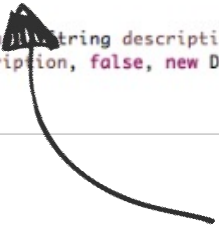
- › In prossimità di righe di codice che richiedono attenzione si può posizionare un segnalino accessibile dalla Task View

```
1 package it.gssi.eclipse.ide.todo;
2
3 import java.util.ArrayList;
4 import java.util.Date;
5 import java.util.List;
6
7 public class TodoProvider {
8     private static int current = 0;
9
10    // example data, change if you like
11    public static List<Todo> createInitialModel() {
12        ArrayList<Todo> list = new ArrayList<Todo>();
13        list.add(createTodo("SWT", "Learn Widgets"));
14        list.add(createTodo("JFace", "Especially Viewers!"));
15        list.add(createTodo("DI", "@Inject looks interesting"));
16        list.add(createTodo("OSGi", "Services"));
17        list.add(createTodo("Compatibility Layer", "Run Eclipse 3.x"));
18        //TODO Try to comment the last one and see what happens
19        return list;
20    }
21
22    private static Todo createTodo(String summary, String description) {
23        return new Todo(current++, summary, description, false, new Date());
24    }
25 }
26
```

//FIXME oppure //XXX

si può andare direttamente in

andando in **Windows -> Show View ->**

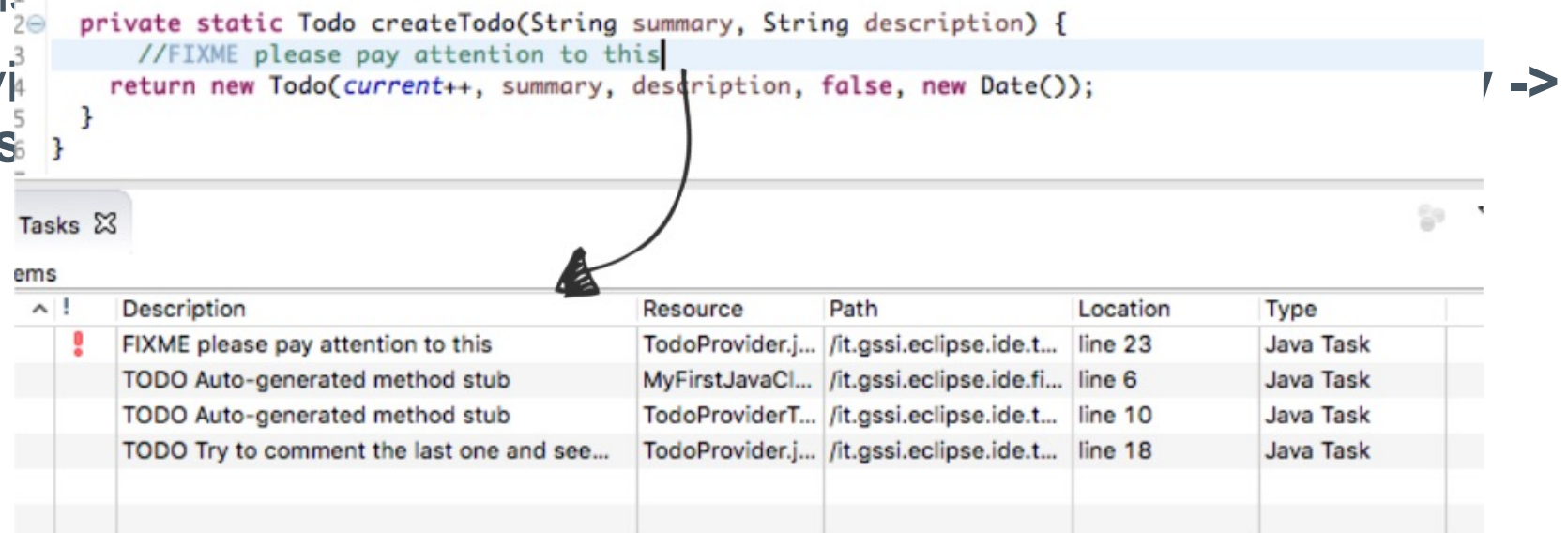


Tasks						
3 items						
✓	^	!	Description	Resource	Path	Location Type
			TODO Auto-generated method stub	MyFirstJavaCl...	/it.gssi.eclipse.ide.fi...	line 6 Java Task
			TODO Auto-generated method stub	TodoProviderT...	/it.gssi.eclipse.ide.t...	line 10 Java Task
			TODO Try to comment the last one and see...	TodoProvider.j...	/it.gssi.eclipse.ide.t...	line 18 Java Task

Gestione dei Task

- › In prossimità di righe di codice che richiedono attenzione si può posizionare un segnalino accessibile dalla Task View
- › Possono essere usati //TODO, //FIXME oppure //XXX
- › Facendo doppio click sul task si può andare direttamente in corrispondenza del codice

› La vista
Tasks



The screenshot shows the Eclipse IDE interface. The top part is the code editor, displaying a Java method `createTodo`. A line of code is highlighted in blue: `//FIXME please pay attention to this`. Below the code editor is the **Tasks** view, which is currently active. It shows a list of tasks. The first task, `FIXME please pay attention to this`, is highlighted in red. An arrow points from this task to the corresponding line in the code editor. The **Tasks** view has a tab labeled **Tasks** and a search icon. The tasks list has columns for **Description**, **Resource**, **Path**, **Location**, and **Type**.

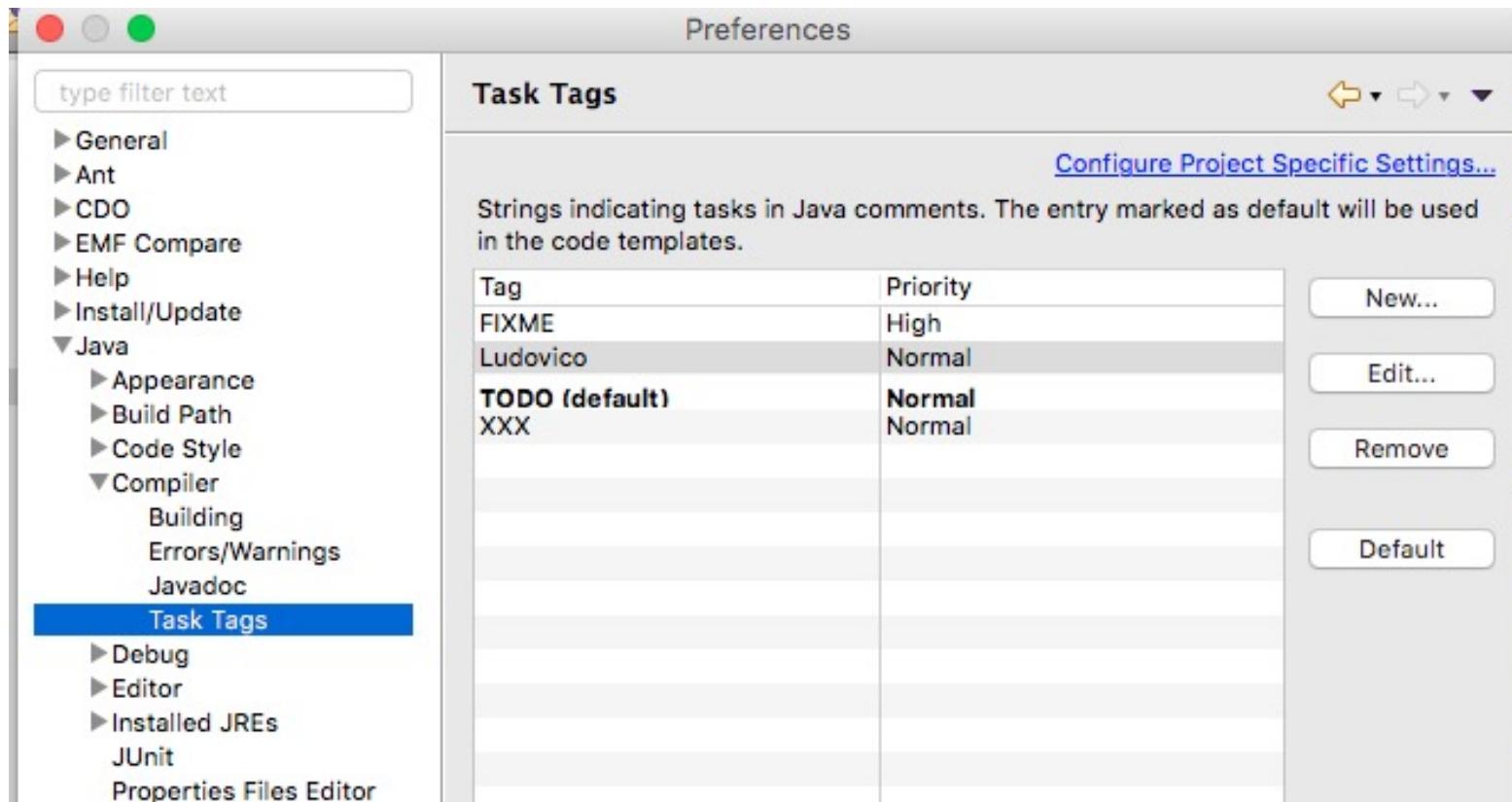
Description	Resource	Path	Location	Type
FIXME please pay attention to this	TodoProvider.j...	/it.gssi.eclipse.ide.t...	line 23	Java Task
TODO Auto-generated method stub	MyFirstJavaCl...	/it.gssi.eclipse.ide.fi...	line 6	Java Task
TODO Auto-generated method stub	TodoProviderT...	/it.gssi.eclipse.ide.t...	line 10	Java Task
TODO Try to comment the last one and see...	TodoProvider.j...	/it.gssi.eclipse.ide.t...	line 18	Java Task

Gestione dei Task: Task Personalizzati

- › Si possono definire i propri task personalizzati andando in Window
-> Preferences -> Java -> Compiler -> Task Tags

Gestione dei Task: Task Personalizzati

- › Si possono definire i propri task personalizzati andando in Window -> Preferences -> Java -> Compiler -> Task Tags



Eclipse Shortcuts (alcune)

- › **CTRL+Maiusc+R**: Interfaccia di dialogo per cercare risorse
- › **CTRL+Maiusc+T**: Interfaccia di dialogo per cercare classi, interfacce, enum
- › **CTRL+1**: Mostra le soluzioni per un problema (sul cursore)
- › **CTRL+Spazio**: Interfaccia dialogo assistenza
- › **CTRL+F**: Cerca sul file corrente
- › **CTRL+H**: Cerca su tutto il workspace
- › **CTRL+7** o **CTRL+ Maiusc +c**: Commenta una riga
- › **CTRL+Maiusc+F**: Riorganizza il codice
- › **CTRL+Maiusc+O**: Organizza gli import