Seminar 2

Eclipse

PROGRAMMING 3

September 2014

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)
Department of Computer Languages and Systems
University of Alicante

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Contents

1 Installing

2 Environment
- Workspace
- Interface

3 Projects
- Creation

4 Classes
- Importing classes
- Class creation

5 Run
- Debug

6 Unit tests

7 Code generation

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents

Installing

Environment
- Workspace
- Interface

Projects
- Creation

Classes
- Importing classes
- Class creation

Run
- Debug

Unit tests

Code generation

Installing

- Browse to `www.eclipse.org`
- Download *Eclipse IDE for Java Developers*
- Uncompress and run `eclipse`

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

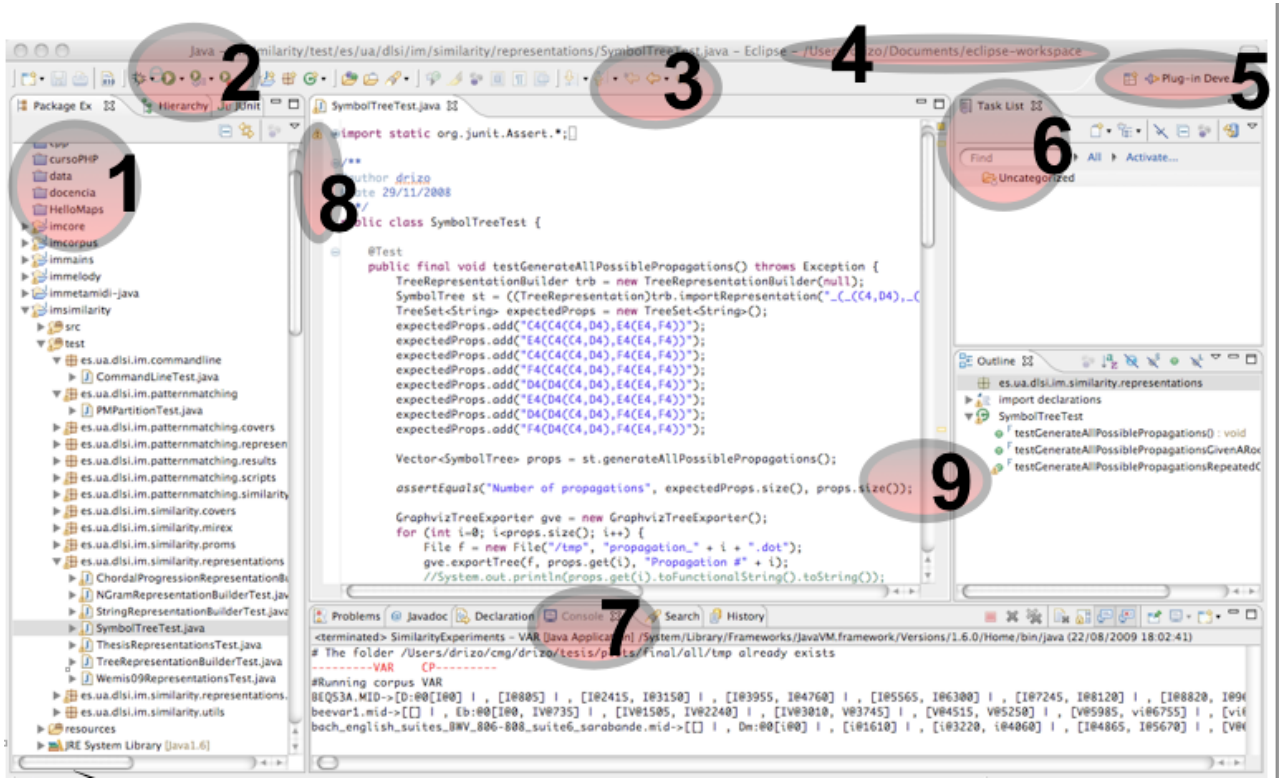
Workspace

- Eclipse stores all the configuration and projects under directory *workspace*
- When Eclipse starts, you are prompted to choose a workspace location
- Select `File>Switch workspace` to change workspace

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Interface



Tools

1. Projects and packages
2. Run and debug
3. File explorer
4. Current workspace
5. Perspective
6. A view: tasks
7. Console
8. Breakpoints
9. Errors, warnings, TO-DO

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Project creation

- `File > New > Java project`
  - Project name
  - Choose a new directory or accept the default one
- A directory will be created containing:
  - `bin`, `src`
  - Hidden files `.project` and `.classpath`
    - These files contain project metadata.
  - When moving projects to a different machine, these files will be used by Eclipse to identify directories containing projects
  - To import a project, select `File > Import > General > Existing Projects into Workspace` and choose the project directory

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Importing classes

To import external `.java` files, copy them to the clipboard under a file explorer and paste them into package view.

Task

Add the source files from your first assignment to the `src` directory of your new project; create packages when necessary.

Classes

- Use `File > New > Class`
- Introduce name, package, and, optionally, if you want an empty `main` method to be added

Task

- Create a new class, open it in the editor, and add an integer field. Type `/**` before the declaration of the attribute, hit *enter* and write the *javadoc* documentation.
- Create the class constructor and document it.
- In case your code contains errors, use the hints on the left edge of the code editor.

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Run

- Since a particular project may include more than one class definition with a `main` method, the easiest way is to right-click on the class containing the `main` method to run and select `Run as > Java application`.
- This will create a new run configuration (menu `Run > Run configurations`), which can be edited to add command-line parameters to your program.

Task

A similar process may be performed at command-line:

- Open a terminal
- Move to the project directory
- Run `java -cp bin mains.Main` (Eclipse automatically compiles your source files and stores the resulting class files in directory `bin`). Replace `mains.Main` with the correct name if other.

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Debug

- Select `Run > Debug` (there is a button for this in the toolbar as well) to run your application in debug mode.
- To set a *breakpoint*, walk through the code and place your cursor on the marker bar (along the left edge of the editor area) on the line with the suspected code; double-click to set the breakpoint.
- Notice that Eclipse has switched to the *Debug* perspective.

Task

Run the `main` method line by line.

JUnit

- Files containing unit tests will be under a different directory.
- Create a directory `test` in your project by right-clicking on the project name in package view and selecting `New > Source folder`
- Paste into *test* the files containing some tests used in the evaluation of the first assignment.
- Make your project use the JUnit framework (`Project / Properties / Java Build Path / Libraries / Add Library`).
- Run tests by right-clicking on the class name and choosing `Run as > JUnit test`

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

JUnit

Task

- Open the source file with the tests.
  - It contains a number of methods with annotations like `@Before` (for test configuration) and `@Test` (for code verification).
  - `assertEquals` checks whether expected and current value match. Parameters are: title (optional), expected value, current value, absolute value of the difference allowed (optional; useful for real numbers) .
- Run a pass with no errors and one where some assertion does not hold. To detect the source of the issue, select panel `Failure trace`.

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

New unit test

To create a new unit test for a class, right-click on its name and select `New > JUnit test case`.

- Choose JUnit 4
- Type `test` (instead of `src`) in the directory field.

Task

- Write a new unit test for your new class which tests its constructor.
- To run all the tests, right-click on the project name and select `Run as > JUnit test`

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation

Code generation

- Implementation of some operations (e.g., `equals` or `toString`) is usually routine.
- Eclipse can write some draft excerpts of code for you; right-click on the source file and select `Source > Generate toString()` and `Source > Generate hashCode and equals()`.

Task

Automatically generate these methods for your new class.

David Rizo, Pedro J. Ponce de León, Juan Antonio Pérez (translator)

Contents
Installing
Environment
Workspace
Interface
Projects
Creation
Classes
Importing classes
Class creation
Run
Debug
Unit tests
Code generation