Eclipse - Nozioni Base Programmazione e analisi di dati Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa http://pages.di.unipi.it/milazzo milazzo@ di.unipi.it

Corso di Laurea Magistrale in Informatica Umanistica A.A. 2019/2020

Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 1/20



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)

イロト イポト イヨト イヨト 二日

Appena si avvia Eclipse compare la seguente finestra

Workspace Launcher	
Select a workspace	
Eclipse SDK stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: [/home/vivek/workspace v	<u>B</u> rowse
□ <u>U</u> se this as the default and do not ask again OK	Cancel

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".

Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 4 / 20

イロト 不得 トイヨト イヨト 二日

Questa è la schermata principale di Eclipse



Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 5/20

▲□▶ ▲圖▶ ▲ 圖▶ ▲ 圖▶ ― 圖 … のへで

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

- La vista centrale ci consentira 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

イロト 不得 トイラト イラト 一日

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:

🛛 🐵 New Java Project	
Create a Java Project	
Create a Java project in the workspace or in an e	xternal location.
Project name: Prova	
✓ Use default location	
Location: //home/milazzo/MyTeaching/Progra	mmazioneJava/workspace/ Browse
IDE	
Itse an everytion environment IDE:	InvaSE-17
O Use an execution environment SRE.	JavaSE-1.7
O use a project specific JRE:	Java-7-oracle
 Use default JRE (currently 'java-7-oracle') 	Configure JREs
Project layout	
O Use project folder as root for sources and	d class files
Oreate separate folders for sources and c	lass files <u>Configure default</u>
Working sets	
Add project to working sets	
Working sets:	\$ Select
()	Next > Cancel Finish

In cui inseriamo il nome del progetto (ad esempio Prova) e confermiamo con Finish Paolo Milazzo (Università di Pisa) Programmazione - Background A.A. 2019/2020 8/20 A questo punto dobbiamo creare la prima classe Java da inserire nel progetto Prova

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

Paolo Milazzo (Università di Pisa)

Programmazione - Background

 ■ ▶ ▲ ■ ▶ ■
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●
 ●

イロト イポト イヨト イヨト

Si apre la seguente finestra:

source rolder:	Prova/src	Browse
Package:	(default)	Browse
Enclosing type:		Browse
Name:	Esempio	
Modifiers:	public O default O private O protected	
	abstract final static	
Superclass:	java.lang.Object	Browse
nterfaces:		Add
		Remove
which method stub	s 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 <u>here</u>)	

In cui inseriamo il nome della classe (ad esempio Esempio). Possiamo (opzionalmente) scegliere di creare il metodo main (facciamolo, in questo caso) Paolo Milazzo (Università di Pisa) Programmazione - Background A.A. 2019/2020 10/20

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: Esempio.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

Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 12/20

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



Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 13 / 20

3

- 4 回 ト 4 ヨ ト 4 ヨ ト



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

Paolo Milazzo (Università di Pisa)

Programmazione - Background

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
- potremo 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 lingauggio!

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



Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 16 / 20

イロト イポト イヨト イヨト 二日

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



Console X 2 Tasks
<terminated> Esempio [Java Application] /usr/lib/jvm/java-7-orac

Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 17/20

< ロ > < 同 > < 回 > < 回 > < 回 > <

Ora facciamo partire il debugger tramite:

Run --> Debug

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



Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 18 / 20

3

- 4 回 ト 4 ヨ ト 4 ヨ ト

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

File Edit Source Refactor Navigate Search Project Run Window Help 💼 🕶 🔕 💩 💩 🐽 🗰 😥 👧 😓 😓 😓 🗮 💥 📚 🐨 🚱 🕶 🚱 🖌 🐼 😹 👘 Q Oulck Access 🗈 🖏 Java 🕸 Debug V • • 🍄 Debug 🛙 🕫 Variables 🗱 💁 Breakpoints 20 AL D Name Value /usr/lib/jvm/java-7-oracle/bin/java (Oct 7, 2013, 2:50:18 PM) o aros String[0] (id=15) Esempio [Java Application] 9 num1 20 * 🔐 Esempio at localhost:58074 30 ▼ → Thread [main] (Suspended (breakpoint at line 9 in Esempio)) Esempio.main(String[]) line: 9 /usr/lib/ivm/iava-7-oracle/bin/iava (Oct 7, 2013, 7:20:27 PM) Esempio.java S • m Cutline 🛙 - public static void main(String[] args) { E 15 N × • ▼ 💬 Esempio int numl=20: int num2=30: main(String[]): void int num3: numl=numl+num2; num2=num1+1080: System.out.println(num1+num2); 😑 Console 🗱 🥥 Tasks 🔳 🗶 🔆 🖬 🖍 🎜 🖉 🎮 M 🛛 🗸 🗂 🗸 🗖 🗆 Esempio [Java Application] /usr/lib/jvm/java-7-oracle/bin/java (Oct 7, 2013, 7:20:27 PM) Smart Insert 9:1

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

Paolo Milazzo (Università di Pisa)

Programmazione - Background

A.A. 2019/2020 19/20

(日) (四) (日) (日) (日)

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

・ロト ・ 母 ト ・ ヨ ト ・ ヨ ト