

# Eclipse - Nozioni Base

## Programmazione e analisi di dati

### Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa  
<http://www.di.unipi.it/~milazzo>  
[milazzo@di.unipi.it](mailto:milazzo@di.unipi.it)

Corso di Laurea Magistrale in Informatica Umanistica  
A.A. 2015/2016

# Eclipse

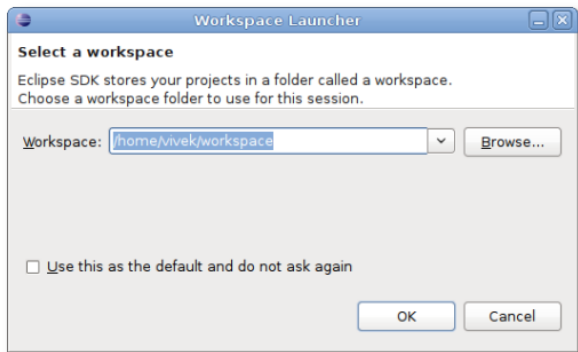
Eclipse è un **ambiente di sviluppo integrato** (Integrated Development Environment – IDE)

- Racchiude in un unico ambiente tutti gli strumenti che servono a un programmatore
- Editor, compilatore, debugger, ....

Eclipse è uno tra i principali IDE disponibili al momento

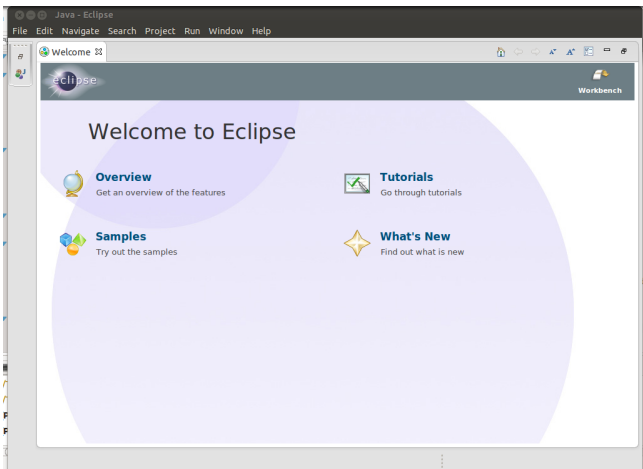
- E' tra i più usati in ambiente aziendale
- Può essere usato per programmare con molti linguaggi diversi (non solo Java)

Appena si avvia Eclipse compare la seguente finestra



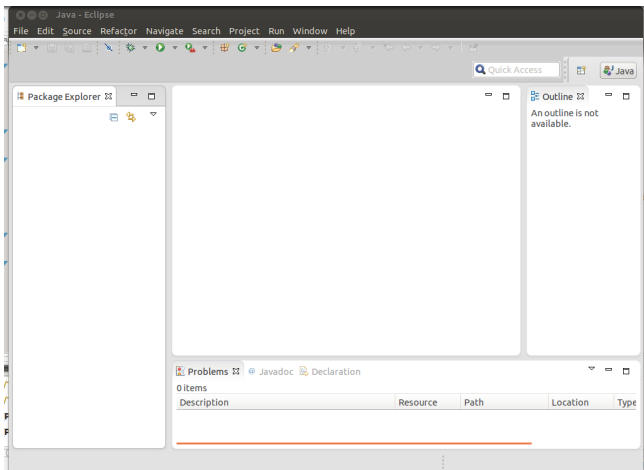
che ci chiede di specificare (o semplicemente confermare) la cartella da utilizzare come “workspace”, ossia in cui verranno salvati tutti i programmi che realizzeremo.

La prima volta che eseguiamo Eclipse compare una schermata di benvenuto



che possiamo chiudere cliccando su "Workbench".

Questa è la schermata principale di Eclipse



Ogni area della schermata principale di Eclipse è detta **Vista** (View)

- La vista centrale ci consentirà di scrivere il nostro programma
- La vista “Package Explorer” (a sinistra) mostrerà tutti i file creati
- La vista “Outline” (a destra) mostrerà alcune informazioni sulla classe corrente
- La vista “Problems” (in basso) riporterà eventuali errori di compilazione
- La vista “Console” (non in figura) ci consentirà di interagire con il programma in esecuzione
- .....

Un'insieme di viste prende il nome di **Prospettiva** (Perspective)

- Vedremo che oltre alla prospettiva mostrata in figura (Java) ne utilizzeremo un'altra (Debug) che include altre viste

Per poter scrivere un programma dobbiamo innanzitutto creare un **progetto**.

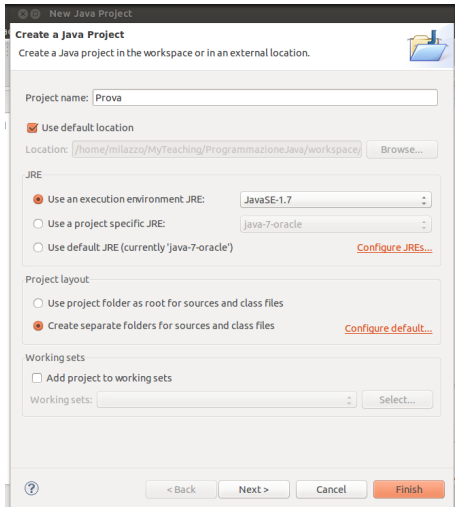
Un progetto sostanzialmente è un contenitore di classi Java che sono in qualche modo collegate tra loro

- Quando si realizza un programma complesso di solito si crea un progetto specifico che conterrà tutte le sue classi
- Noi potremmo creare un progetto per raccogliere tutte le classi realizzate nell'ambito di una lezione in laboratorio

Per creare un progetto:

File --> New --> Java project

Si apre la seguente finestra:



In cui inseriamo il nome del progetto (ad esempio Prova) e confermiamo con **Finish**



A questo punto dobbiamo creare la prima classe Java da inserire nel progetto Prova

Per creare una classe: File --> New --> Class

Si apre la seguente finestra:

New Java Class

**Java Class**

⚠ The use of the default package is discouraged.

Source Folder: Prova/src

Package: (default)

Enclosing type:

Name: Esempio

Modifiers:  public  default  private  protected

abstract  final  static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

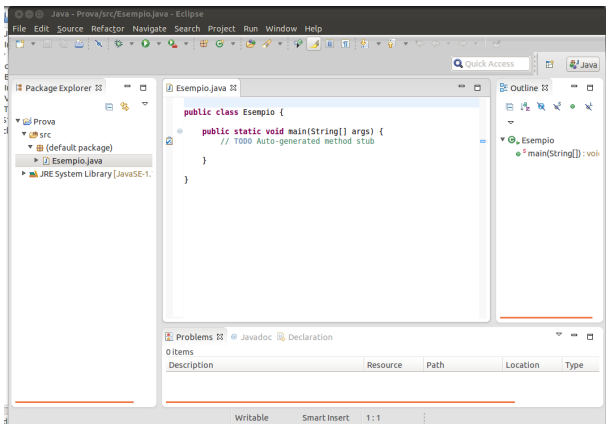
Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

In cui inseriamo il nome della classe (ad esempio Esempio).  
Possiamo (opzionalmente) scegliere di creare il metodo main (facciamolo, in questo caso)

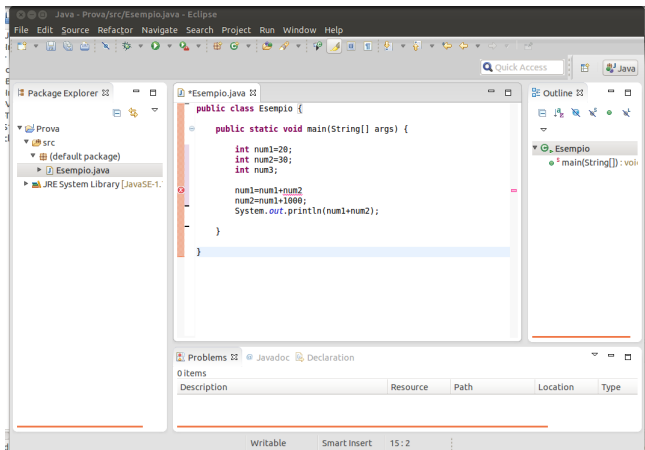
Ci troviamo ora in questa situazione...



...con il codice della nostra classe al centro, già parzialmente scritto!  
La riga con il TODO è un commento automatico che possiamo anche cancellare

A sinistra, nel **Package Explorer** troviamo (tra le altre cose) l'elenco dei file che sono stati creati. In questo caso: `Esemplio.java`.

## Scriviamo un programma di prova nel main:



L'editor di Eclipse ci segnala alcuni **errori** in tempo reale sottolineandoli in rosso (in figura manca un punto e virgola)

Vengono invece sottolineati in giallo situazioni anomale (non necessariamente errori) dette **warning**

Una volta corretti eventuali errori possiamo **compilare ed eseguire** il programma tramite:

Run --> Run

oppure, più semplicemente, cliccando sull'icona a forma di pallina verde con il triangolino bianco nella barra in alto



```
public class Esempio {  
    public static void main(String[] args) {  
        int num1=20;  
        int num2=30;  
        int num3;  
  
        num1=num1+num2;  
        num2=num1+1000;  
        System.out.println(num1+num2);  
    }  
}
```

<terminated> Esempio [Java Application] /usr/lib/jvm/java-7-oracle/bin/java (Oct 7, 2013, 6:53:15 PM)  
1100

Il risultato dell'esecuzione (1100) è nella vista **Console** (se non si apre in automatico la si può aprire con **Window --> Show view --> Console**)

Anche l'eventuale input viene richiesto all'utente nella vista **Console**

Uno strumento molto importante fornito da Eclipse è il **debugger**

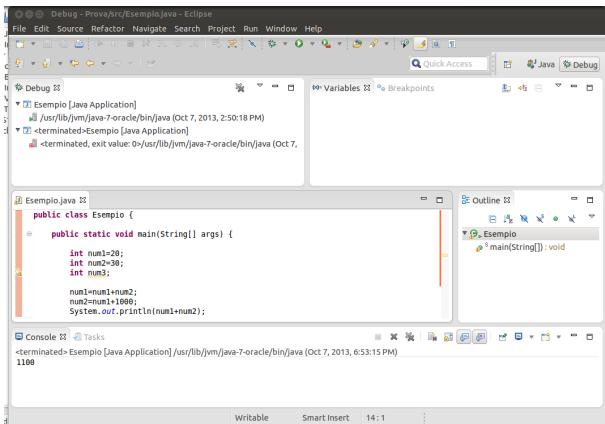
Il debugger consente di far interrompere l'esecuzione del nostro programma in un punto prescelto

- una volta interrotto, potremo vedere il valore delle tutte variabili in quel momento
- potrem inoltre far procedere il programma un passo alla volta, monitorando la situazione

Il debugger è uno strumento essenziale per ricercare errori nei programmi

Useremo il debugger anche come **strumento didattico**, per capire meglio cosa fanno i vari comandi del linguaggio!

Per usare il debugger bisogna innanzitutto **cambiare prospettiva**  
Window --> Open perspective --> Debug





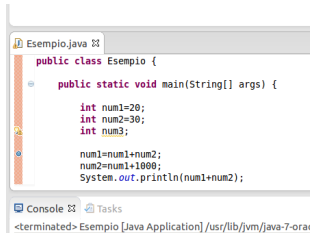
Ora scegliamo il **breakpoint**, ossia il punto del programma in cui vogliamo interrompere l'esecuzione.

Per fare ciò si clicca con il tasto destro nella barra verticale a sinistra, all'altezza della riga in cui vogliamo fermarci.

- Nell'esempio, la riga `num1=num1+num2;`

Si apre il menù contestuale da cui selezioniamo la voce "Toggle breakpoint".

Come risultato, comparirà un pallino blu nel punto in cui abbiamo cliccato



```
Esempio.java
public class Esempio {
    public static void main(String[] args) {
        int num1=20;
        int num2=30;
        int num3;

        num1=num1+num2;
        num2=num1+1000;
        System.out.println(num1+num2);
    }
}
```

Console Tasks  
<terminated> Esempio [Java Application] /usr/lib/jvm/java-7-oracle

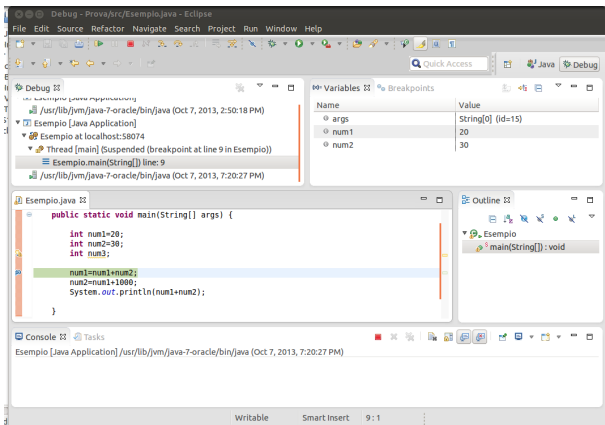
Ora facciamo partire il debugger tramite:

Run --> Debug

oppure, più semplicemente, cliccando sull'icona a forma di scarafaggio  
nella barra in alto



Partirà l'esecuzione del programma e si fermerà esattamente dove richiesto



In alto a destra (nella vista **Variable**) sono visibili tutte le variabili e i loro valori

Si può procedere passo passo nell'esecuzione tramite:  
Run --> Step over o più semplicemente cliccando sull'icona  
corrispondente nella barra in alto



In qualunque momento si può far ripartire l'esecuzione o terminarla  
definitivamente usando gli appositi controlli nella barra in alto



Una volta concluso il debug si può **cancellare il Breakpoint** cliccandoci di  
nuovo sopra con il tasto destro e selezionando "Toggle Breakpoint"

Infine si può tornare alla prospettiva standard tramite  
Window --> Open perspective --> Java