Eclipse Cheat Sheet
(version 1.0)

Pr. Olivier Gruber

The idea is to group here all the major tricks and tips I use everyday when coding in Java with Eclipse.

Installing Eclipse, see the web for that.

Read basics of Eclipse on the web:
Concept of Workspace.
Concept of Project.
Concept of Perspective.
Concept of View.
Concept of plugins.

And you have a good start...

Know that you can import existing Eclipse projects in a workspace, see also the web.

Global Preferences and setup:

Windows → Preferences → and you can setup a lot of interesting things.
For Java, you can setup pretty printing preferences,
I usually use 2 spaces instead of tabs of 4 spaces, source code is more compact.

Install a Java JDK, not a JRE... you will have the sources for the Java class libraries.
Make sure that Eclipse knows about your installed JDK:
Windows → Preferences → Java → Installed JRE

Also setup the Java compliance level for your Java project,
Windows → Preferences → Java → Compiler
so that it matches your installed Java JDK.

Updates and installing new plugins:
Help → Check for updates
Help → Install new software

Project preferences and setup:
Right-click on the project → properties
Pay special attention to Java Build Path, source folders, and libraries.
Notice that you can have one project depends on others, so you can structure your code in multiple projects when it makes sense.

Most useful shortcuts:

Ctrl-Shift-T: search for a class (wildcards do work)
Ctrl-Shift-G: select something, you will see where it is used
for example, select a method and you will see all the places it is called from.
Ctrl-E: select a tab.
Ctrl-Shift-W: close all opened tabs... cool once you have so many opened that you are lost.
Ctrl-B: rebuild all.
F3: select and navigate with F3
F4: select a class and see its hierarchy

Select a class that implements an interface or extends a super class, and for which the compiler complains that some methods are not implemented...
Crtl-1 → Add unimplemented methods
Voilà !

Refactoring: read on the web about it.
Alt-Shift-R: enables the renaming of the currently selected item (class, method, variable)

Degug:
F5: step in
F6: step over
F7: step out
F8: go
F11: relaunch last launch

Most useful functionality:
Windows → Project → Build / Clean / etc..
Windows → Editor → Toggle Split Editor
will give you two editors on the same file, great to see different parts of the same file at once.

Right-click → Compare
You can select multiple classes or files in the navigator view and compare them.

Navigator view:
Top button, like a dual yellow arrow → synchronize/de-synchronize the navigator with the currently selected source.

Global search → toolbar → the icon with a torch lamp.
Debug Configurations:
the bug icon in the toolbar, click its down arrow, select debug configuration...
you can basically tweak everything about the way a Java application is launched.

Breakpoints:
Breakpoints can be set on lines, rather traditional.
But breakpoints can also be setup on exceptions.
See the breakpoint view, in the debugger perspective, and click on the "!" button.

Refactoring... please read about Eclipse refactoring in Java on the web.

Eclipse as builtin support for JUnit:
When developing, you are writing your code, then you are debugging it, usually running some basic tests. Most of us approach testing through ad-hoc small programs... They tend to get lost... and more importantly, as your software evolves, you are not running the same tests systematically... so correcting new bugs might re-introduce bugs in parts of your code that you tested already.

JUnit is useful for two main reasons. First, it gives you a simple framework to structure your tests. Second, it gives you the ability to run all your tests easily and get a report of what is running and what is not.

http://junit.sourceforge.net/doc/cookbook/cookbook.htm
http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm

Eclipse also as a plugin for coverage:
Now, you wrote a number of tests, you are running them regurlarly, but how do you know that you tested everything? This is where covereage comes in. Run your tests, with coverage, and you will see which lines are code have been tested and which ones have not been tested.

→ Project right-click → Coverage As → Java application or JUnit test
→ Get hightlighted sources and a report coverage.
→ In EclEmma report view, can delete the current session (with the cross button).

http://www.eclemma.org/
http://www.eclemma.org/installation.html

Note that Cobertura is also an interesting coverage framework:

http://sourceforge.net/projects/cobertura/?source=typ_redirect
http://cobertura.github.io/cobertura/

Miscellaneous:

Eclipse supports printing Java sources, with multiple source pages per printed page, that is cool.

Eclipse CDT (the C/C++ plugin) is nice.

If your Eclipse crashes and refuses to restart, complaining that your workspace is already locked... then you need to remove the lock file in the .metadata directory in your workspace:

```
$ cd workspace
$ rm .metadata/.lock
```

Last but not least:

Oracle JvisualVM is a tool that comes with the Oracle Java JDK. **It is an absolutely fantastic tool.**
Launch it like:

```
$ $(JDK_HOME)/bin/jvisualvm
```

It will launch a window, in which all running Java processes will show up. You can select any one of them and connect to it, seeing everything you wanted to know about the runtime characteristics of your Java program.

Use it and you will love it.