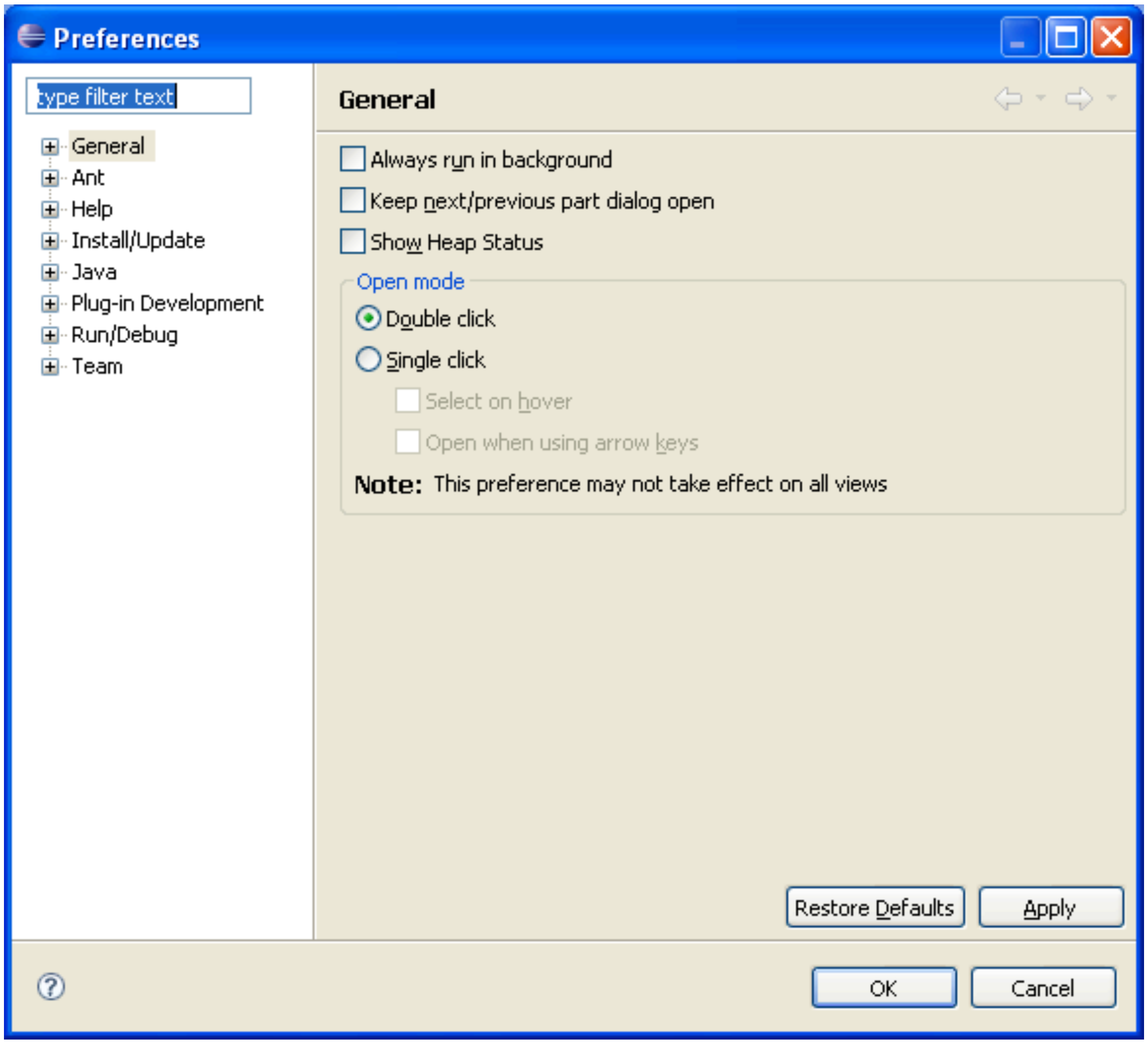
Tutorial 04: Preferences, Launching, and Debugging

Contents:

1. Showing the Preferences window.
2. Editing project specific settings.
3. Launching a Java application.
4. Debugging a Java application.

Showing the Preferences window

1. Click Window > Preferences…



General

☐ Always run in background

☐ Keep next/previous part dialog open

☐ Show Heap Status

Open mode

● Double click

○ Single click

☐ Select on hover

☐ Open when using arrow keys

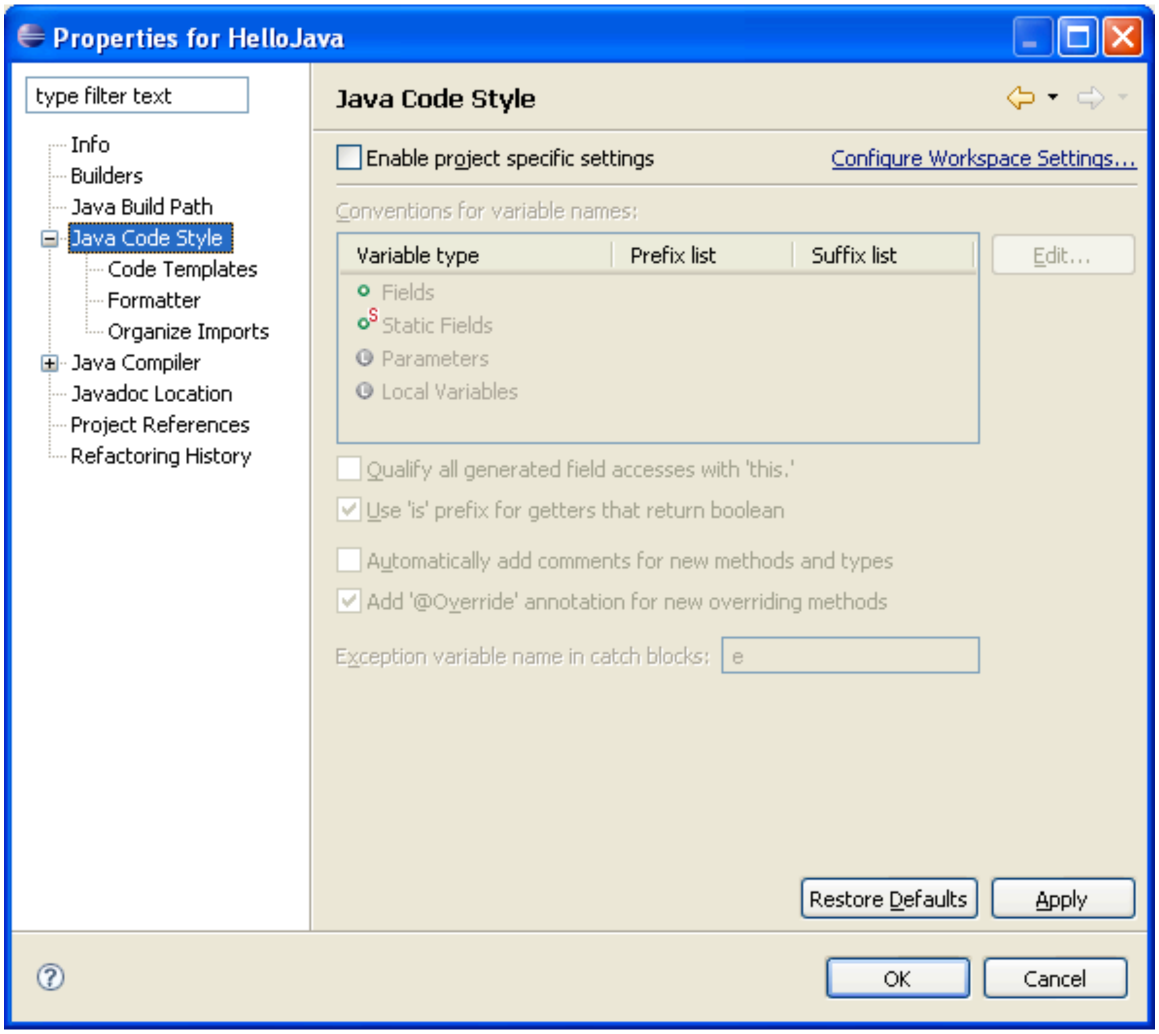
Note: This preference may not take effect on all views

2. Click Help > Help Contents.
3. Click Workbench User Guide > Reference > Preferences.

4. Explore and examine Eclipse Preferences. Feel free to change them.

Editing project specific settings

1. Select and right-click HelloJava in the Package Explorer view, and click Properties.


Properties for HelloJava

type filter text

- Info
- Builders
- Java Build Path
- Java Code Style
  - Code Templates
  - Formatter
  - Organize Imports
- Java Compiler
- Javadoc Location
- Project References
- Refactoring History

Java Code Style

[ ] Enable project specific settings   Configure Workspace Settings...

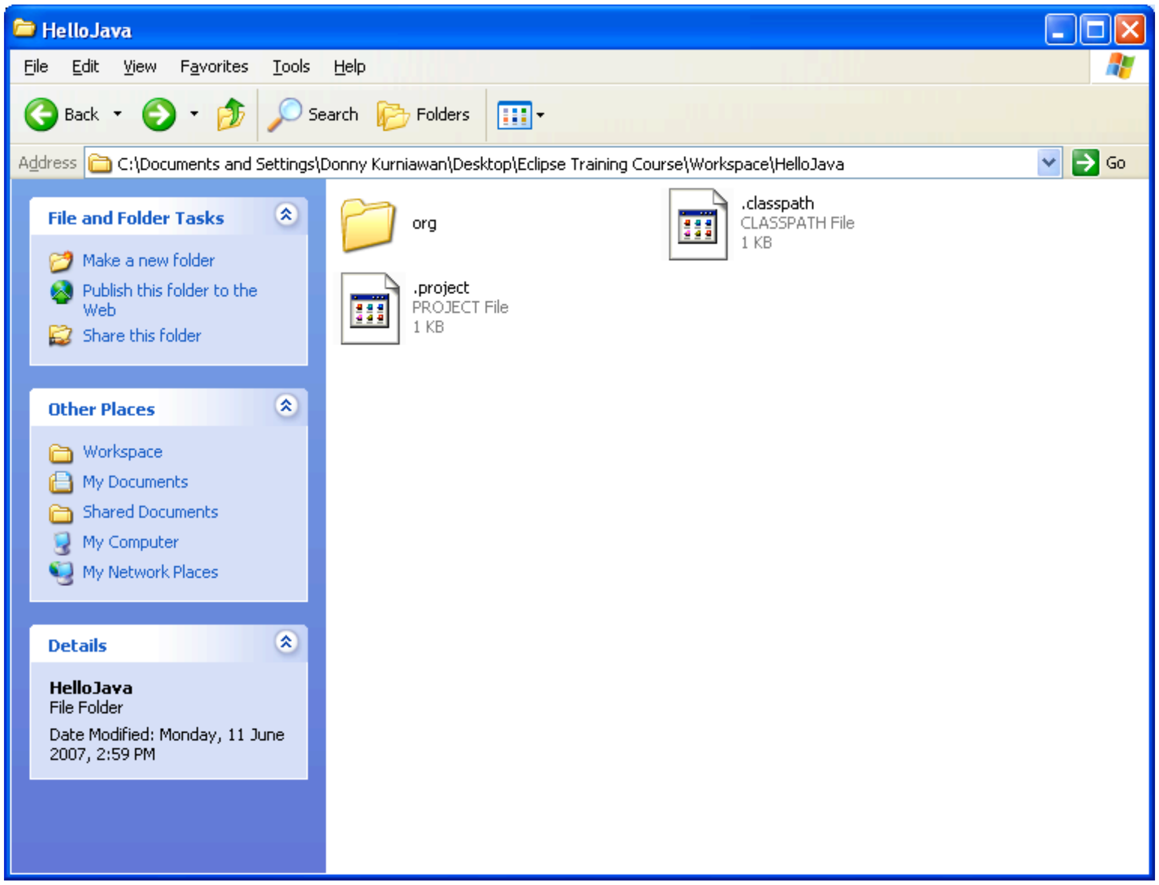
Conventions for variable names:

Variable type	Prefix list	Suffix list
Fields		
Static Fields		
Parameters		
Local Variables		

[ ] Qualify all generated field accesses with 'this.'
[✓] Use 'is' prefix for getters that return boolean
[ ] Automatically add comments for new methods and types
[✓] Add '@Override' annotation for new overriding methods

Exception variable name in catch blocks: e

2. Examine the available options.
3. Open Windows Explorer and navigate to the project folder (HelloJava).



Eclipse stores the project configuration in two XML files (.project and .classpath).

4. Use Notepad to view the raw .classpath file.

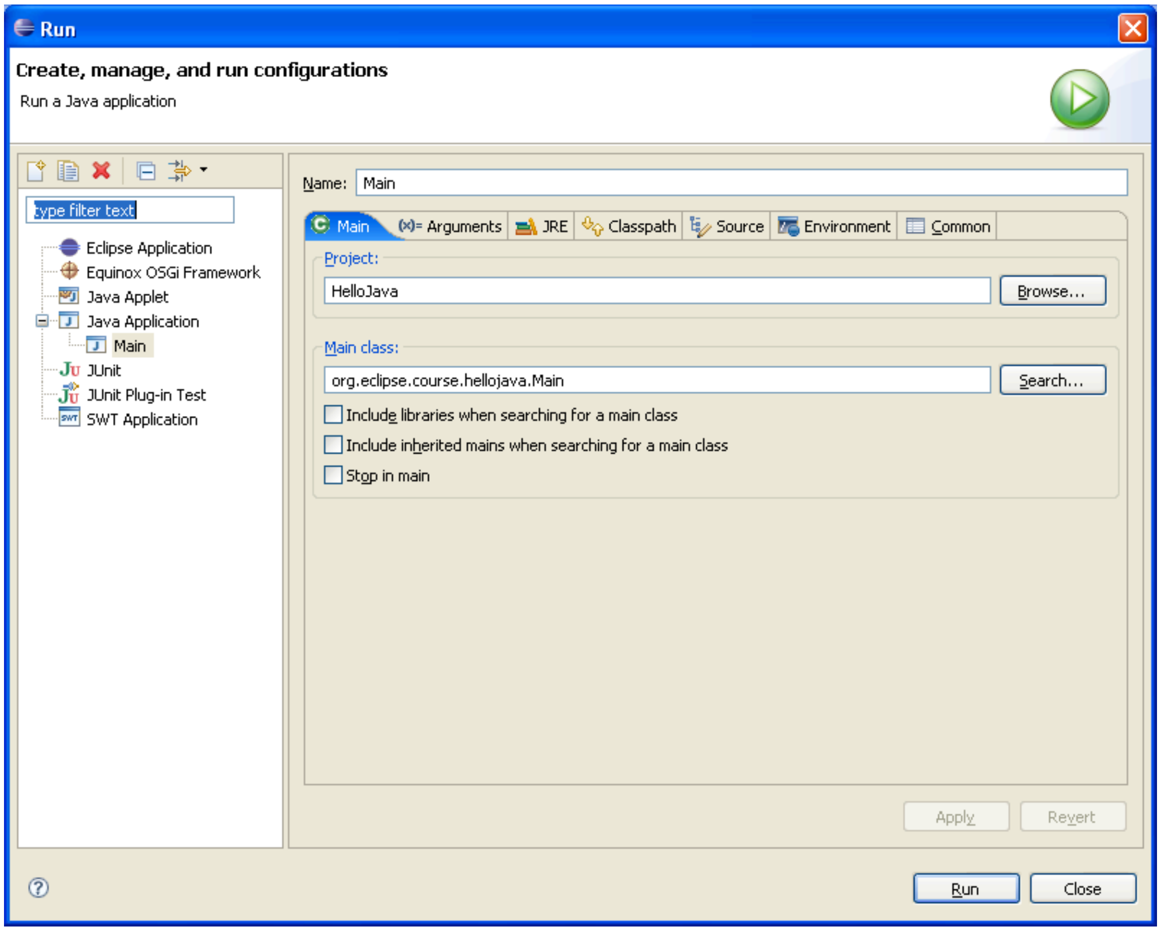
Rather than editing the .classpath file directly, Eclipse provides a more user-friendly approach. Right-click on the project and select Properties. In the Properties dialog, select Java Build Path.
“Java classpath” is a generic term describing both the classpath used at compile-time and the classpath used at runtime. In Eclipse, the compile-time classpath is called the Java build path. When you are running or debugging Java application code, the runtime classpath is determined by the launch configuration (see the next section).

5. Use Notepad to view the raw .project file.

The .project file provides a complete description of the project suitable for recreating it in the workbench if it is exported and then imported.
The nature tag indicates what kind of project this is. The nature, org.eclipse.jdt.core.javanature, indicates that it is a Java project

Launching a Java application

1. Open Main.java in the editor. Notice that in the Outline view, the Main class (or any Java application with a main() method) is marked with the runnable icon decoration (a small green triangle).
2. Click Run > Run As > Java Application. The output (in blue) and error text (in red) are displayed in the Console view.
3. Click the Run toolbar button to execute the application again.
4. Click Run > Run… to open the Launch Configuration window.



Name: Main

Project: HelloJava

Main class: org.eclipse.course.hellojava.Main

☐ Include libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

5. Click Java Application > Main

Whenever you run an application for the first time, a launch configuration is created. A launch configuration records the information needed to launch an application. In the case of Java applications, the launch configuration can be opened to modify the name of the Java class, the virtual machine arguments, the JRE, and so forth.

6. Click the various tabs in the window.

The Main tab specifies the project and class to be run; the Arguments tab records the program parameters (as space-separated strings) and VM arguments; the JRE tab specifies the JRE to use (it defaults to the JRE specified in your Java > Installed JREs preferences); the Classpath tab is used to override or augment the default classpath used to find the class files needed to run the application; the Source tab specifies the location of the source files used to display the source of an application while debugging; the Environment tab is used to set environment variables; and the Common tab records information about where the launch configuration is stored and where standard input and output should be directed.

Debugging a Java application

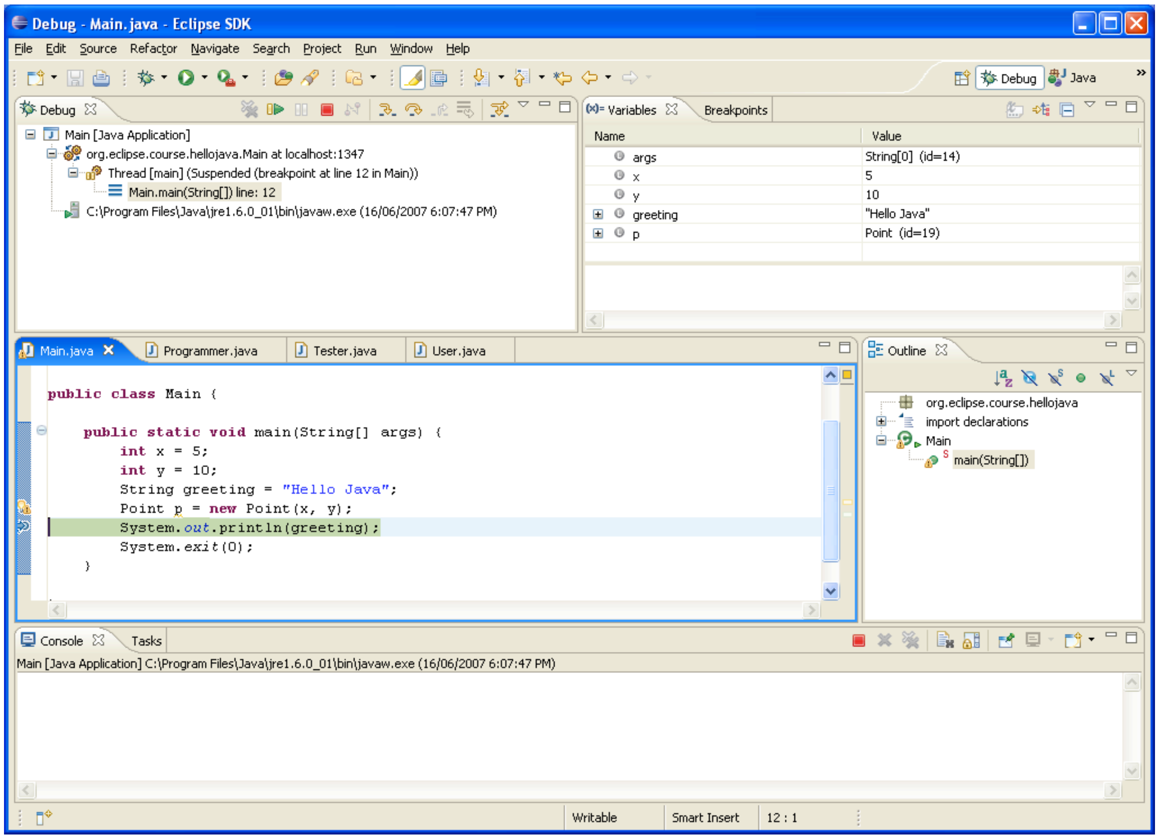
1. Open Main.java in the editor
2. Type the following code in Main.java

```
package org.eclipse.course.hellojava;

public class Main {

     public static void main(String[] args) {
          int x = 5;
          int y = 10;
          String greeting = "Hello Java";
          Point p = new Point(x, y);
          System.out.println(greeting);
     }

}
```

3. Notice that there is an error marker. You can click it to import java.awt.Point or you can click Source > Organize Imports (on the menubar) to let Eclipse imports the necessary classes automatically.
4. Double-click the marker bar to the left of the line “System.out.println” to place a breakpoint. A breakpoint marker appears next to the appropriate source code line.
5. Click the Debug toolbar button.

If you encounter an error: ERROR: JDWP Unable to get JNI 1.2 environment
Put System.exit(0); at the end of your main method(). It’s a well known Java 1.6 bug.

6. You can choose whether you want to switch to the Debug perspective automatically or not when Eclipse encounters a breakpoint.



```
Main [Java Application]
  org.eclipse.course.hellojava.Main at localhost:1347
    Thread [main] (Suspended (breakpoint at line 12 in Main))
      Main.main(String[]) line: 12
  C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (16/06/2007 6:07:47 PM)
```

Name	Value
args	String[0] (id=14)
x	5
y	10
greeting	"Hello Java"
p	Point (id=19)

```
public class Main {

    public static void main(String[] args) {
        int x = 5;
        int y = 10;
        String greeting = "Hello Java";
        Point p = new Point(x, y);
        System.out.println(greeting);
        System.exit(0);
    }
```

```
org.eclipse.course.hellojava
  import declarations
  Main
    main(String[])
```

```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (16/06/2007 6:07:47 PM)
```

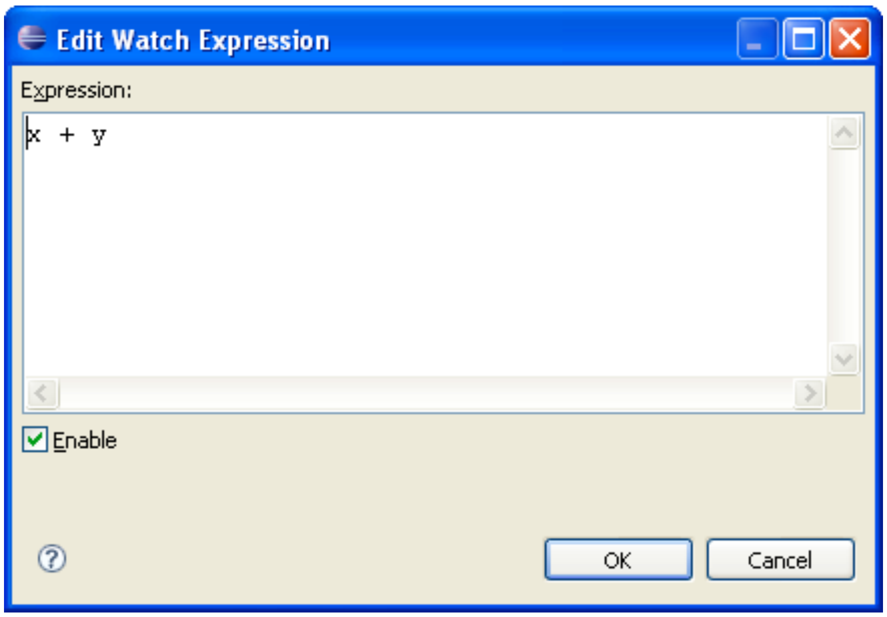
7. Click the various debug buttons.

The Resume button (also the F8 key) in the Debug view resumes the execution of a program until it either ends on its own or encounters another breakpoint, while the Terminate button stops execution of a program entirely. The Step Into button (also the F3 key) executes the next expression in the highlighted statement, while the Step Over button (also the F6 key) steps over the highlighted statement and stops on the next statement.

8. Examine the Variables and the Breakpoints views.
9. Click Window > Show View > Expressions.
10. Select “y” in the editor and right click it, and click Watch. Notice that there is a new entry in the Expressions view.

If you select the Inspect command, a popup window containing the results of the expression appears. Pressing Ctrl+Shift+I will move the results to the Expressions view. As with the Variables view, primitive variable types show their values directly while object types can be expanded to show their individual elements.

11. Right-click the entry and select Edit Watch Expression…



```
x + y
```

12. Change the text to “x + y” and click OK. Notice that the entry in the Expressions view gives 15 as the correct answer.