03
Eclipse IDE, Debug

Danilo Pianini
Giovanni Ciatto, Angelo Croatti, Mirko Viroli

Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

5 ottobre 2018

Outline

Outline

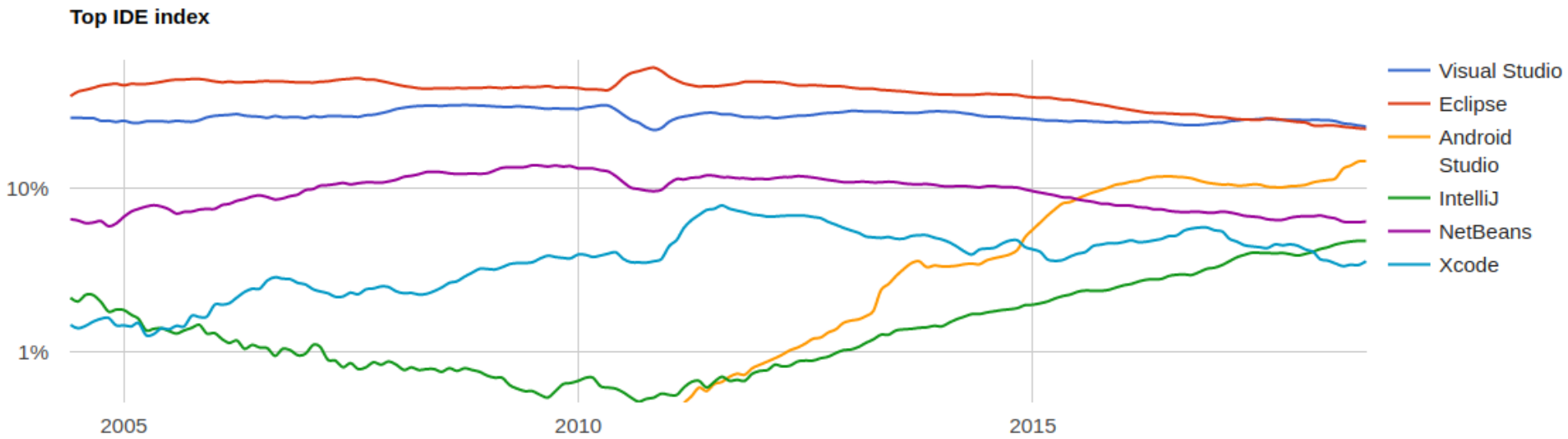
Outline

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 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`

Eclipse Overview

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..

Un po' di storia

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

Astrazioni di base I

Workspace

Directory dove Eclipse salva e carica le impostazioni di un certo numero di progetti. Tipicamente contiene anche i progetti (che possono però anche essere importati da altre directory)

Progetto

Directory contenente una collezione di risorse opportunamente organizzate, tipicamente rappresentanti un software o una parte di un software

Astrazioni di base II

Plug-in

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...);

Astrazioni di base III

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 IV

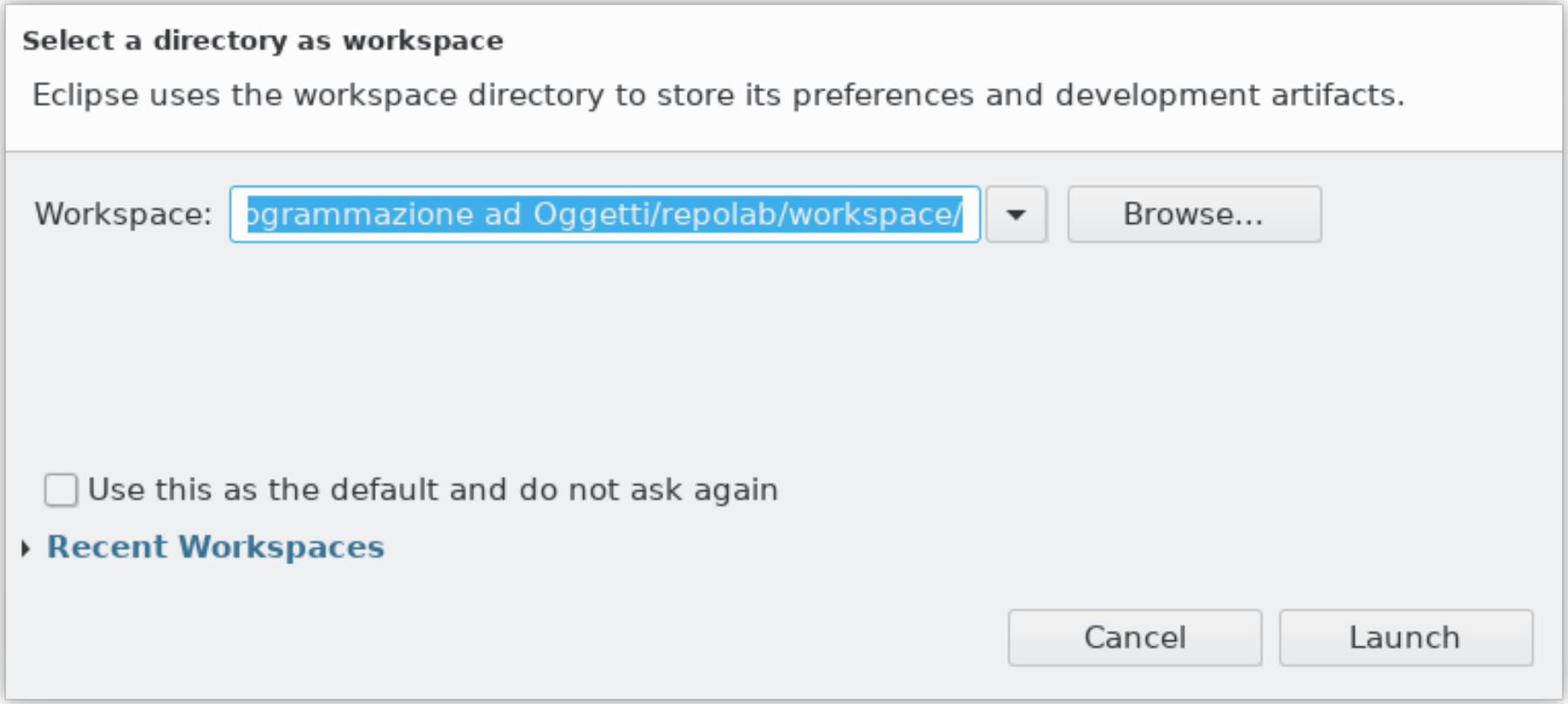
Perspective

Insieme di view che vengono. Cambiare perspective consente di cambiare rapidamente le view attive. Tipicamente, si usano perspective diverse per fasi diverse dello sviluppo. Ne vedremo sicuramente due:

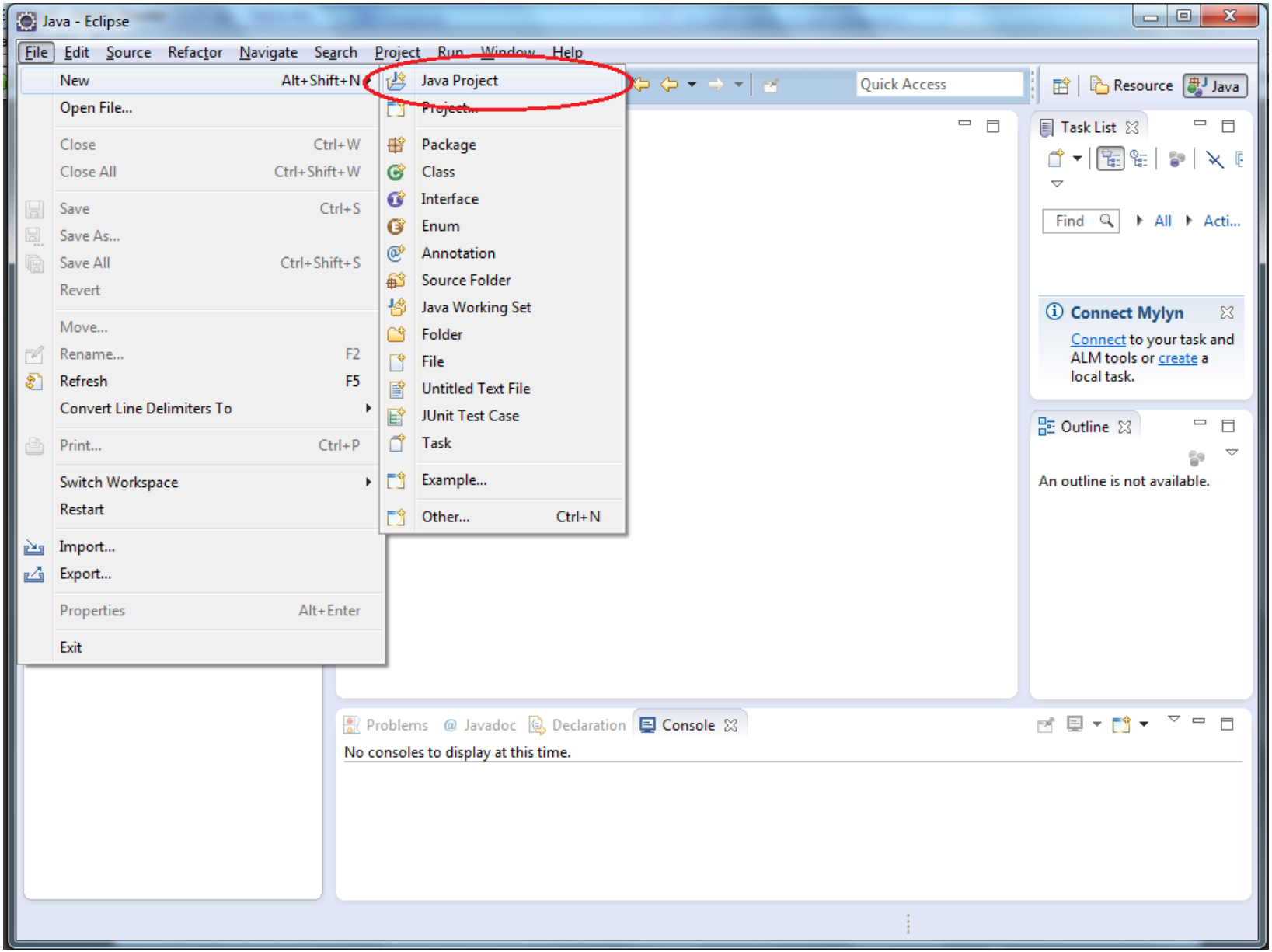
- Java — per lo sviluppo di applicazioni Java
- Debug — per il debug di applicazioni (prossima lezione!)

Outline

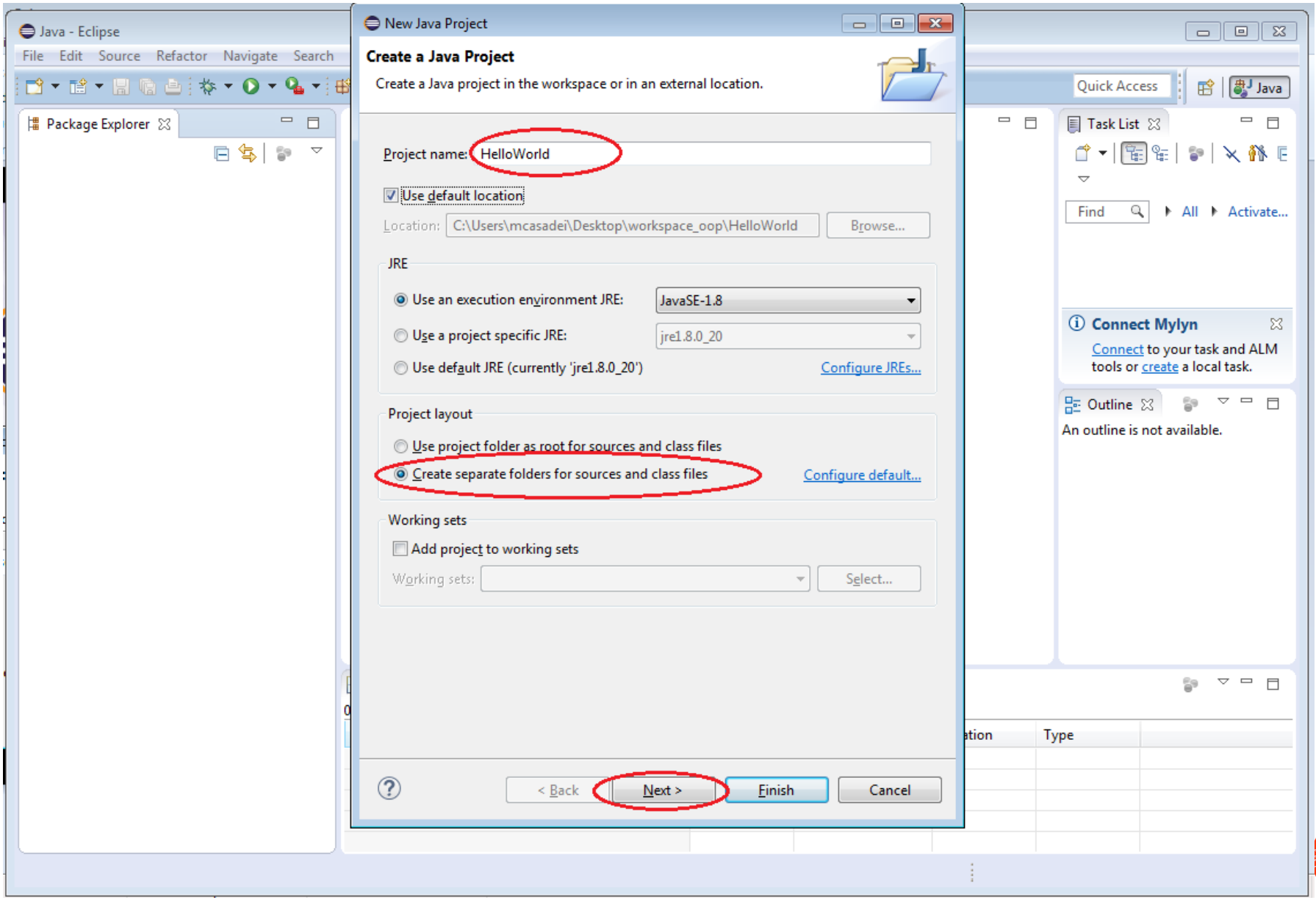
Avvio di Eclipse: Scelta del Workspace



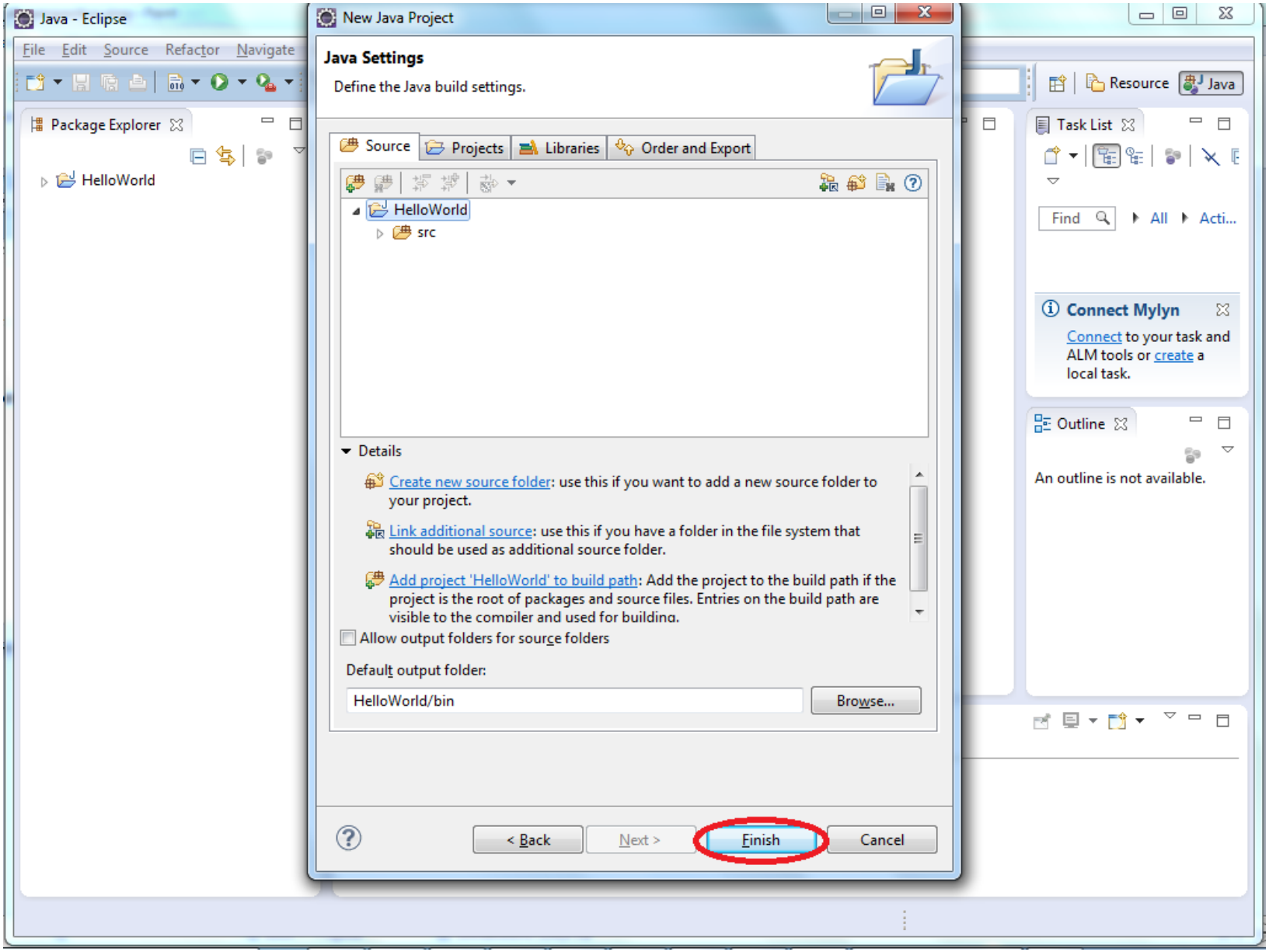
Creazione di Progetti Java 1/3



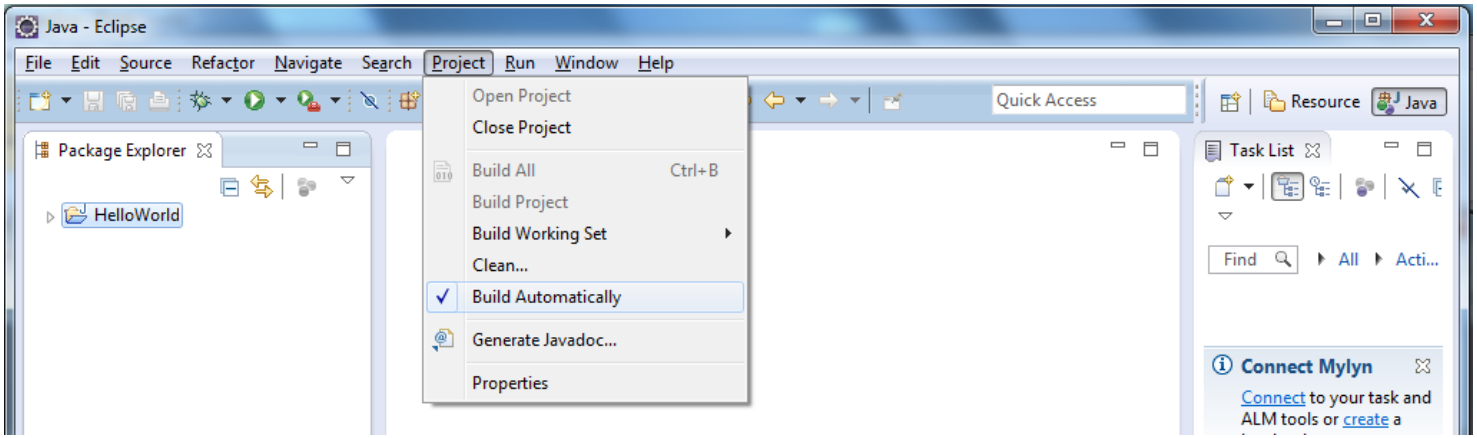
Creazione di Progetti Java 2/3



Creazione di Progetti Java 3/3



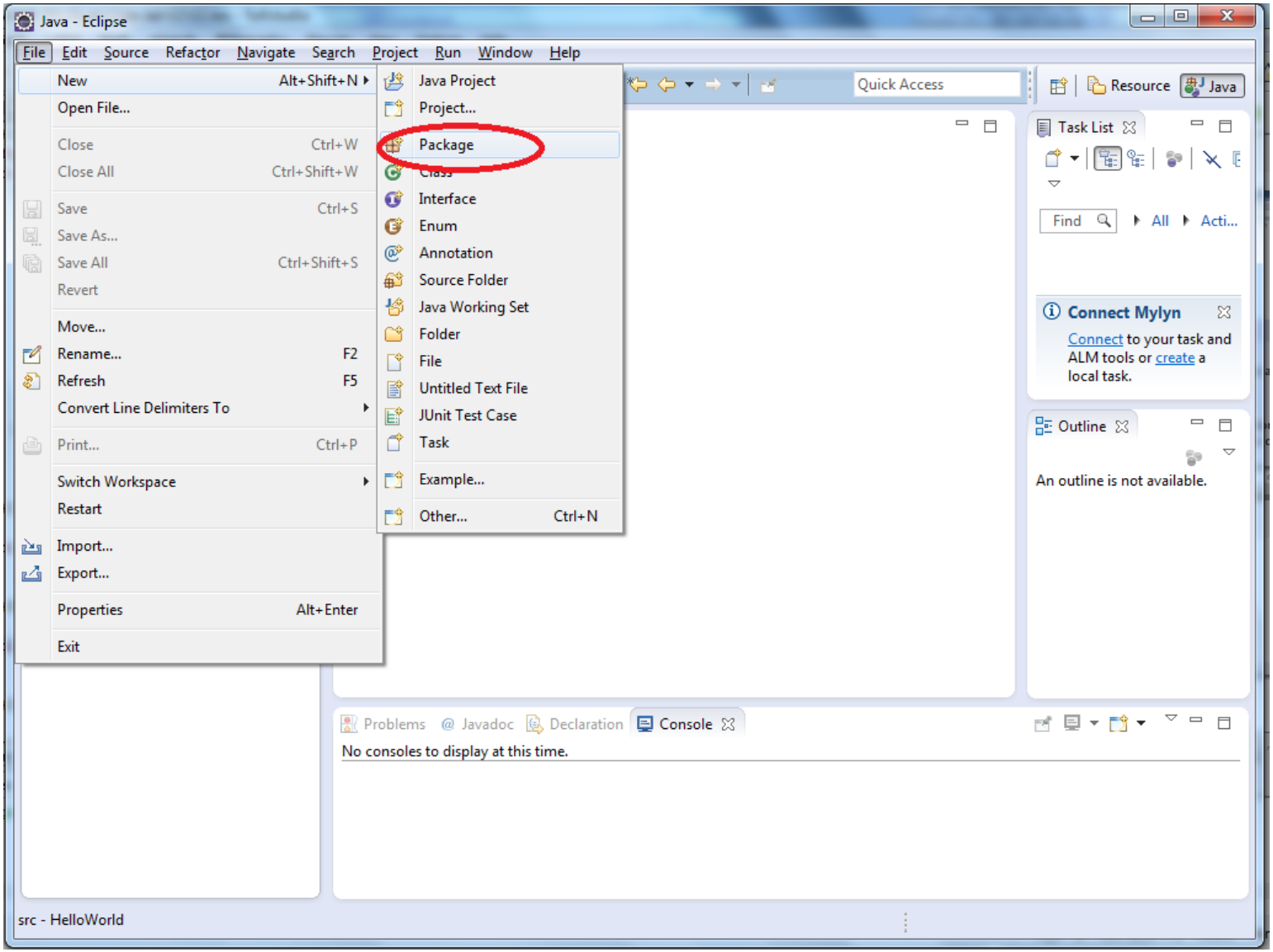
Compilazione dei Sorgenti



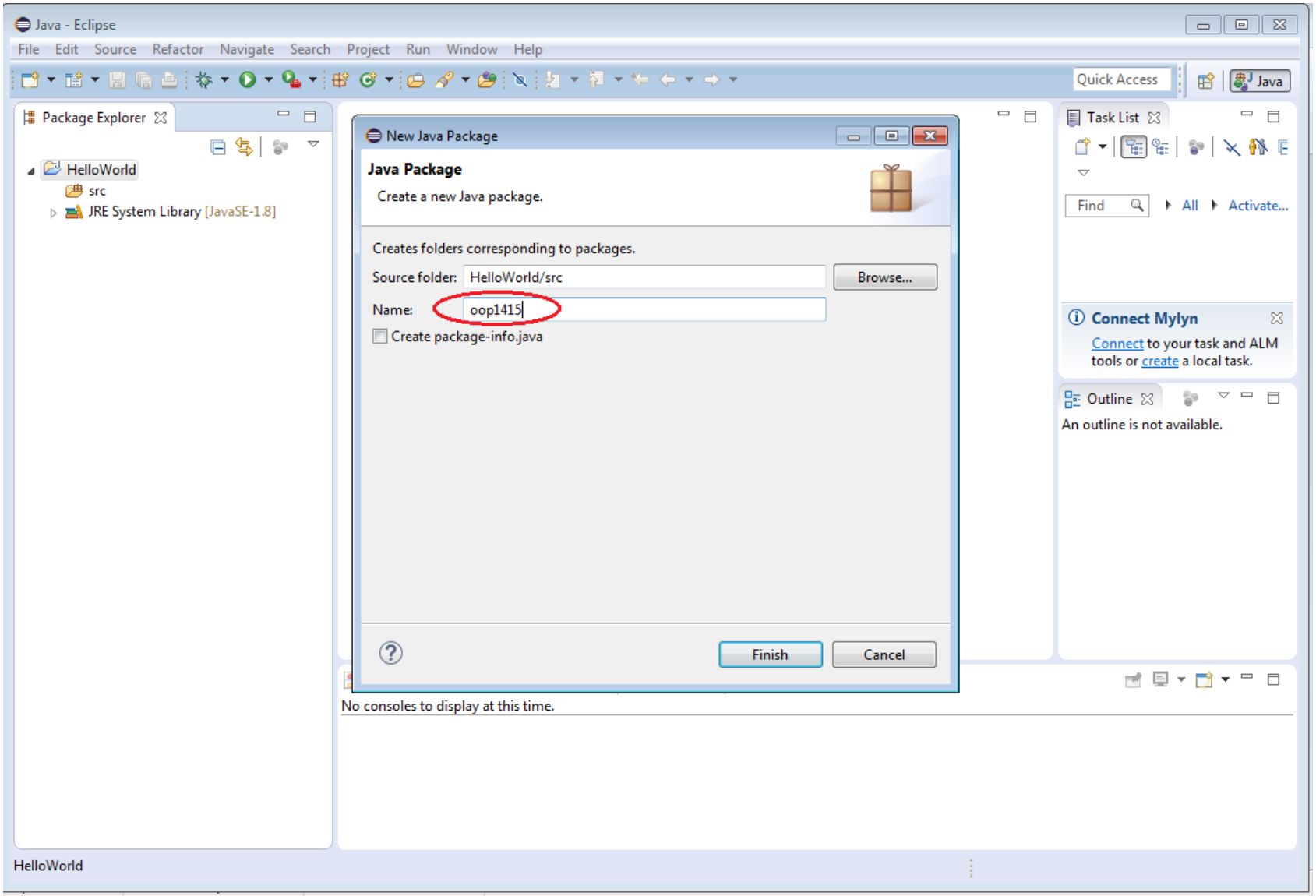
Due diverse possibilità

- Compilazione automatica
  - Il compilatore viene invocato automaticamente dall'IDE ad ogni modifica (salvata) ai sorgenti del progetto
- Compilazione manuale
  - Attiva quando è disabilitata la compilazione automatica
  - Il compilatore viene invocato a seguito di un esplicito click dello sviluppatore su *"Build All"*

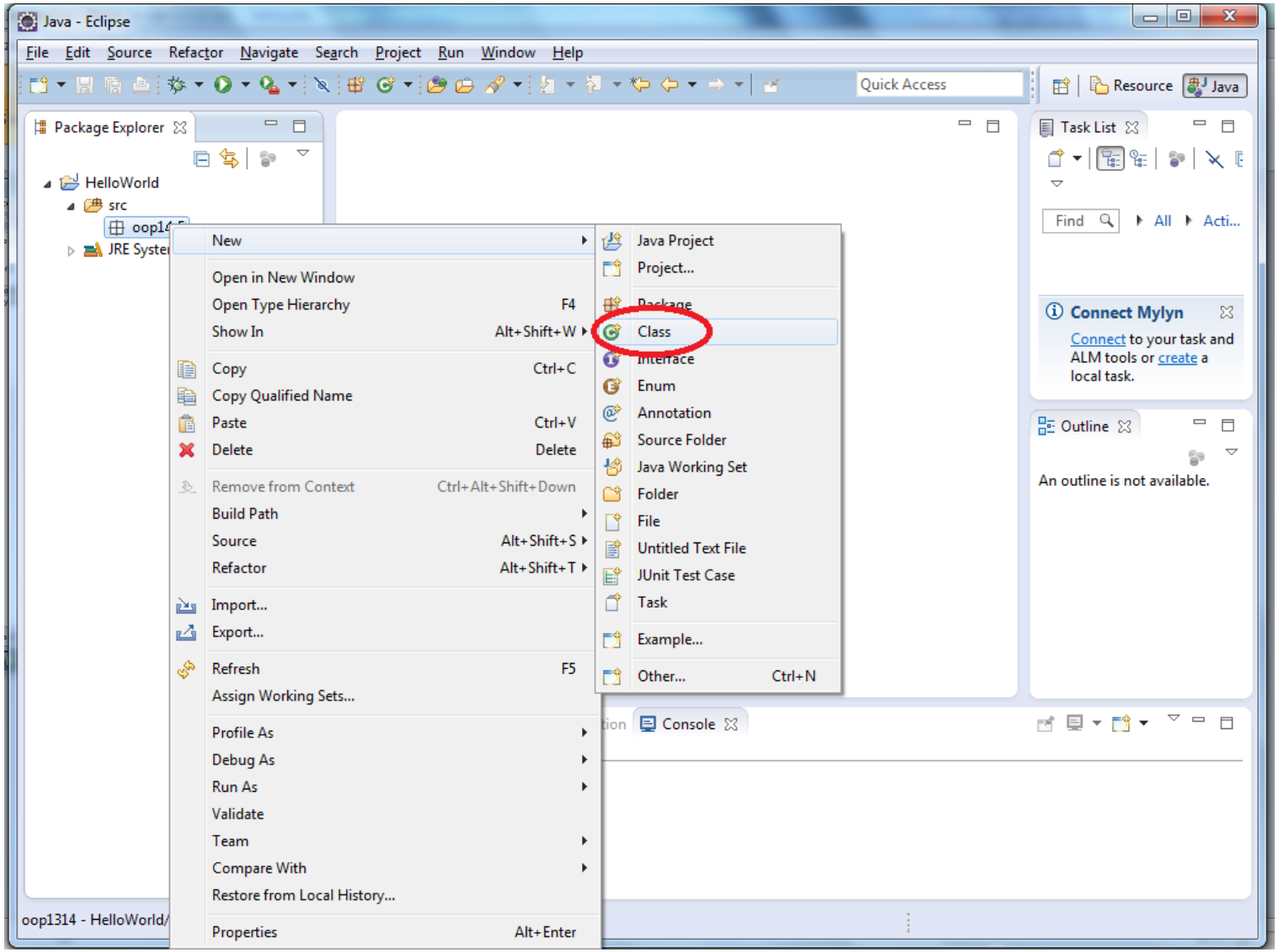
Creazione di un Package 1/2



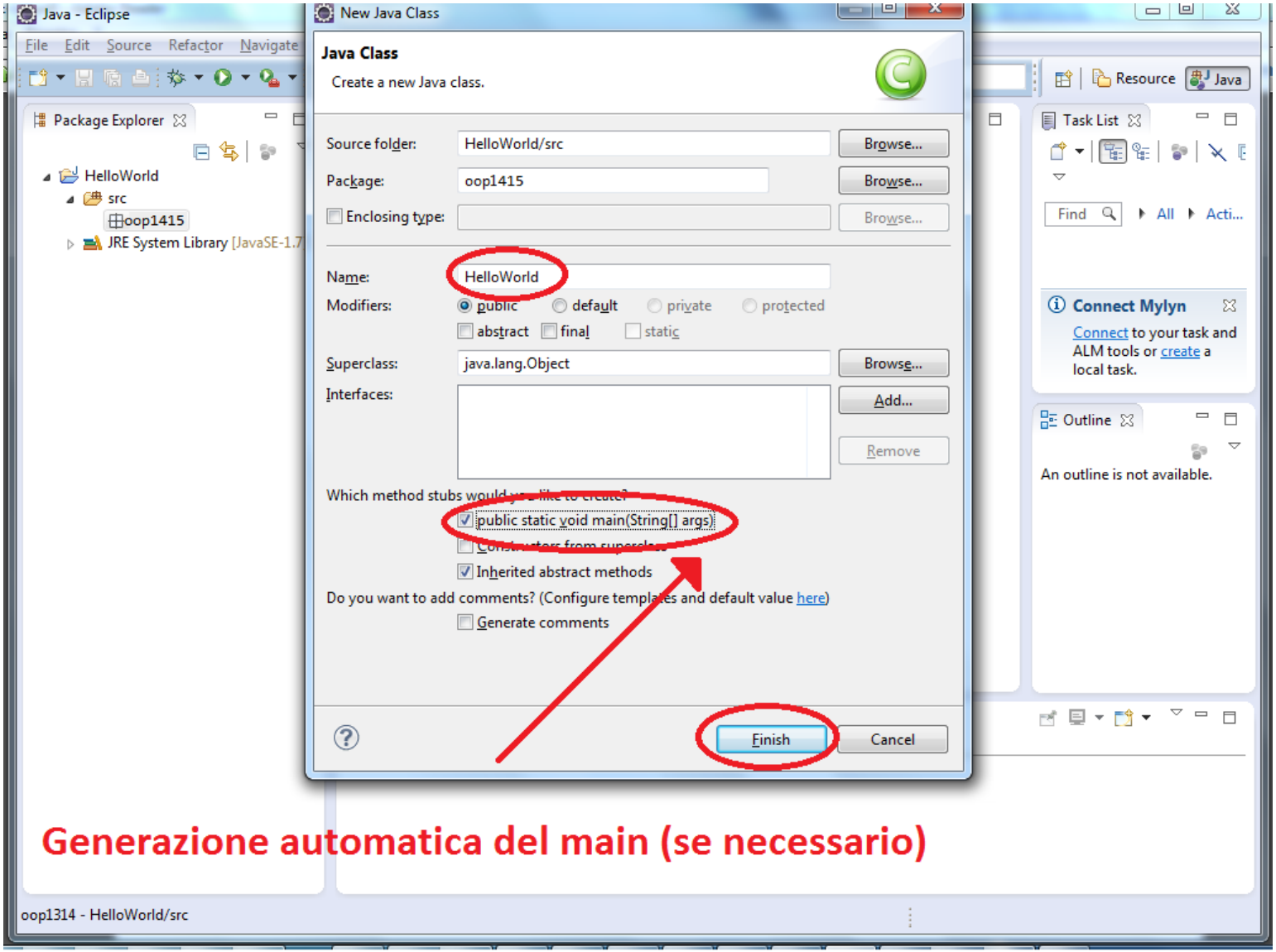
Creazione di un Package 2/2



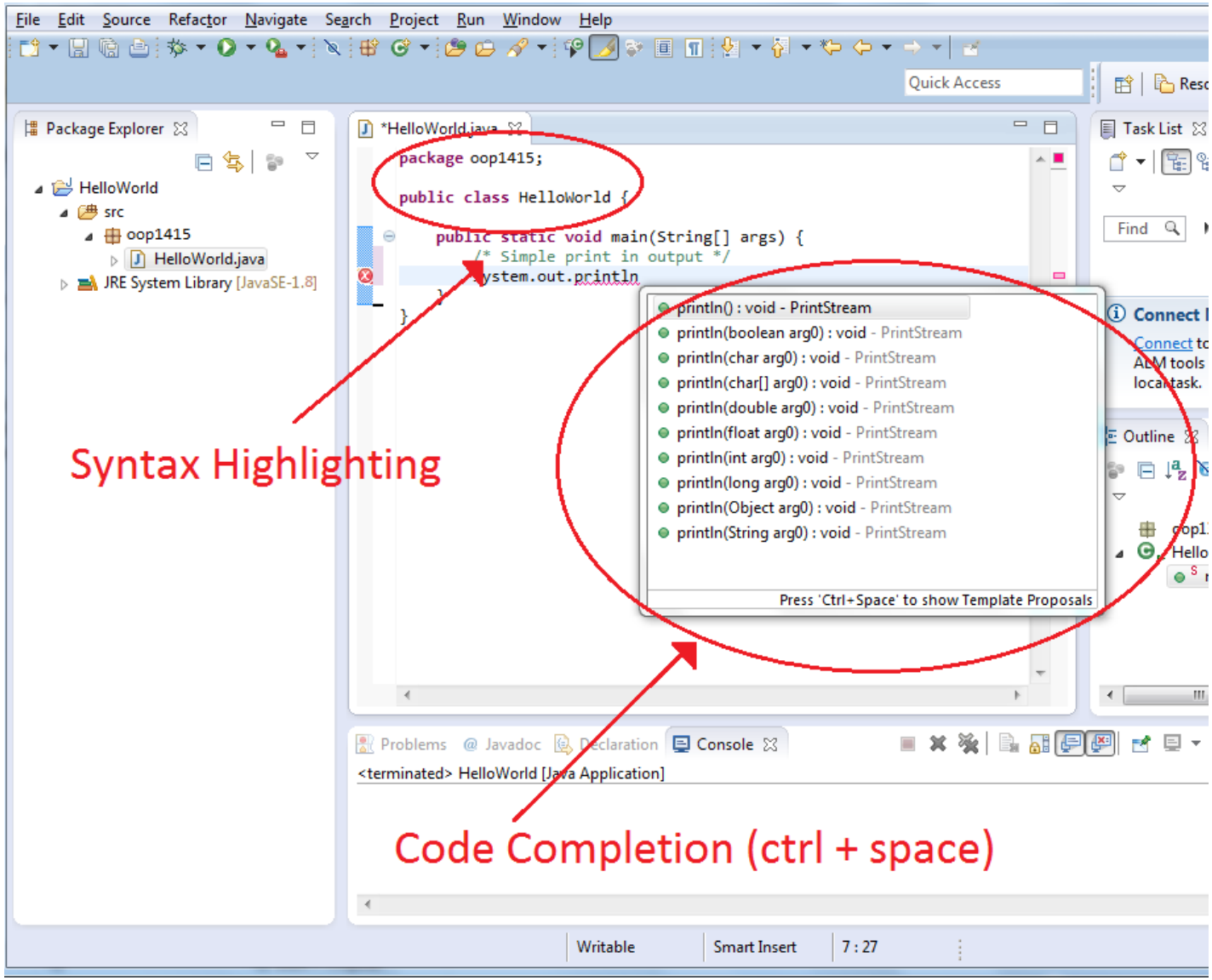
Creazione di una Classe 1/2



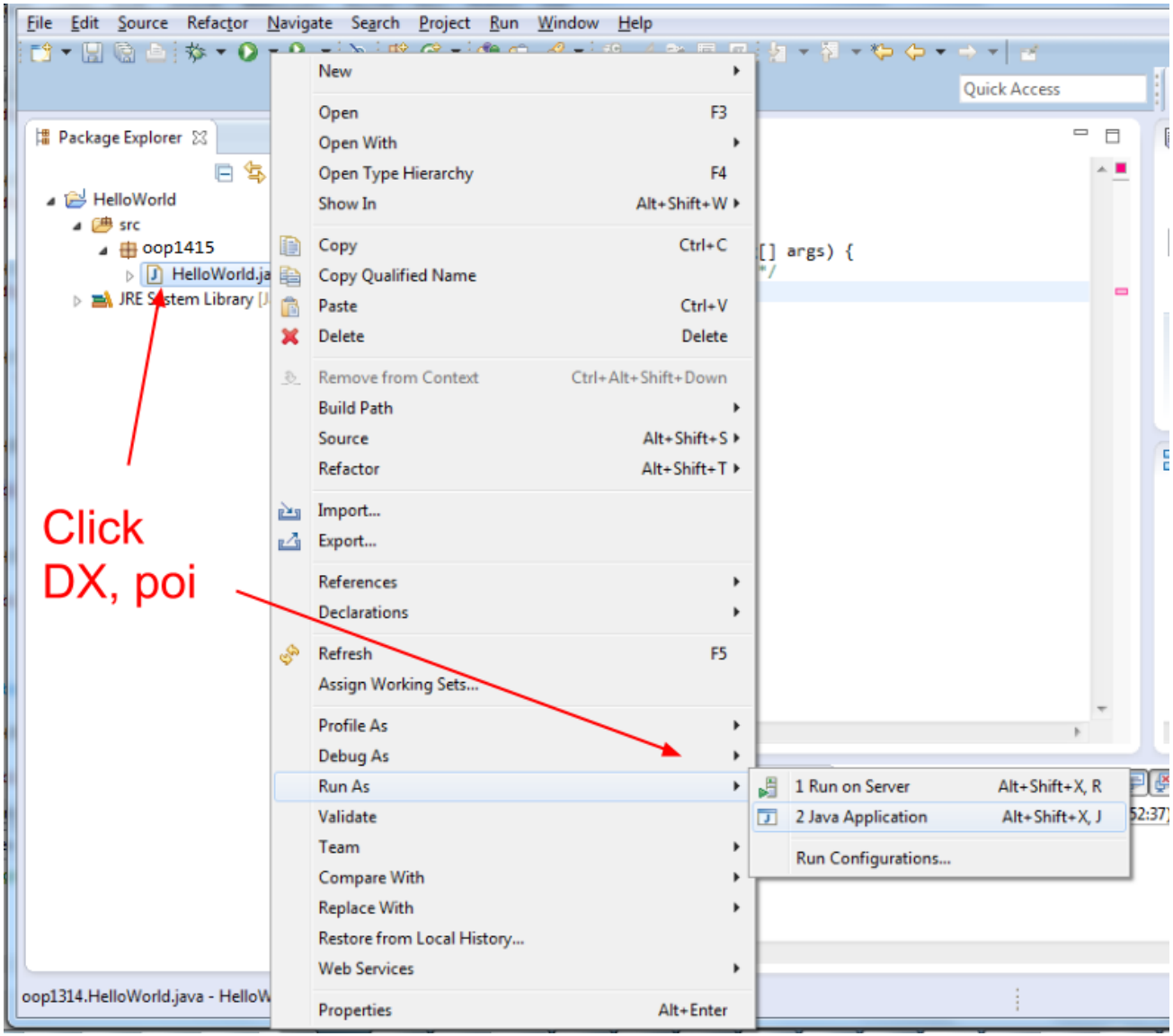
Creazione di una Classe 2/2



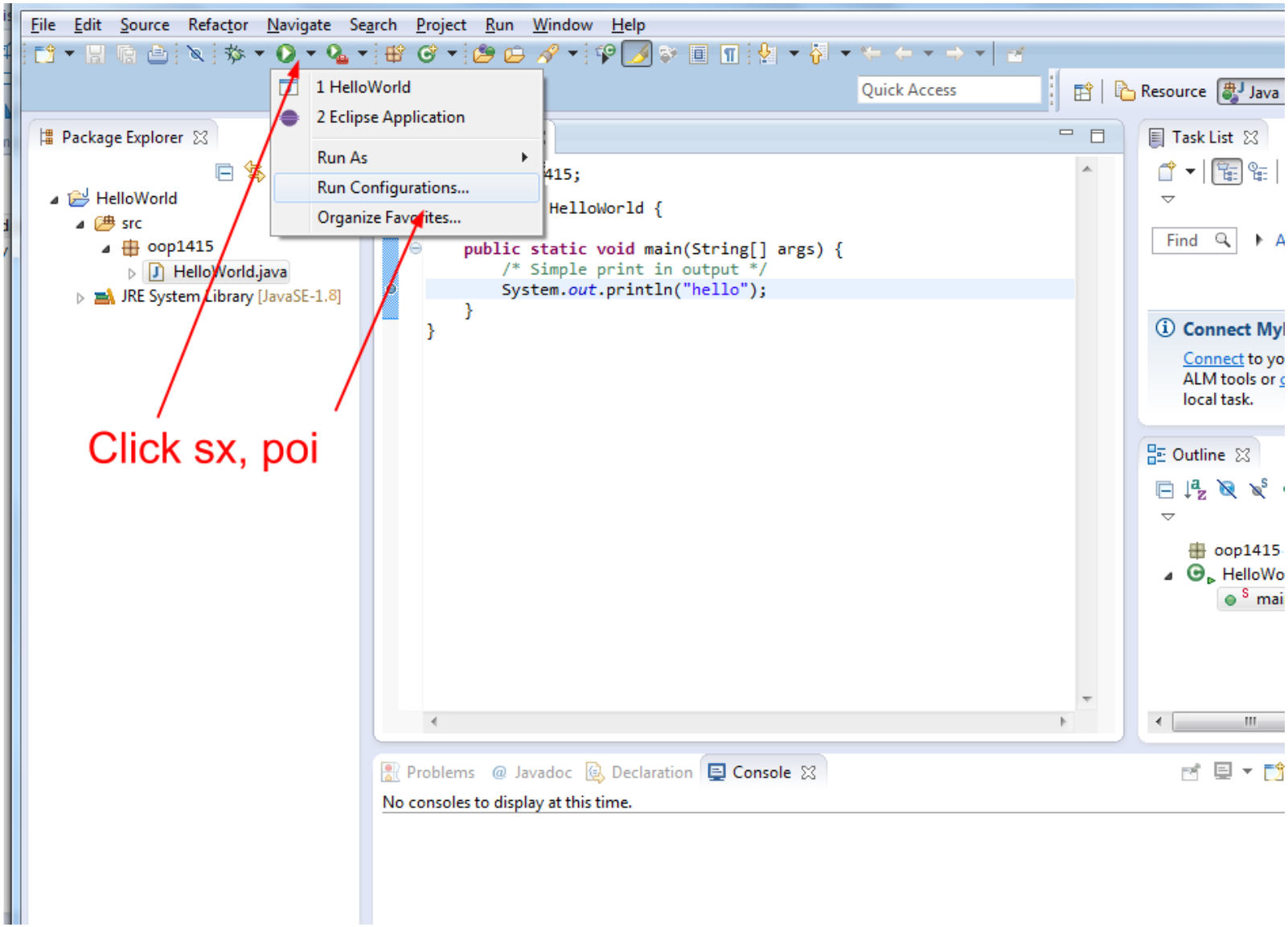
JDT: Code Completion e Syntax Highlighting



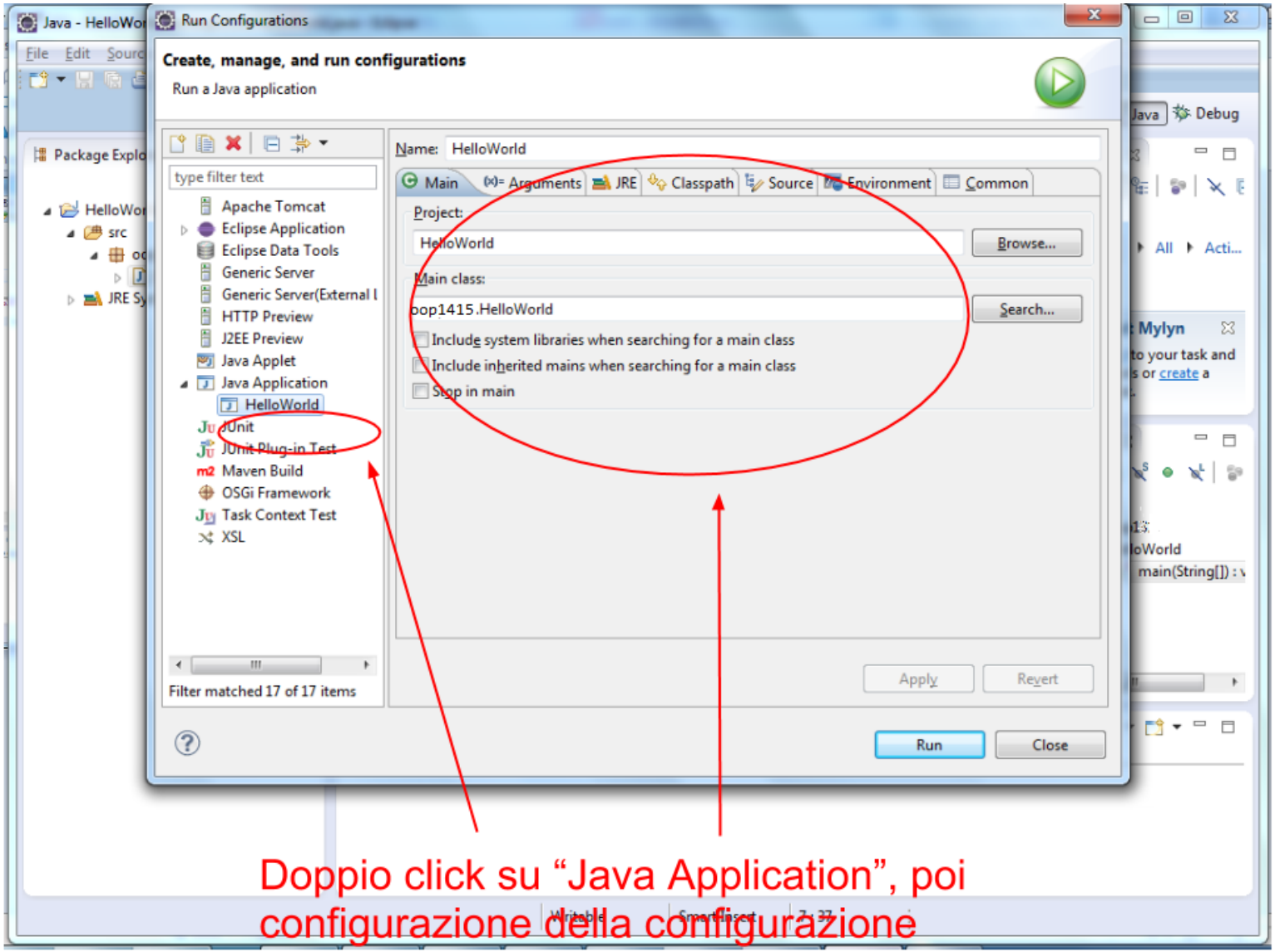
Esecuzione di Applicazioni



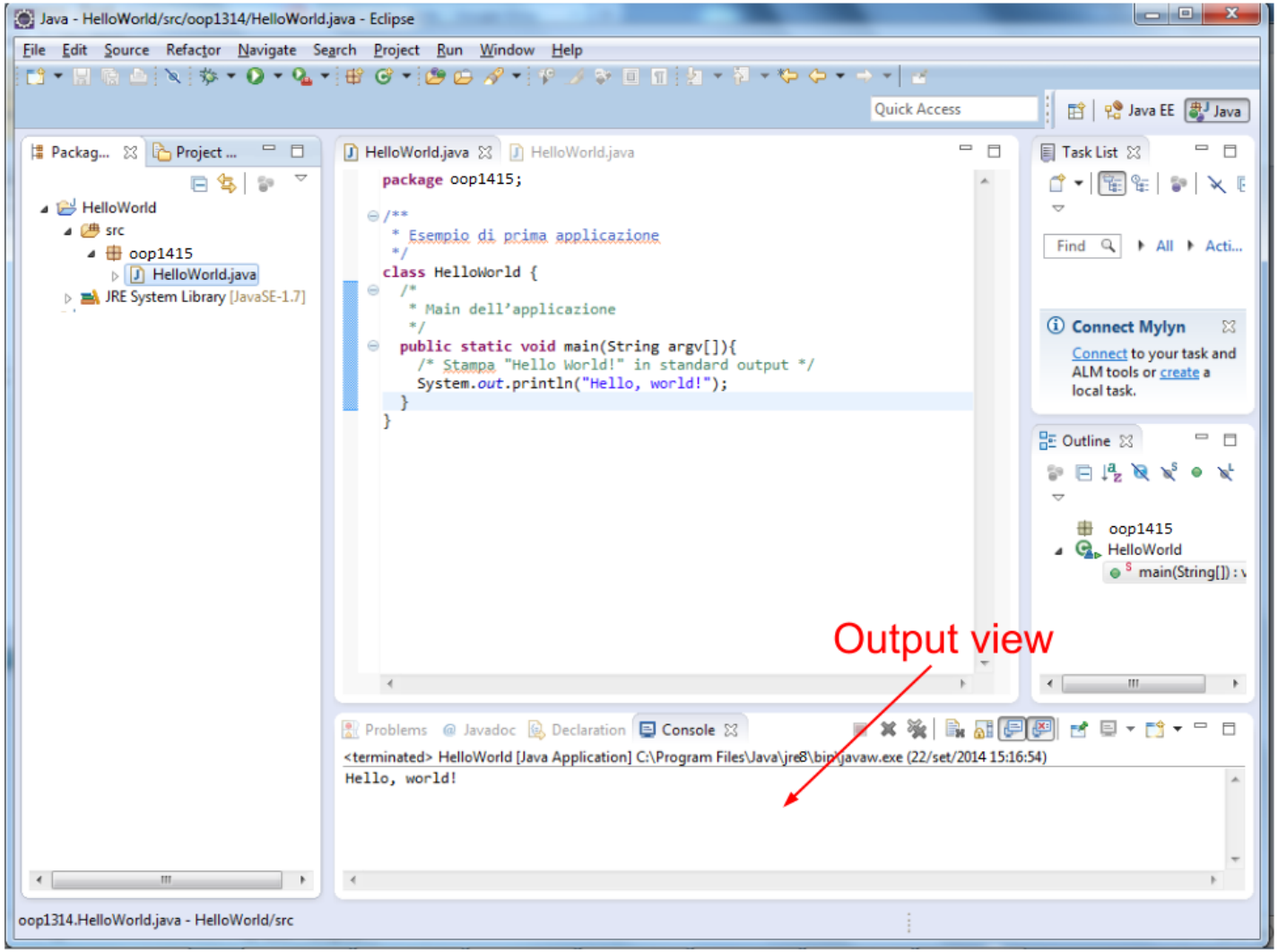
Esecuzione di Applicazioni: Un Alternativa 1/2



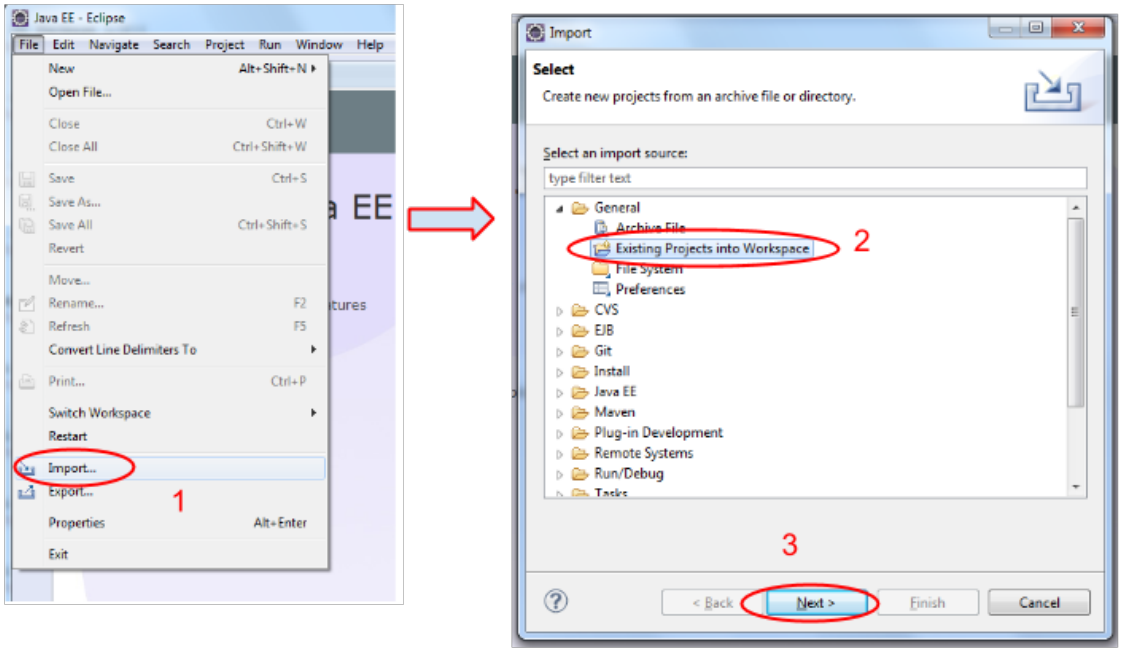
Esecuzione di Applicazioni: Un Alternativa 2/2



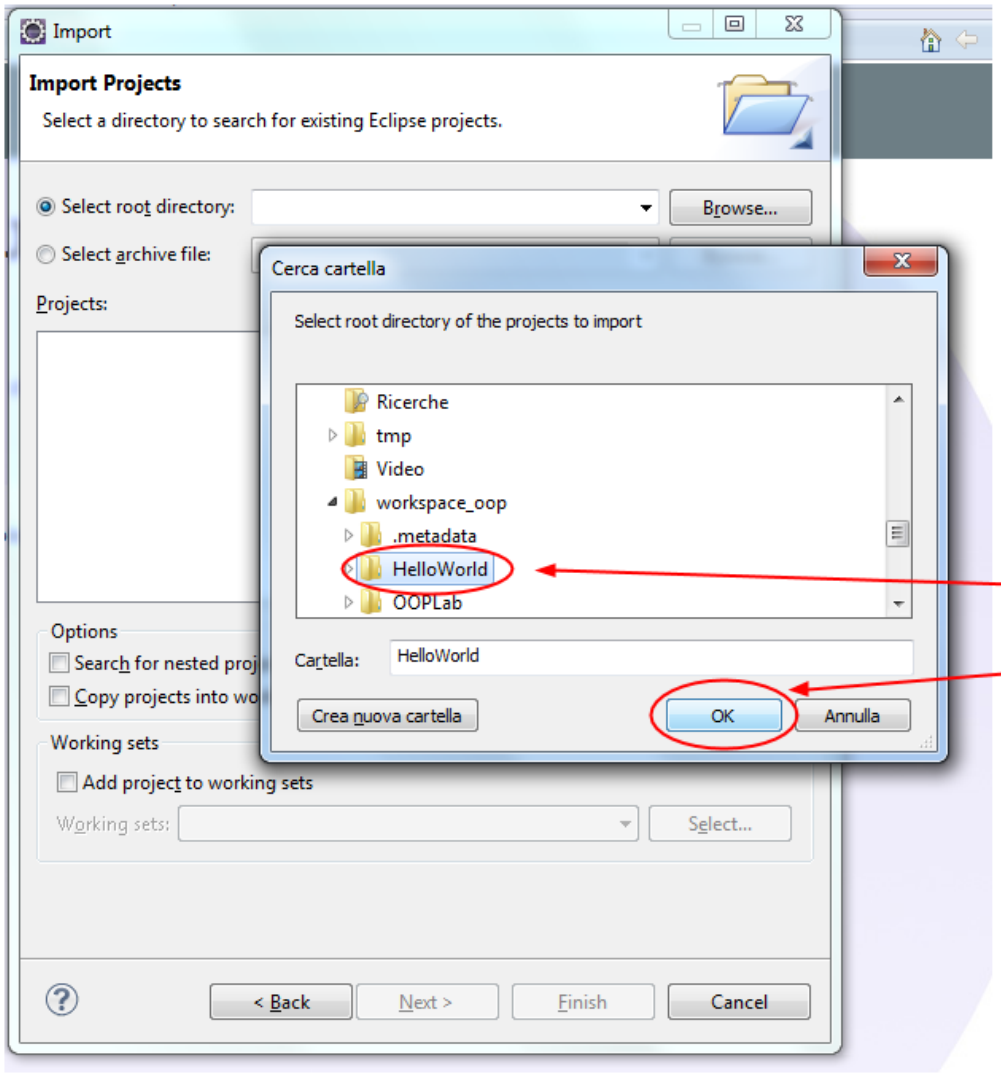
Esecuzione di Applicazioni: Output



Importazione di Progetti 1/2



Importazione di Progetti Esistenti 2/2



Scelta della directory contenente il prj, poi click sx

Outline

Eclipse Architecture

Eclipse ha una architettura organizzata a plug-in

- Un plug-in è un modulo (classi + meta-informazioni + manifest file) auto-contenuto che incapsula una certa funzionalità
  - File editor specifici, wizard, build, compilazione e debugging di sorgenti
  - Costituisce l'unità funzionale elementare dell'architettura
- Possono essere definite relazioni di dipendenza, composizione, etc. specifiche dell'astrazione plug-in (non parte dell'OOP)
  - Non le tratteremo nel dettaglio in questo corso

Eclipse come IDE Modulare

- Nuovi plug-in possono essere liberamente installati al bisogno
  - Ne esistono moltissimi open-source, altri sviluppati da aziende e università
- Supporto nativo lo sviluppo di nuovi plug-in
  - Plug-in Development Environment (PDE)

Plug-in and Perspective

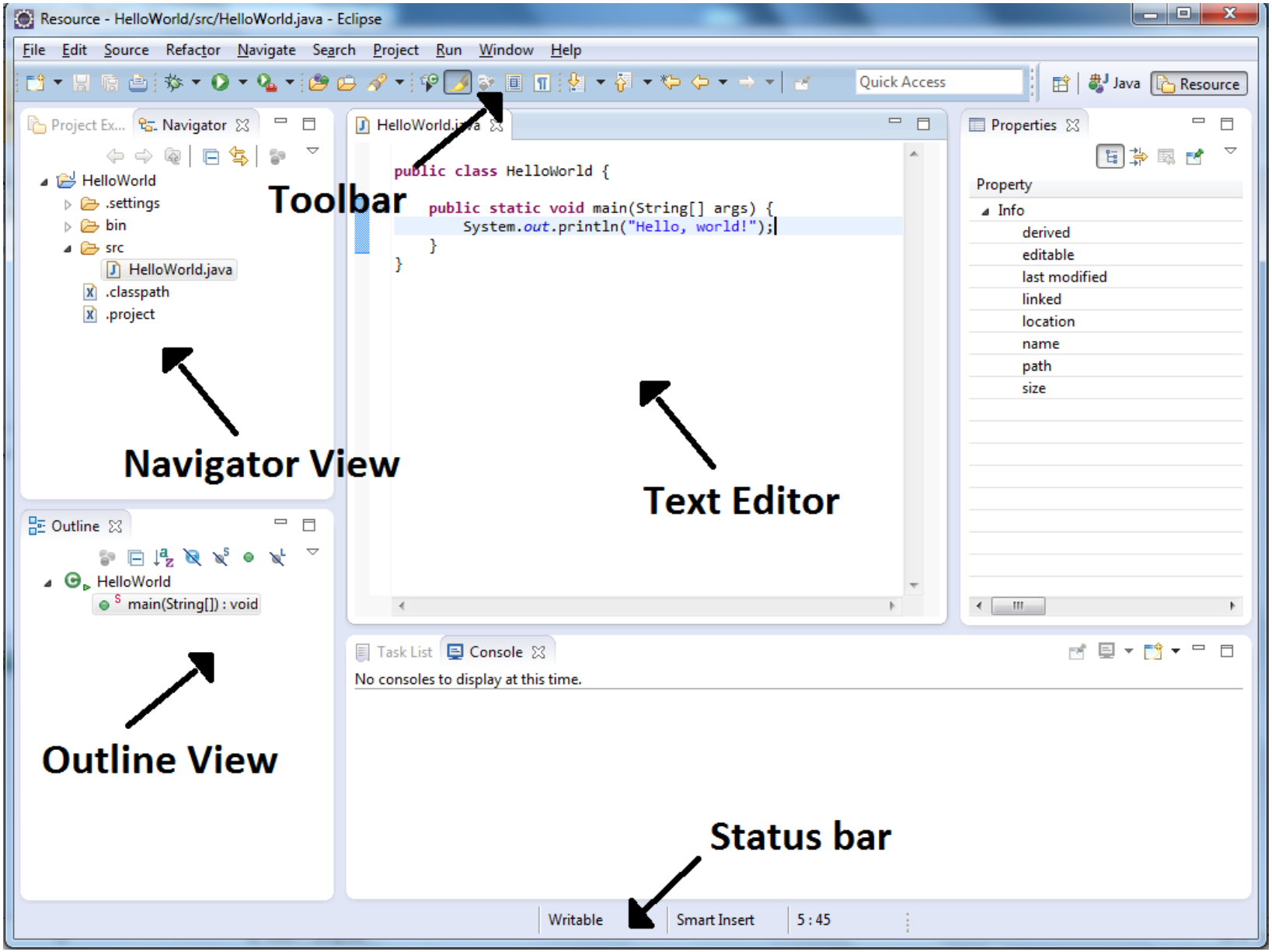
Che relazione tra plug-in e perspective?

- Le prospettive disponibili sono definite dai plug-in installati
- Prospettive disponibili di default (e.g. Resource, Java, Debug, Git,..)

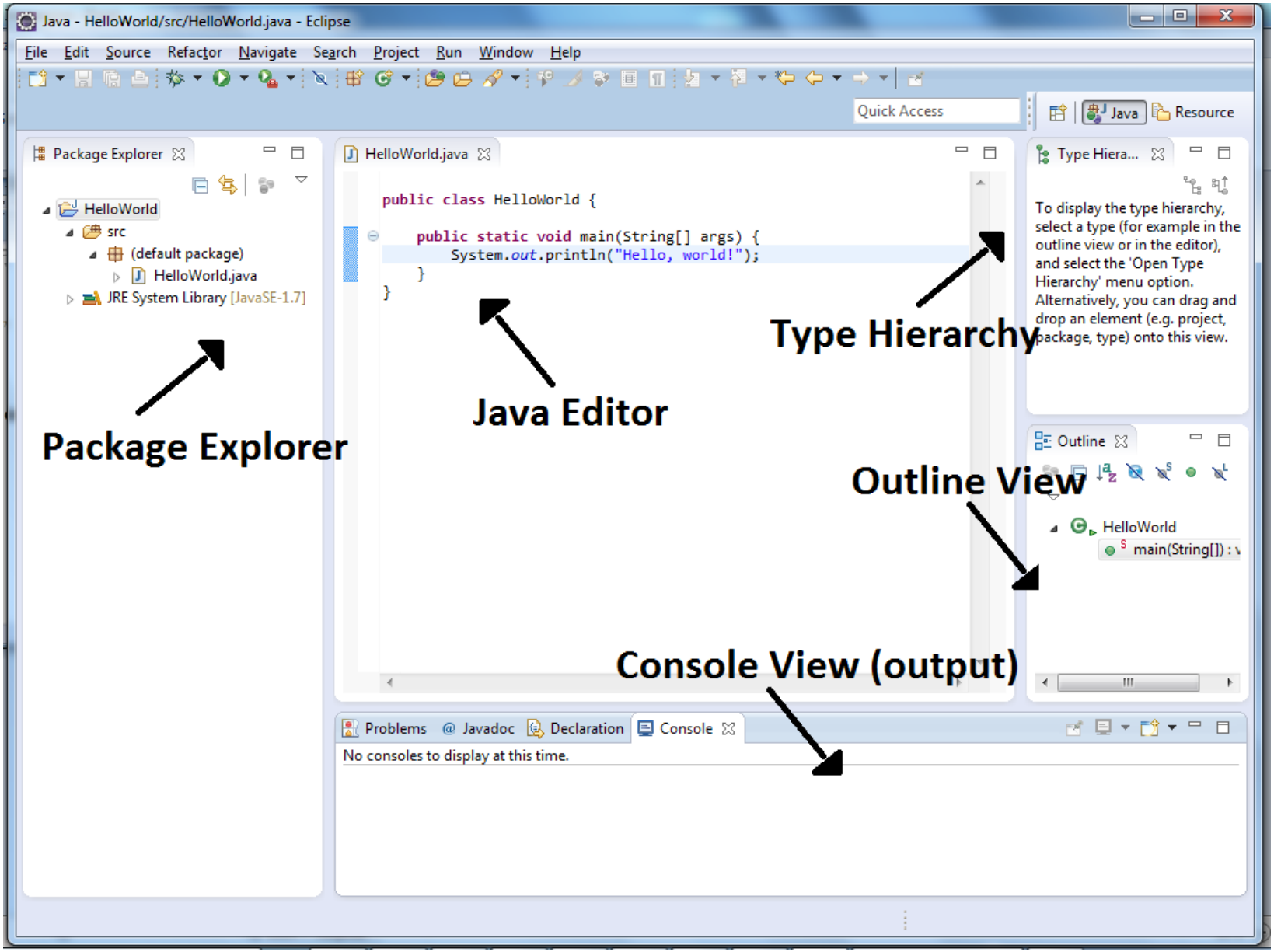
Descrizione di alcune delle perspective di default

- Resource perspective: permette di navigare e interagire con le risorse in termini di file
  - Il progetto è visto come una collezione di file e directory, che è possibile creare, modificare, spostare, etc.
- Java perspective: fornisce viste ed editor per gestire tutte le attività più significative per lo sviluppo di progetti Java
  - Realizzata da un insieme di plug-in che prendono il nome di JDT (Java Development Tools)
- ...

Resource Perspective (Default Perspective)



JDT e Java Perspective



Outline

Refactoring

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

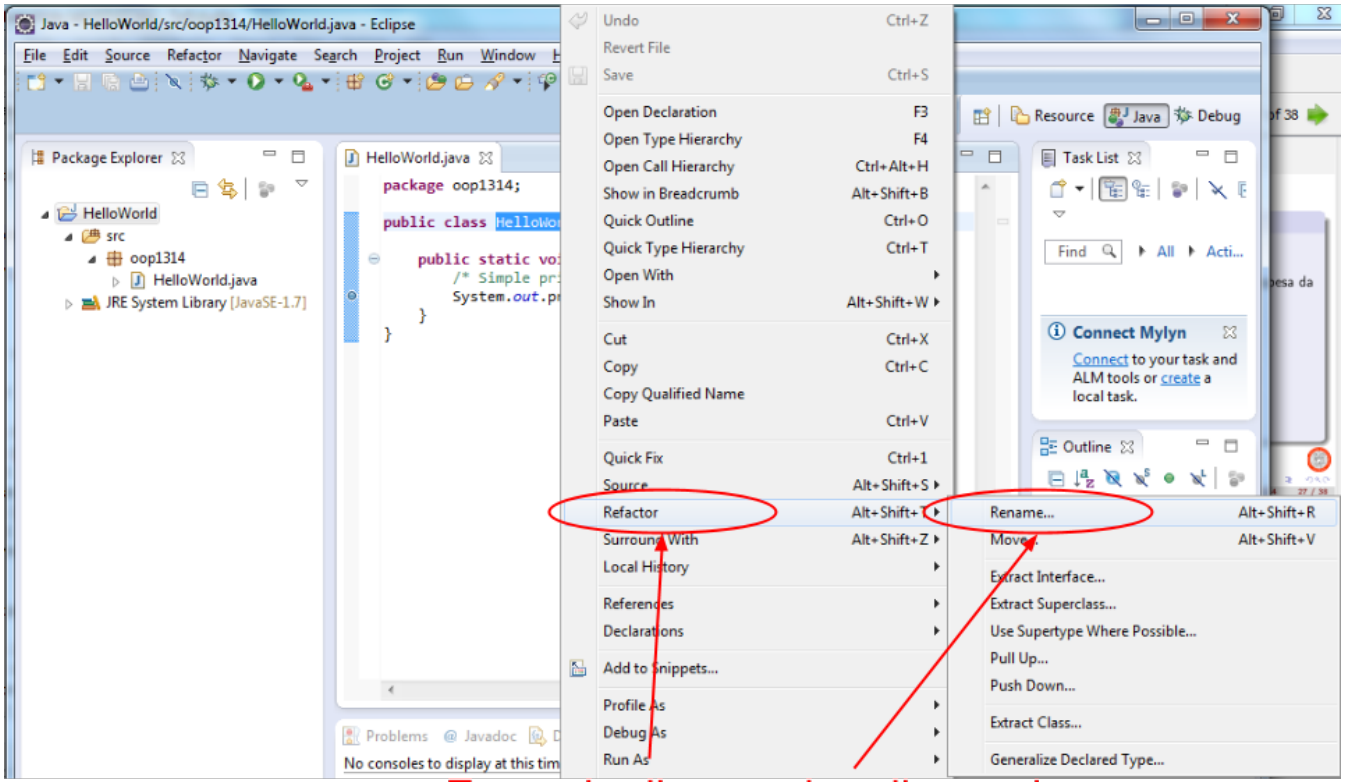
Vantaggi nel refactoring supportato dall'IDE

- 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

Operazioni di modifica del nome di variabili, classi, metodi etc.

- Gestite in maniera automatica e safe dall'IDE
- Attivabile anche tramite ALT+SHIFT+R sulla selezione di interesse



Esempio di renaming di una classe

Outline

Keyboard Shortcuts 1/2

Perchè usarle?

- Velocizzano l'esecuzione di operazioni ricorrenti
- Diminuiscono la probabilità di *"infortuni"* legati all'uso del mouse
  - I tennisti hanno problemi al gomito, gli informatici al tunnel carpale
  - (più terminale $\Rightarrow$ più salute)

Principali shortcut (per Mac OSX sostiuire CMD a CTRL)

- 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

Keyboard Shortcuts 2/2

Altre shortcut (per Mac OSX sostiuire CMD a CTRL)

- CTRL+SHIFT+O: include in automatico tutti gli import necessari sulla base delle classi utilizzate nel sorgente corrente
- CTRL + .: sposta il cursorse 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.

Outline

Debugging di Applicazioni [1, 2] I

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

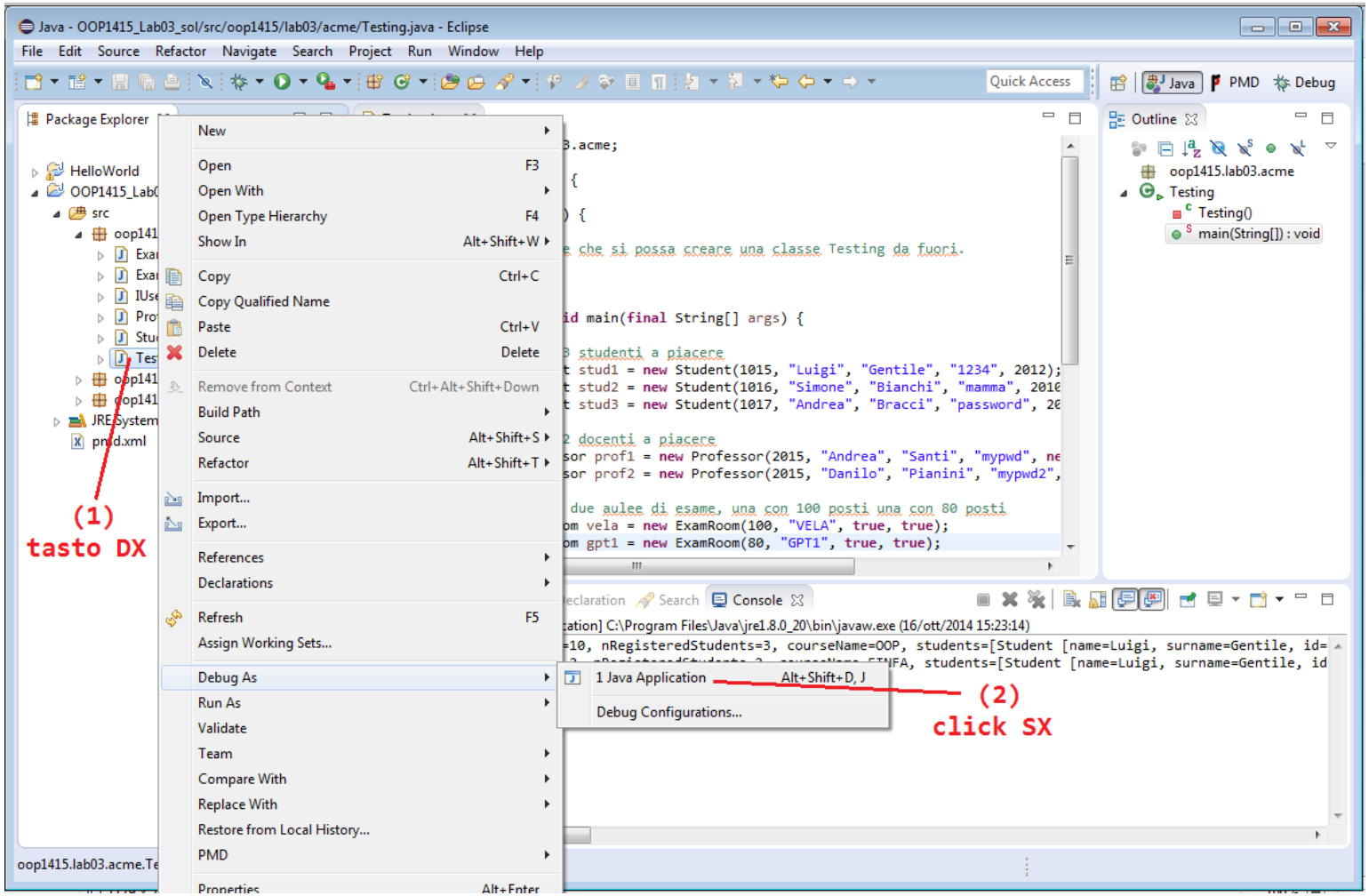
La capacità, a flusso di lavoro sospeso (e, quindi, una volta incontrato un breakpoint) di proseguirla andando avanti di un'istruzione alla volta

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.

Debugging in Eclipse: strumenti III

Iniezione di codice a caldo

Nel caso in cui si effettui una modifica al software mentre è in esecuzione in modalità debug e lo si ricompili (o si abbia la modalità di autocompilazione attivata), Eclipse tenterà di "iniettare" i nuovi compilati all'interno della JVM in esecuzione:

- Viene "scaricata" la classe precedentemente inserita nel classpath
- Viene "iniettata" la nuova classe

L'operazione può fallire, nel qual caso l'IDE notificherà che ci sono in esecuzione classi e metodi obsoleti. Per massimizzare la riuscita:

1. Scegliere uno stack frame precedente all'uso della classe da sostituire;
2. Usare la funzione "Drop to frame" per riportare il flusso di controllo all'inizio dello stack frame che si sta eseguendo;
   - Eventuali side effects non vengono cancellati!
   - (altro esempio in cui l'immutabilità è d'aiuto)
3. Solo a questo punto salvare il sorgente modificato.

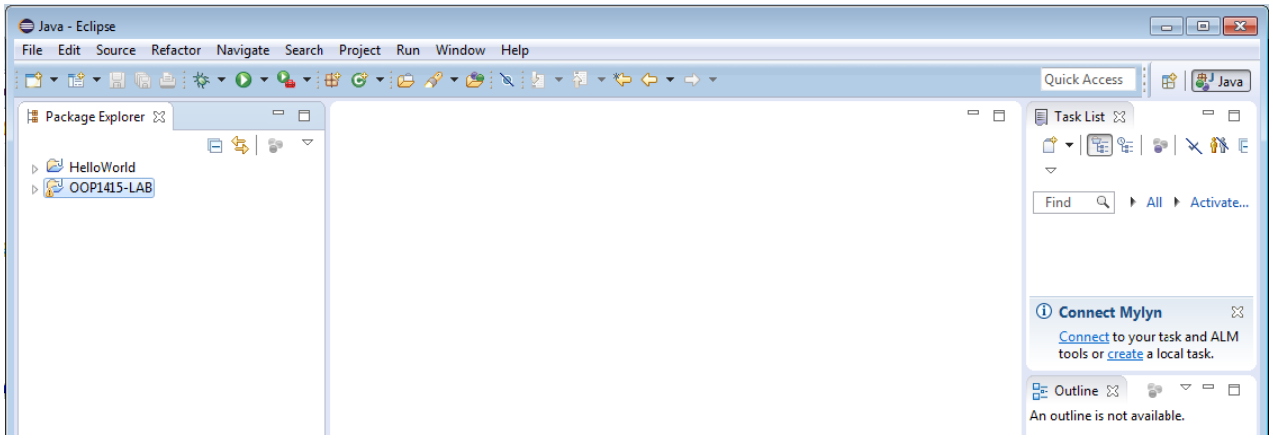
Outline

Preparazione Ambiente di Lavoro 1/2

- Accendere il PC
- Loggarsi sul sito del corso
  - `https://bit.ly/oop2017cesena`
- Scaricare dalla sezione `lab` del sito il file `lab03.zip` contenente il materiale dell'esercitazione odierna
- Spostare il file scaricato sul Desktop
- Decomprimere il file usando 7zip (o un programma analogo) sul Desktop

Preparazione Ambiente di Lavoro 2/2

- Copiare la cartella scompattata all'interno del workspace di Eclipse
- Importare il progetto `lab03` con la procedura di importazione dei progetti mostrata nelle slide precedenti
- Al termine della procedura l'ambiente di lavoro sarà circa come segue



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.