ECE355 Tutorial
Eclipse Setup

Presented by Igor Ivković
iivkovic@swen.uwaterloo.ca

Why Eclipse?

- Eclipse platform is an open-source (free) integrated development environment (IDE) that is fully extensible
- Eclipse comes with many built-in features for simplifying and streamlining Java development
- Eclipse views and perspectives help separates areas of concern in development of complex software systems
- Eclipse is the basis for many commercial IDEs such as Rational XDE, WebSphere Application Developer, etc.

Agenda

- ➢ Installing Eclipse
- o Setup and Structure
- o Referencing and Navigation
- o Debugging in Eclipse
- o Refactoring and Formatting
- o Project Website and References

Installing Eclipse

- You can run Eclipse from the NEXUS machines in the labs, or you can install Eclipse on your Windows/Linux machine
- To Install Eclipse on your own machine, you will need
  - Eclipse SDK v3.0 or higher from http://www.eclipse.org/downloads/index.php
  - Java Runtime Environment v1.4.2 or compatible JRE/JDK http://java.sun.com/j2se/1.4.2/download.html

Eclipse Workspace

- Once the installation is complete, you must select your workspace folder where your projects are stored
- You can store multiple projects in the same workspace but in different subfolders
- Eclipse supports multiple workspaces and you can select the workspace when you start Eclipse

Agenda

✓ Installing Eclipse

➢ Setup and Structure

o Referencing and Navigation

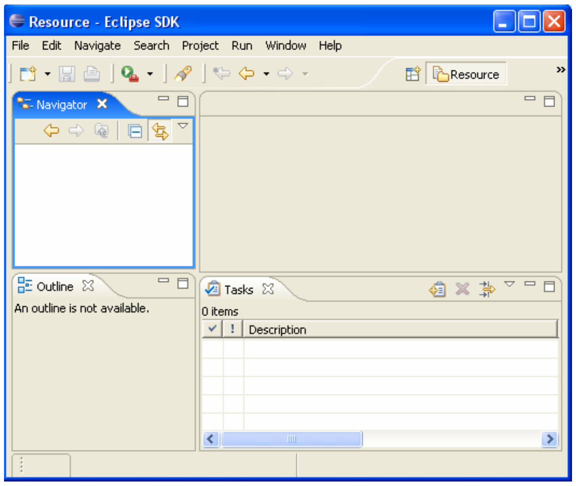
o Debugging in Eclipse

o Refactoring and Formatting

o Project Website and References

Eclipse Setup

- Once the Eclipse is running, you should see the following:



An outline is not available.

0 items

First Eclipse Project

- To create a project in Eclipse, select File -> New -> Project
- To create a Java project, select Java Project -> Next -> Enter Project Name (FirstEclipseProject) -> Select Create Separate Source and Output Folders -> Finish
- Once completed, Eclipse will ask you to switch to the Java Perspective, select Yes and then perspective will open
- Note different Views and Perspectives under the Window menu by selecting Open Perspective or Show View
- Expand FirstEclipseProject under PackageExplorer view to observe the empty project structure

First Eclipse Class

- To create a Java class in Eclipse, select File -> New -> Class -> Enter Package Name (ca.uwaterloo.firstEclipsePackage) -> Enter Class Name (FirstEclipseClass) -> Check public static void main -> Uncheck Inherited abstract methods -> Finish
- Observe the source code file opened in the main view
- Under the main method, enter the following System.out.println(“My First Eclipse Class is Running");
- Save the file through File -> Save or using CTRL+S
- Execute the main class by selecting Run -> Run… -> Java Application -> New -> Run, and see results in Console view

Agenda

✓ Installing Eclipse

✓ Setup and Structure

➢ Referencing and Navigation

o Refactoring and Formatting

o Debugging in Eclipse

o Project Website and References

Importing External Packages

- Under the FirstEclipseClass, create a new method public static void printVector(Vector input)
- Save the file and observe term Vector underlined in red and under Problems listed as “cannot be resolved”
- Select Source -> Organize Imports or select CTRL+SHIFT+O
- Save the file and note that Vector was resolved and that a new package java.util.Vector was imported
- Alternative: instead of Source -> Organize Imports, highlight term Vector and select Source -> Add Import

Navigating External Types

- Under the printVector method, type “for” and press CTRL+Space to open type navigator
- To reference printVector body from the main method, hold CTRL and click on printVector; the same holds for attributes
- Select Iterate over array and then replace “array.length” with “input.” to get a listing of type members for Vector
- Select size() and in the body of the for loop type the following and observe the type navigator open after each dot is typed System.out.println(input.get(i));

Navigating Type Hierarchies

- To get more information about a specific type, you may select it and open it in different views
- For example, highlight Vector and select Navigate -> Open Type Hierarchy
- If the code is available, you can also select Navigate -> Open Declaration; this command also works for attributes
- You can also see all call declarations using Navigate -> Open Call Hierarchy; this command also works for methods

Source Code Browsing

- As you are moving from file to file using links and references, it may be necessary to go back and forth and use bookmarks
- To go back or forward, select Navigate -> Back or Navigate -> Forward, or ALT+Left or ALT+Right
- You may also use left or right arrows above the source code viewer, with browsing history as drop-down menus
- To go back to the last edit location, select Navigate -> Last Edit Location, or CTRL+Q

Agenda

- ✓ Installing Eclipse
- ✓ Setup and Structure
- ✓ Referencing and Navigation
- ➢ Debugging in Eclipse
- o Refactoring and Formatting
- o Project Website and References

Start the Debugging

- Under main method remove the print statement, create a new Vector of four strings: “A”, “B”, “C”, and “D”, and print the strings using the printVector method using
  Vector input = new Vector();
  input.add("A"); input.add("B"); input.add("C"); input.add("D");
  printVector(input);
- To the left of the print statement in printVector, click on the blue area to insert a break point
- Select Run -> Debug Last Launched and observe the Debug Perspective open (select Yes to switch over)

Stepping through the Code

- Once the debugging perspective opens, the line with the print statement should be highlighted in green
  - To step into the called method, enter F5
  - To step over to the next line, enter F6
  - To step out to the caller, enter F7
  - To resume regular execution, enter F8
- To terminate, click on Terminate (red button) or select Run -> Terminate; do not forget to this if the session does not close

Inspecting Variables

- In the debugging perspective in the top right there is a Variables view
- Using this view, you can view values of primitive data types and expand and view the members of complex data types
- You can also view the values of primitive data types by moving your mouse over the variable name in the code
- Finally, when finished debugging do not forget to close or terminate the current session to prevent waste of resources

Debugging Threads

- Eclipse allows you to run multiple process at the same time
- Debug view within the debug perspective allows you to:
  - View which processes are running
  - Switch between threads for debugging
  - Go up and down the call stack for each process
- Each call stack placement provides its own inspection context for inspecting primitive and complex variables

Agenda

✓ Installing Eclipse

✓ Setup and Structure

✓ Referencing and Navigation

✓ Debugging in Eclipse

➢ Refactoring and Formatting

o Project Website and References

Refactoring

- Right click on the definition of the printVector method and select Refactor -> Change Method Signature
- Change Method Name to printVectorElements, add a new parameter count of type int with value null, and click Preview
- Observe the conflict with the call in main method
- Select Back -> Enter Value of 3 -> Preview (all fine) -> OK
- Under printVectorElements, replace input.size() with count and then run; observe only A, B, and C now printed

Formatting

- To make code more readable and easier to navigate, it may be necessary to reformat it
- Eclipse provides automatic reformatting of files and projects using Source -> Format or CTRL+SHIFT+F command
- To reformat now, right click on the FirstEclipseProject in the Package Explorer and click Format; note the source changes