

Eclipse IDE, Debug

Programmazione ad Oggetti – Lab03

Docenti: Danilo **Pianini**, Roberto **Casadei**
Tutor: Simone **Costanzi**, Luca **Tremamunno**

C.D.S. Ingegneria e Scienze Informatiche
ALMA MATER STUDIORUM—Università di Bologna, Campus di Cesena

12 ottobre 2021



- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



1 Integrated Development Environments

2 Eclipse IDE per Java

- Astrazioni di Base
- Refactoring e generazione di codice
- Keyboard Shortcuts

3 Debugging di Applicazioni

4 Lab Startup



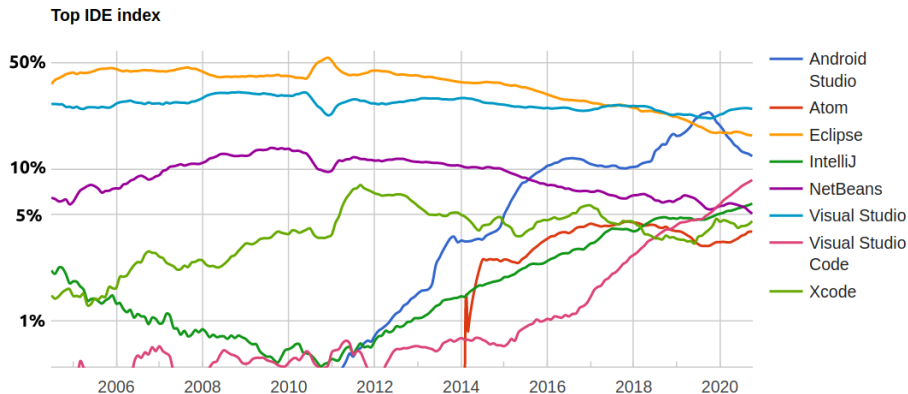
Integrated Development Environment (IDE)

A software suite that consolidates the basic tools developers need to write and test software. An IDE contains a code editor, a compiler and a debugger that the developer accesses through a single user interface.

- Ambiente integrato per la creazione e la gestione di progetti software.
 - ▶ Un progetto, per un IDE, è composto da una collezione organizzata di risorse: sorgenti, librerie, compilati, documentazione, ...
- Componenti tipici di un IDE:
 - ▶ Editor di testo con syntax highlighting (e code completion)
 - ▶ Compilatore
 - ▶ Macchina/e virtuale/i per l'esecuzione e il debugging delle applicazioni
 - ▶ Strumenti per agevolare il deployment (distribuzione) delle applicazioni
- Esempi di IDE:
 - ▶ Eclipse, NetBeans, IntelliJ IDEA, Microsoft Visual Studio, XCode, ...



Diffusione dei principali IDE



Dati da <https://pypl.github.io/IDE.html>



1 Integrated Development Environments

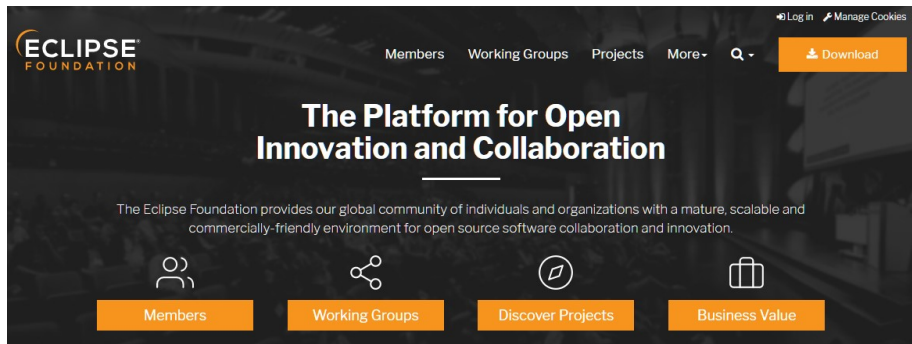
2 Eclipse IDE per Java

- Astrazioni di Base
- Refactoring e generazione di codice
- Keyboard Shortcuts

3 Debugging di Applicazioni

4 Lab Startup





The screenshot shows the Eclipse Foundation website. At the top left is the Eclipse Foundation logo. The top navigation bar includes links for 'Members', 'Working Groups', 'Projects', 'More', a search icon, and a 'Download' button. The main heading reads 'The Platform for Open Innovation and Collaboration'. Below this, a paragraph states: 'The Eclipse Foundation provides our global community of individuals and organizations with a mature, scalable and commercially-friendly environment for open source software collaboration and innovation.' At the bottom, there are four orange buttons with icons and labels: 'Members' (person icon), 'Working Groups' (network icon), 'Discover Projects' (diamond icon), and 'Business Value' (briefcase icon).

ECLIPSE
FOUNDATION

Log in Manage Cookies

Members Working Groups Projects More - Q - Download

The Platform for Open Innovation and Collaboration

The Eclipse Foundation provides our global community of individuals and organizations with a mature, scalable and commercially-friendly environment for open source software collaboration and innovation.

Members Working Groups Discover Projects Business Value

`https://www.eclipse.org/`



Overview

- **Eclipse** è un **IDE open-source** scritto in **Java**
- Disponibile per diverse **piattaforme** (Windows, Linux, Mac OSX, ..)
- ... e sotto forma di diverse **distribuzioni**
 - ▶ Standard, per sviluppatori J2EE/C/C++/Python/Web, ...
- Architettura **modulare** (via plug-ins)

Un po' di storia...

- Nato nei laboratori di ricerca IBM (alphaWorks) nel 2001
- Successivamente donato alla comunità open-source che ora ne cura lo sviluppo per mezzo di un'apposita fondazione (dal 2004)
- Correntemente supportato/utilizzato da un vasto numero di sviluppatori, sia in ambito accademico che industriale

- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



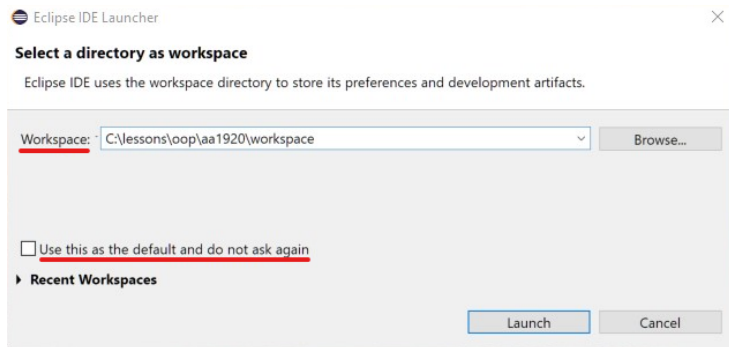
Astrazioni di base in Eclipse (1/5)

Workspace

- Directory in cui Eclipse mantiene una collezione di **risorse** (file, cartelle, progetti)
- Include le “*preferenze di una specifica sessione di lavoro*”
 - ▶ Le configurazioni dell’ambiente
 - ▶ L’elenco dei progetti coinvolti
 - ▶ View e Perspective aperte/disponibili
 - ▶ Stato dei plugin installati
 - ▶ ...
- I **metadati** di un workspace si trovano nella directory *.metadata*
- Tipicamente, un workspace contiene anche i **progetti**, ognuno in una cartella `<workspaceDir>/<projectDir>`
- In alternativa, i progetti possono essere “importati” pur risiedendo in altre posizioni del file system (*linked resources*)

Gestione del Workspace

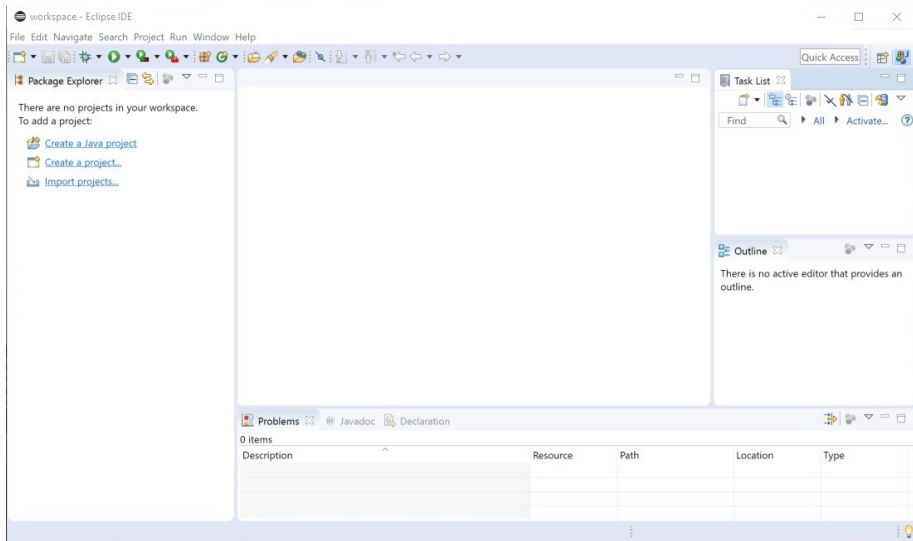
- Il workspace d'interesse deve essere creato/scelto all'avvio di Eclipse



- Può essere cambiato anche dopo
 - ▶ File > Switch Workspace > ...



Primo avvio



Astrazioni di base in Eclipse (2/5)

View

- Componente dell'interfaccia di Eclipse
- Tipicamente numerose view sono aperte contemporaneamente;
- Ciascuna offre una micro-funzionalità, ad esempio:
 - ▶ *Package Explorer* — fornisce una vista dei progetti e della loro struttura;
 - ▶ *Console* — consente di usare un terminale interno all'IDE invece del terminale di sistema;
 - ▶ *Outline* — mostra un riassunto dei componenti della classe attualmente aperta, elencando i membri, consentendo di filtrarli (ad esempio, nascondendo i privati) e di ordinarli (ad esempio in ordine alfabetico);
 - ▶ *Problems* — elenca gli eventuali problemi che affliggono il progetto (errori di configurazione, sorgenti che non compilano, warnings...)



Astrazioni di base in Eclipse (3/5)

Perspective

- Insieme di view ed editor opportunamente organizzati
 - ▶ Cambiare perspective consente di cambiare rapidamente le view attive
 - ▶ Tipicamente, si usano perspective diverse per fasi diverse dello sviluppo
- Ne vedremo sicuramente due:
 - ▶ Java — per lo sviluppo di applicazioni Java
 - ▶ Debug — per il debug di applicazioni (prossima lezione!)



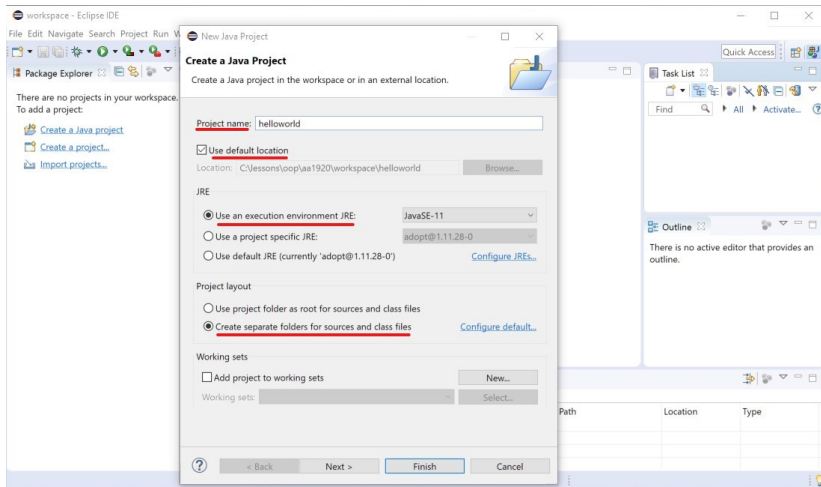
Astrazioni di base in Eclipse (4/5)

Progetto

- Directory contenente una collezione di **risorse** opportunamente organizzate, tipicamente rappresentanti un software o una parte di un software
 - ▶ codice sorgente
 - ▶ file di configurazione
 - ▶ risorse (immagini, video, ...)
 - ▶ file compilati (bytecode, in java)
 - ▶ librerie esterne
 - ▶ ...
- Ciascun progetto fa generalmente riferimento ad uno specifico linguaggio di programmazione
- Ogni progetto, a seconda della propria **tipologia** (*nature*), può utilizzare diversi strumenti per la gestione delle proprie risorse

Creazione di un Progetto Java

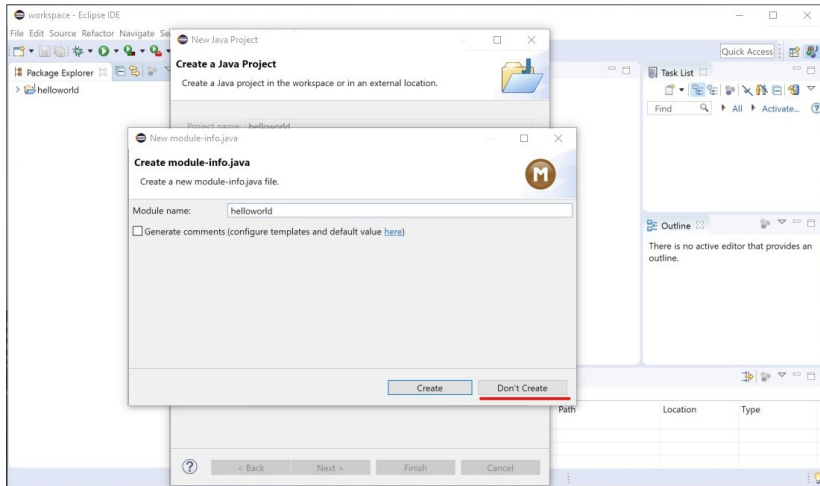
- File > New > Java Project



- → Finish

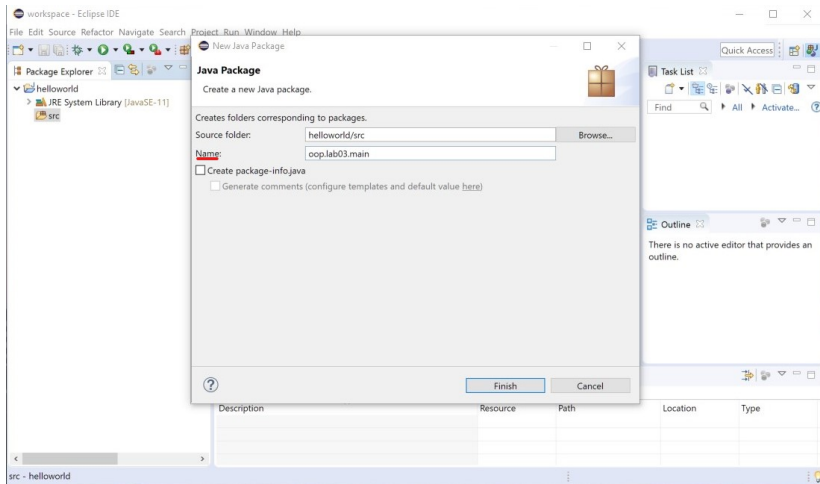
Creazione di un Progetto Java (nota sui moduli)

- Nell'ambito del corso, il concetto di *Modulo* non è utilizzato
 - ▶ Per ogni nuovo progetto è quindi necessario evitare la creazione...



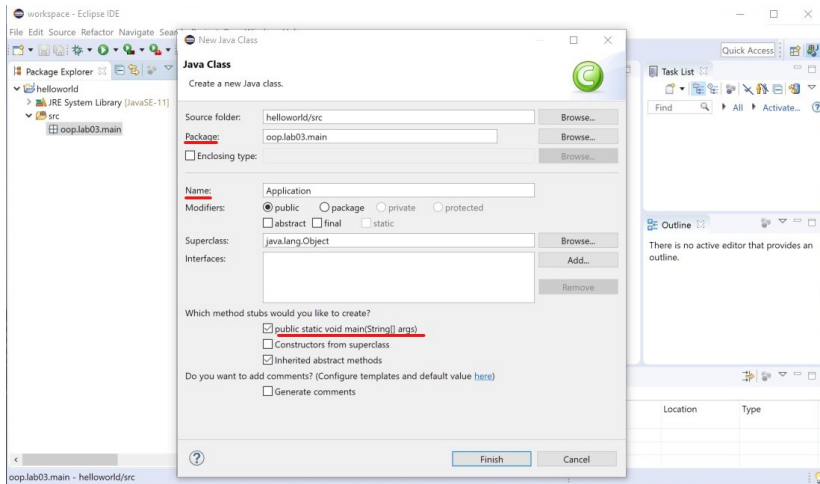
Creazione di Package e Classi

- File > New > Package

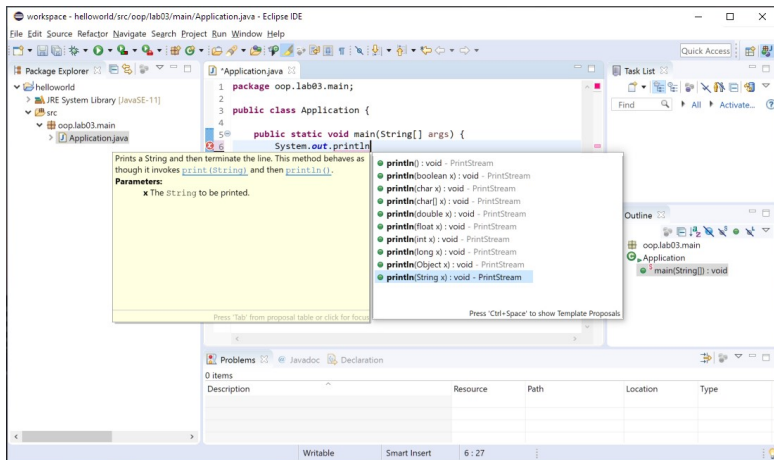


Creazione di Package e Classi

- (click dx su package in Package Explorer View) > New > Class



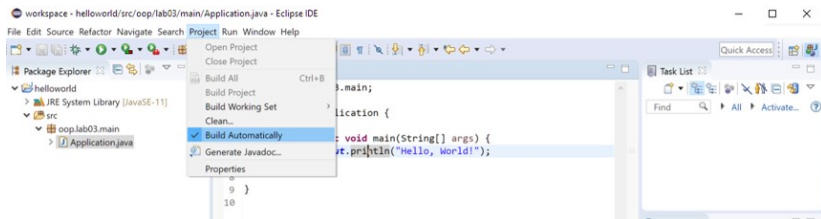
JDT: Code Completion e Syntax Highlighting



- CTRL+Space : richiama la code completion su una porzione di codice



Compilazione dei Sorgenti



Due diverse possibilità

- **Compilazione automatica**

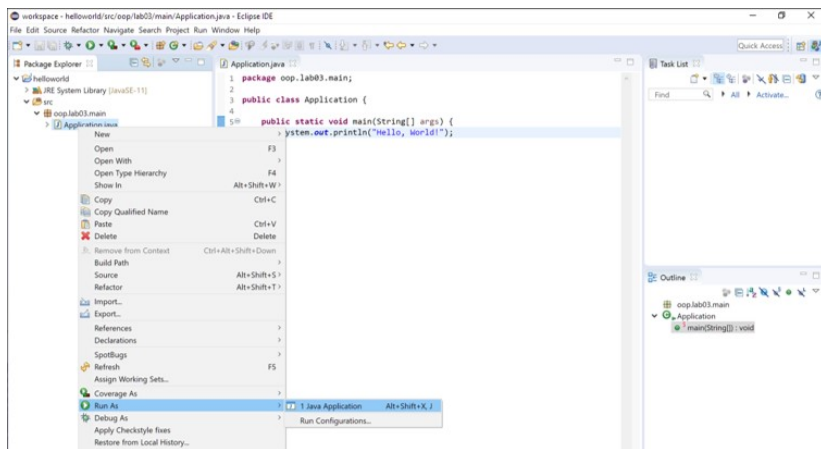
- ▶ Il compilatore viene invocato automaticamente dall'IDE ad ogni modifica (salvata) ai sorgenti del progetto
- ▶ E' la modalità che generalmente è da preferire

- **Compilazione manuale**

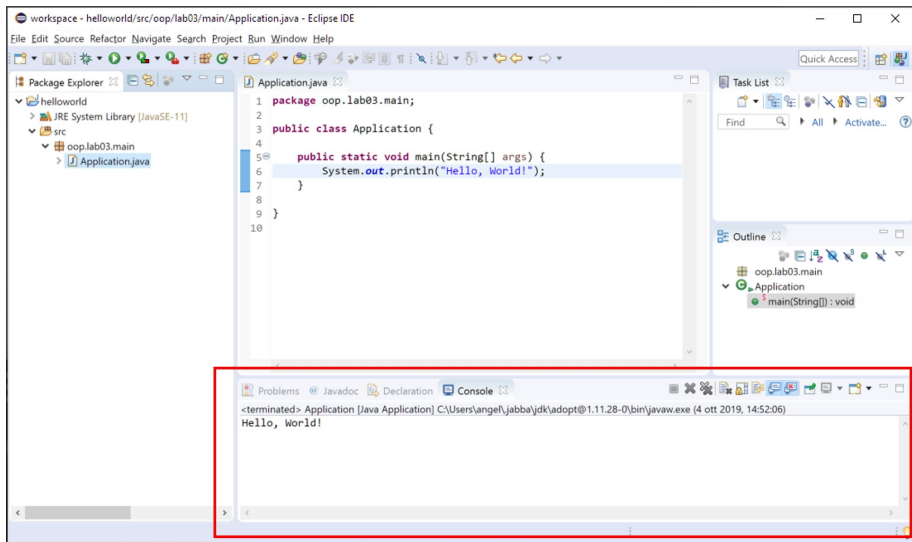
- ▶ Attiva solo quando è disabilitata la compilazione automatica
- ▶ Il compilatore viene invocato a seguito di un esplicito click dello sviluppatore su *"Build All"* (CTRL+B)

Esecuzione di Applicazioni

- (click dx su classe principale) > Run As > Java Application

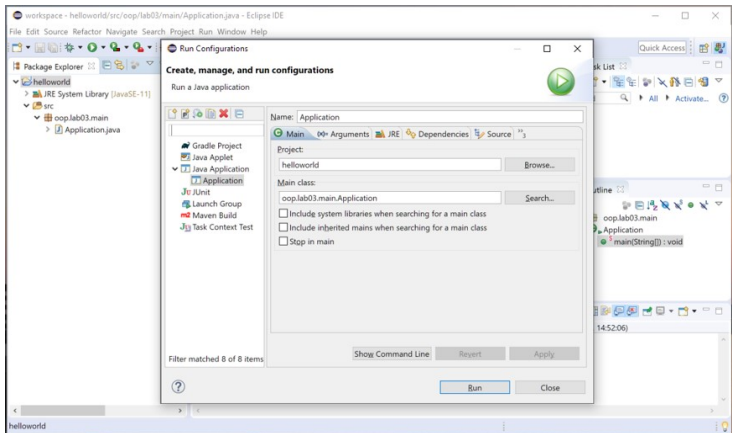


Esecuzione di Applicazioni – Output View



Run Configuration

- Una **run configuration** consente di personalizzare la configurazione con la quale l'applicazione va in esecuzione
 - ▶ Ad esempio, i parametri da passare in ingresso al metodo main
- Run > Run Configurations



Run Configuration: Program Arguments

The screenshot displays the Eclipse IDE interface. The main editor shows the source code of `Application.java` in the package `oop.lab03.main`. The code defines a `main` method that prints "Hello, World!" and iterates over command-line arguments.

```
1 package oop.lab03.main;
2
3 public class Application {
4
5     public static void main(String[] args) {
6         System.out.println("Hello, World!");
7
8         for(String s : args) {
9             System.out.println(s);
10        }
11    }
12 }
13
14
```

The Package Explorer on the left shows the project structure: `helloworld` > `src` > `oop.lab03.main` > `Application.java`.

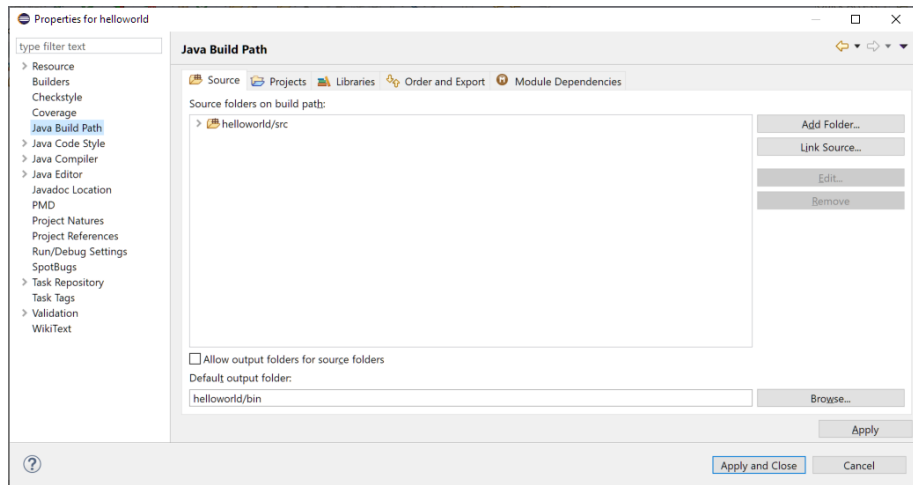
The Run Configurations dialog is open, showing the configuration for the `Application` class. The `Main` tab is selected, and the `Program arguments` field contains `value01 value02 value03`. The `VM arguments` field is empty.

The Output console at the bottom left shows the execution results:

```
<terminated> Application
Hello, World!
value01
value02
value03
```

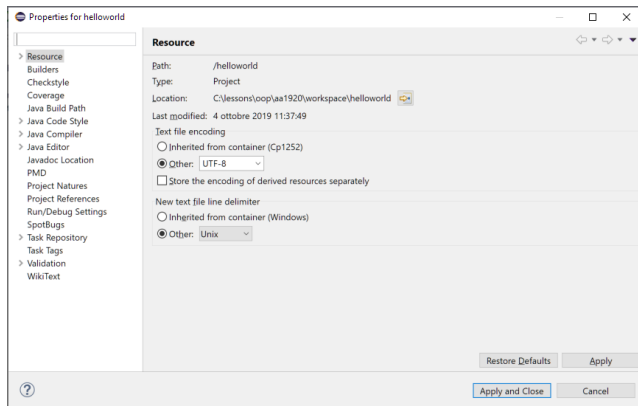
Java Build Path

- Il **java build path** specifica le risorse (ad es. sorgenti, altri progetti, cartelle di compilati, file JAR) visibili durante la compilazione (build)
- (click dx su progetto) > Build Path > Configure Build Path



Java Project Resources

- (click dx su progetto) > Properties > Resource



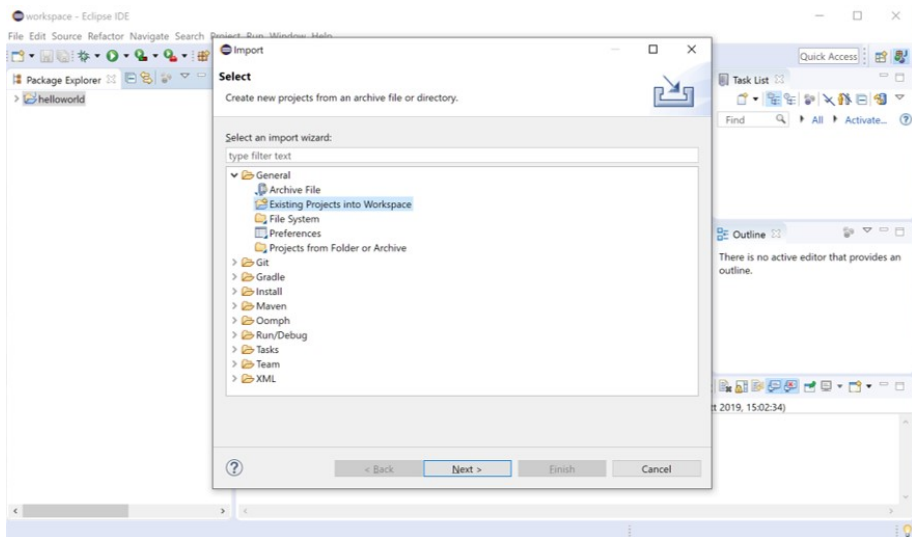
Importante:

- Text File Encoding: UTF-8
- New text file line Delimiter: Unix



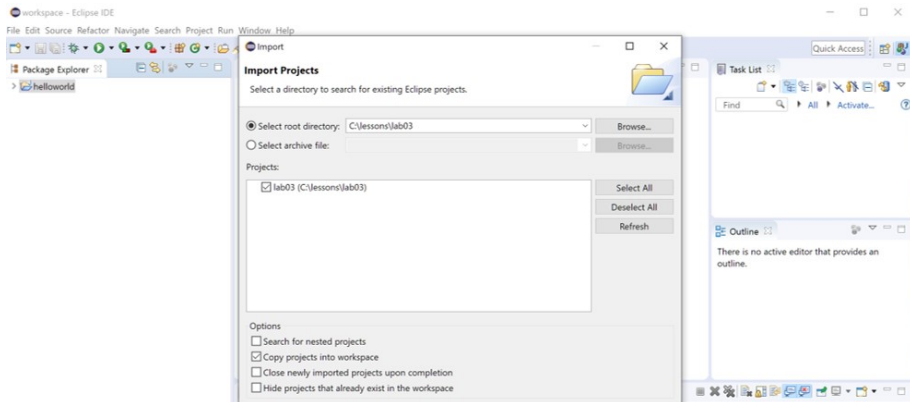
Importazione di Progetti Esistenti (1/2)

- File > Import... > General > Existing Project Into Workspace



Importazione di Progetti Esistenti (2/2)

- Selezionare la directory del progetto
 - ▶ Generalmente, una directory di un progetto Java importabile in Eclipse contiene almeno i file `.project`, `.classpath` e la directory `.settings`, oltre ai sorgenti del progetto.
- Decidere se copiare il progetto nel workspace oppure se continuare a modificare il progetto nella sua posizione originale sul filesystem



Astrazioni di base in Eclipse (5/5)

Plug-in

Un **plug-in** è un *componente software*, installabile opzionalmente, che estende le capacità dell'IDE. Ad esempio:

- controllare la qualità del codice Java;
- aggiungere supporto ad ulteriori linguaggi;
- usare sistemi di build diversi da quello predefinito (Gradle, Maven...);
- Ne vedremo diversi durante il corso...



- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



Refactoring

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.
— refactoring.com

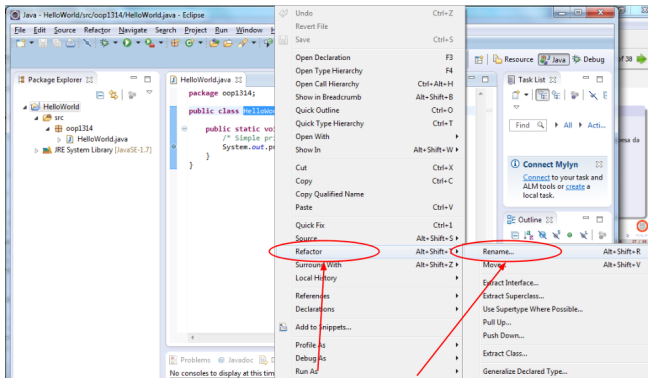
Vantaggi nel refactoring supportato dall'IDE

- Le modifiche sono gestite dall'IDE in maniera consistente
 - ▶ Es (1). *Rinominare una variabile*: se si procede “a mano”, si dovrà modificare il nome della variabile sia nella riga in cui la si dichiara sia in tutte le sue occorrenze di utilizzo. Viceversa, avvalendosi dell'IDE, la modifica da fare è una sola.
 - ▶ Es (2). *Spostare una classe da un package ad un altro*: utilizzando l'IDE vengono aggiornati tutti i riferimenti nell'intero progetto
- Minimizza l'introduzione di errori in fase di refactoring



Refactoring in Eclipse

- Attivabile con *tasto destro* → Refactor o tramite ALT+SHIFT+T sulla selezione di interesse
- Es.: modifica nome di variabili, metodi, classi etc. (ALT+SHIFT+R)
 - ▶ Gestite in maniera automatica e safe dall'IDE

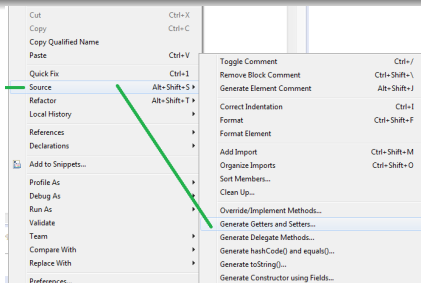


Esempio di renaming di una classe



Generazione di codice in Eclipse

- Attivabile con *tasto destro* → Source o tramite ALT+SHIFT+S
- Esempi
 - ▶ aggiunta/rimozione di commenti
 - ▶ formattazione di codice (es. indentazione)
 - ▶ Generazione di metodi
 - costruttori
 - getter e setter
 - toString()
 - hashCode() e equals()



- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



Keyboard Shortcuts

- **CTRL+1**: quick-fix contestuale per errori (molto potente)
- **ALT+SHIFT+R**: refactoring di campi/metodi/classi
- **CTRL+SHIFT+ /**: commento delle linee selezionate
- **CTRL+PGUP/PGDOWN**: per muoversi di uno step avanti (PGUP) o indietro (PGDOWN) tra la lista di sorgenti correntemente aperti
- **F3** (o **CTRL+CLICK**): ci sposta alla definizione di un dato elemento
- **CTRL+SHIFT+L**: lista delle shortcut disponibili per il dato contesto
- **CTRL+SHIFT+O**: include in automatico tutti gli import necessari sulla base delle classi utilizzate nel sorgente corrente
- **CTRL + .**: sposta il cursore al successivo errore/warning
- **CTRL+F8**: consente di spostarsi tra le varie perspective
- **CTRL+J**: search incrementale, senza l'uso di GUI
- **ALT+SHIFT+S**: da l'accesso a un insieme di wizard con cui automatizzare la scrittura di costruttori, getter, setter, etc.



- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



Motivazioni

- Difficilmente le applicazioni software risultano essere totalmente esenti da problemi/errori alla prima stesura del codice sorgente
- Spesso lo sviluppatore tende a sottovalutare e/o ignora alcuni effetti collaterali (*side-effects*) che porzioni di codice sorgente possono provocare
- In genere, un software privo di errori lo si ottiene step-by-step
 - ▶ Una buona progettazione a priori consente di ridurre il numero di bug che si potrebbero inserire nel codice sorgente, tuttavia...
 - ▶ ... qualche bug sarà inevitabilmente presente, e dovrà essere identificato e corretto!



Debugging di Applicazioni [1, 2] II

Approcci

1. Meccanismi di logging

- ▶ Si aggiunge al codice sorgente la possibilità di fare logging in modo da tracciare lo stato interno del programma in fase di esecuzione: valore corrente di variabili, campi, parametri, ...
- ▶ Utile in alcuni casi, ma in generale **da evitare!**
- ▶ Appesantisce il sorgente
- ▶ Appesantisce il software (a meno di non usare opportune librerie)
- ▶ Basta un side effect per generare un Heisenbug [3]

2. Utilizzo di strumenti di debugging forniti dall'IDE

- ▶ Controllo efficace sull'esecuzione dell'applicazione
- ▶ Ispezionabilità a run-time
- ▶ Potenti strumenti per indagare cosa succede nel programma
- ▶ Molto difficile che causi Heisenbugs
 - Finché non si mette in mezzo la concorrenza...



Debug: concetti principali I

Breakpoint

Speciale punto di controllo che, se raggiunto dal flusso di controllo, sospende l'esecuzione

Espressioni e osservabilità

La capacità valutare espressioni che includono variabili attualmente accessibili dal flusso di controllo, osservandone il valore corrente

Modifica dei valori a tempo d'esecuzione

La capacità modificare i valori di variabili e campi manualmente a programma in esecuzione, una volta che il flusso di controllo è sospeso



Debug: concetti principali II

Step-by-step execution

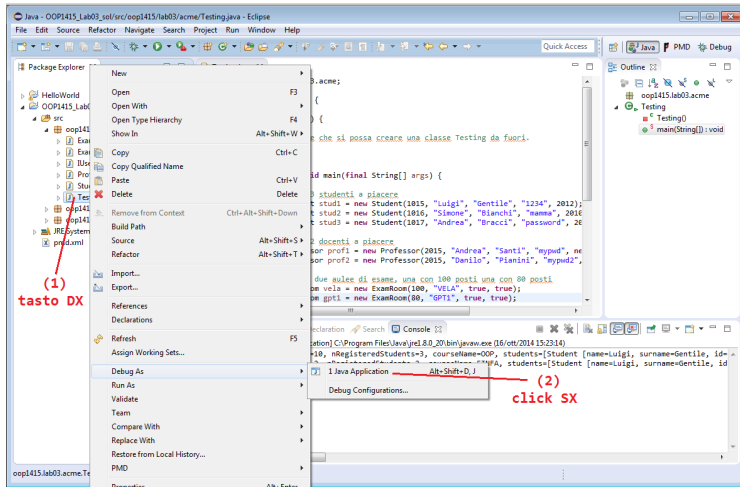
La capacità, a flusso di lavoro sospeso (e, quindi, una volta incontrato un breakpoint) di proseguirla andando avanti di un'istruzione alla volta

Step-into execution

La capacità, a flusso di lavoro sospeso (e, quindi, una volta incontrato un breakpoint) di proseguirla “entrando” nel record di attivazione di un metodo che viene invocato

Debugging di Applicazioni in Eclipse

- L'avvio del debugging è molto simile all'esecuzione delle applicazioni
 - ▶ (menu contestuale di progetto) > Debug As > Java Application



Debugging in Eclipse: manipolazione del flusso di controllo

- L'avvio del debugger in Eclipse comporta l'apertura della perspective *Debug*, con gli strumenti utili per le operazioni di debugging

Principali operazioni di manipolazione del flusso di controllo

1. Inserimento di un breakpoint
 - ▶ Doppio click sulla linea di interesse (o CTRL+SHIFT+B)
2. Esecuzione **step-by-step**: F6
 - ▶ Una volta che l'esecuzione è stata sospesa da un breakpoint
3. **Step into**: F5
 - ▶ Debugging corpo di un metodo/costruttore
4. **Step return**: F7
 - ▶ Ritorno dal debugging del corpo di un metodo/costruttore
5. **Resume**: F8
 - ▶ Esecuzione fino al prossimo breakpoint

Debugging in Eclipse: strumenti I

Breakpoint condizionali

Tramite click destro su breakpoint e “Properties”, è possibile scrivere una condizione per il breakpoint, ad esempio di scattare solo dopo un certo numero di volte, oppure di scattare se una certa condizione è vera (estremamente utile).

Osservazione e manipolazione dei valori

La view “Variables” consente di visualizzare il valore di tutti i campi e delle variabili locali, esplorando anche l'interno degli oggetti. Inoltre, consente di modificarne i valori a runtime, per ispezionare il funzionamento del programma.



Debugging in Eclipse: strumenti II

Valutazione di espressioni

È possibile, nella tab “Expressions”, scrivere un’espressione java valida nel contesto (quindi, con accesso alle variabili locali e ai campi). L’IDE la valuterà e scriverà il risultato. Molto utile per visualizzare valori che richiederebbero altrimenti l’inserimento di una stampa ed il riavvio del debug, oppure un uso complicato dell’osservazione dentro Variables.



- 1 Integrated Development Environments
- 2 Eclipse IDE per Java
 - Astrazioni di Base
 - Refactoring e generazione di codice
 - Keyboard Shortcuts
- 3 Debugging di Applicazioni
- 4 Lab Startup



Preparazione ambiente di lavoro I

- Collegarsi al sito del corso
- Scaricare il file zip contenente gli esercizi del laboratorio
- Decomprime la directory sul Desktop
- Aprire Eclipse
- Eseguire la procedura di importazione del progetto esistente



Modalità di Lavoro

1. Leggere la consegna
2. Risolvere l'esercizio in autonomia
3. Cercare di risolvere autonomamente eventuali piccoli problemi che possono verificarsi durante lo svolgimento degli esercizi
4. Utilizzare le funzioni di test presenti nei sorgenti per il testing dell'esercizio
5. Contattare i docenti nel caso vi troviate a lungo bloccati nella risoluzione di uno specifico esercizio
6. A esercizio ultimato contattare i docenti per un rapido controllo della soluzione realizzata
7. Proseguire con l'esercizio seguente



Bibliography I

[1] Wikipedia.

Debugger.

Online, available at: <http://en.wikipedia.org/wiki/Debugger> – Last Retrieved: October 14, 2016.

[2] Wikipedia.

Debugging.

Online, available at: <http://en.wikipedia.org/wiki/Debugging> – Last Retrieved: October 14, 2016.

[3] Wikipedia.

Heisenbug.

Online, available at: <https://en.wikipedia.org/wiki/Heisenbug> – Last Retrieved: October 14, 2016.

