Eclipse IDE – Keyboard Shortcuts
CSC 440 – Team X

Version: 1.00
Last revision date:02/10/2011
Created on: 10/02/2011 16:30:19
File: CSC440_TeamX_Eclipse_Shortcut_Keys_02102011.doc

1.0 Revision History

Name	Date	Reason for Change	Revision
Jason Niklas	02/10/11	Original	1.00
Brian Santisi	02/11/11	Additional contributions	1.01

2.0 Introduction

This document provides a basic summary of some of the more commonly used Eclipse keyboard shortcuts.

2.1 Purpose

The main purpose of this document is to provide Eclipse developers an easy to understand reference manual which comprehensively covers commonly used keyboard shortcuts. The goal is that developers, through the use of this manual, will become more productive and produce more efficient code when creating Eclipse applications.

2.2 Intended Audience

This document is intended for:

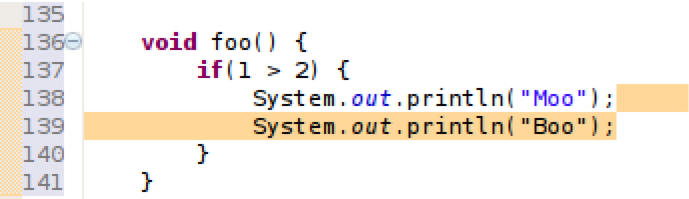
- Designers and Developers

CTRL + D

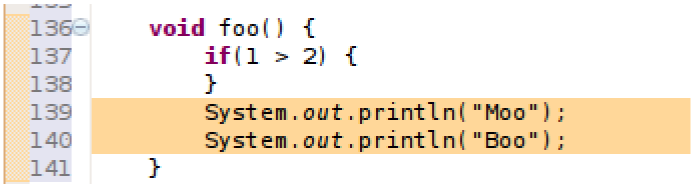
Delete row. Try it! You no more need to grab the mouse and select the line, no more Home, Shift + End, Delete. Quick and clean.

ALT + Up/Down Arrow

Move the row (or the entire selection) up or down. Very useful when rearranging code. You can even select more rows and move them all. Notice, that it will be always correctly indented.



```
    void foo() {
        if(1 > 2) {
            System.out.println("Moo");
            System.out.println("Boo");
        }
    }
```



```
    void foo() {
        if(1 > 2) {
        }
        System.out.println("Moo");
        System.out.println("Boo");
    }
```

ALT + Left/Right Arrow

Move to the last location you edited. Imagine you just created a class Foo, and now you are working on a class Boo. Now, if you need to look at the Foo class, just press Alt+Left Arrow. Alt+Right Arrow brings you back to Boo.

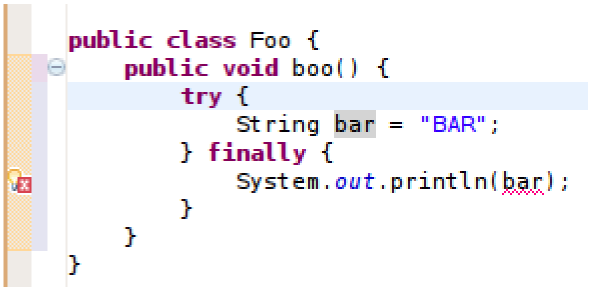
CTRL+SHIFT+O

Organize imports. What happens when you first use a class you have not yet imported? You will see an error. But when you press this magical combination, all your missing classes will be imported, and the unused imports will vanish.

CTRL+1

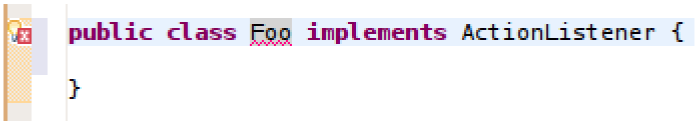
Probably the most useful one. It activates the quick fix. Imagine you create a class, which implements some interface. You will get an error, because the inherited methods are not yet implemented. While you are on line where the error occurs, press this combination to activate the quick fix. Now, select the "Add unimplemented methods" option. You can use the quick fix at every error you ever receive.

Quick fix comes handy in other situations too. My favorite is the "Split variable declaration". Sometimes I need to broaden the scope of a variable. I activate the quick fix, split declaration, and use alt + arrow to put it where it belongs. You can find even more usages: Convert local variable to field, rename in file, Inline local variable..



```
public class Foo {
    public void boo() {
        try {
            String bar = "BAR";
        } finally {
            System.out.println(bar);
        }
    }
}
```

You could use the "Split variable declaration" on the bar variable, and then move it with Alt+Arrows above the try block..



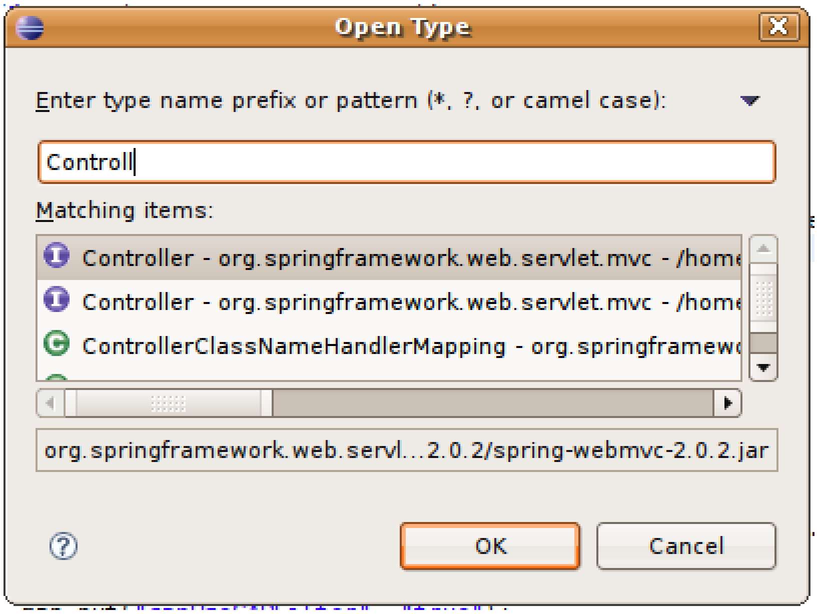
```
public class Foo implements ActionListener {

}
```

Or you could use the "Add unimplemented methods" fix here.

The best thing you can do if you see an error is to use the quick fix.

CTRL+SHIFT+T

Open Type. Imagine, that you need to have a look at the Foo class. But, where is the Foo class? Is it in the Boo project and in the foo.bar package? Or somewhere else? With this shortcut, you don't need to know. Just press it, type Foo and you are in.



Open Type

Enter type name prefix or pattern (*, ?, or camel case):

Controll

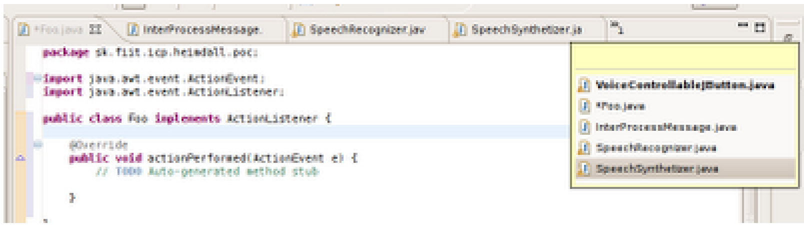
Matching items:

Controller - org.springframework.web.servlet.mvc - /home
Controller - org.springframework.web.servlet.mvc - /home
ControllerClassNameHandlerMapping - org.springframew

org.springframework.web.servl...2.0.2/spring-webmvc-2.0.2.jar

CTRL+E

Shows you a list of all open editors.



```
package sk.fiit.icp.heimdall.poc;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Foo implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub

    }
```

CTRL+F6

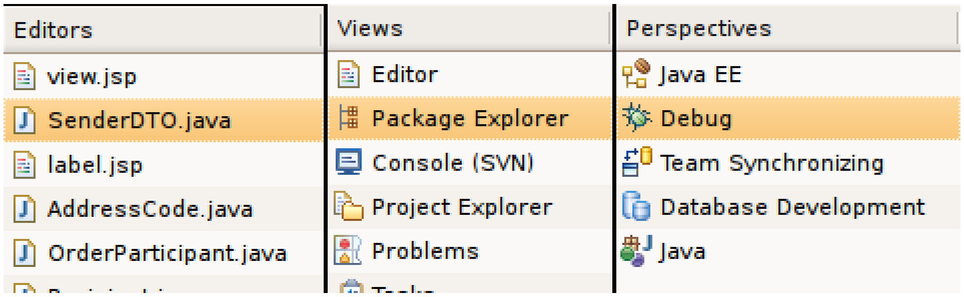
Use to move between open editors. This is an slower alternative to Ctrl + E. Comes handy in a situation when you want to periodically switch between two editors, something, what is nearly impossible with Ctrl+E as it sorts entries quite randomly. Or you might just use Alt+Arrows..

CTRL+F7

Move between views. When in editor, press Ctrl+F7 to switch to the Package Explorer, or hold Ctrl and press F7 multiple times to switch to other views.

CTRL+F8

Move between perspectives. The same as previous.



Editors	Views	Perspectives
view.jsp	Editor	Java EE
SenderDTO.java	Package Explorer	Debug
label.jsp	Console (SVN)	Team Synchronizing
AddressCode.java	Project Explorer	Database Development
OrderParticipant.java	Problems	Java

CTRL + F11

Runs the application. What gets launched depends on your settings. It will either launch the last launched class (my preffered way) or it will launch currently selected resource (the default way). If you want to change its behavior read the previous post.

CTL + N

Open new type wizard. This is not very quick because you have to select the wizard type (weather you want to create new class, jsp, xml or something else) in the next step. Much faster way would be if you could just hit the shortcut and invoke the particular wizard. It is possible, just keep reading..

CTRL + M

Maximize or umaximize current tab.

CTRL + I

Corrects indentation.

CTRL + SHIFT + F

Formats code. You can make a beautiful looking code out of a mess with this. It requires a bit of setup, but it is well worth it. You can find its settings under Window->Preferences->Java->Code style->Formatter

CTRL + J

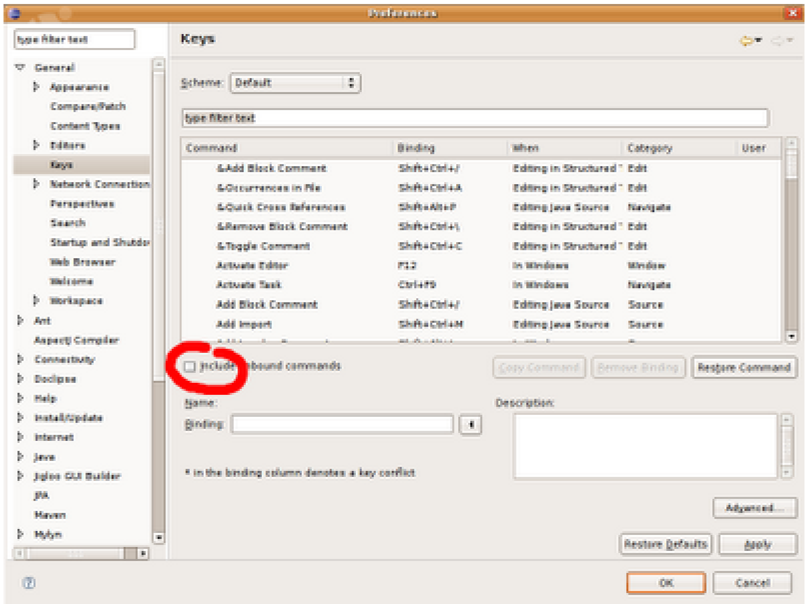
Incremental search. Similar to the search in firefox. It shows you results as you type. Don't be surprised, if you hit this combination, nothing happens - at the first glance. Just start typing and eclipse will move your cursor to the first ocurence.

CTRL + SHIFT + L

Shows you a list of your currently defined shortcut keys.

I don't like your shortcuts

Such is life nowadays. Remember, you can always change those bindings to match your preferences. Open *Window->Preferences->General->Keys*. Now you can use the filter to find your shortcut and change its binding.
The real fun begins when you cannot find the command you are looking for. The key here, is to have the "Include unbounds commands" checkbox checked. It will show you all commands, even those, which have no keys bound.



Scheme: Default

type filter text

Command	Binding	When	Category	User
&Add Block Comment	Shift+Ctrl+/	Editing in Structured"	Edit	
&Occurrences in File	Shift+Ctrl+A	Editing in Structured"	Edit	
&Quick Cross References	Shift+Alt+P	Editing Java Source	Navigate	
&Remove Block Comment	Shift+Ctrl+\	Editing in Structured"	Edit	
&Toggle Comment	Shift+Ctrl+C	Editing in Structured"	Edit	
Activate Editor	F12	In Windows	Window	
Activate Task	Ctrl+F9	In Windows	Navigate	
Add Block Comment	Shift+Ctrl+/	Editing Java Source	Source	
Add Import	Shift+Ctrl+M	Editing Java Source	Source	

☐ Include unbound commands

Name:

Binding:

Description:

* in the binding column denotes a key conflict

While you are here, I recommend to add the following bindings:

CTRL+SHIFT+G

Bind this to "Generate getters and setters". This is a "must have".

ALT+C

Bind this to SVN/CVS "Commit".

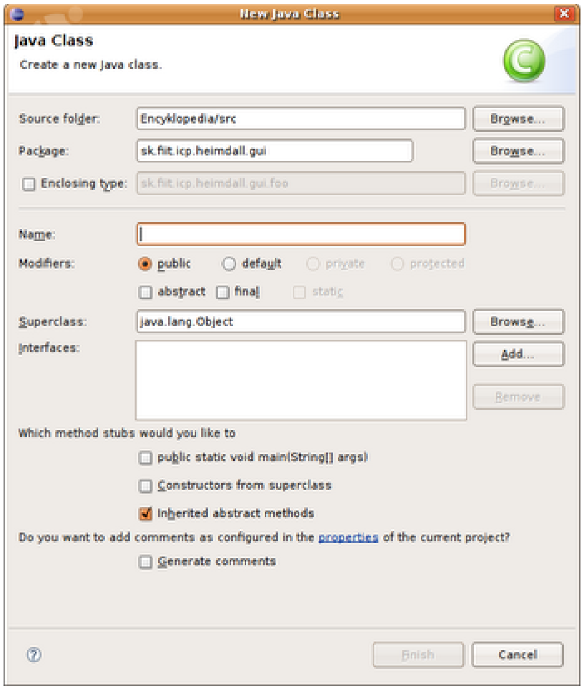
ALT+U

Bind this to SVN/CVS "Update".

Now, type "new" (without quotes) in the filter text. You should see a list of all new type wizards. Choose the most frequently used and assign them a shortcut. For example, the most used wizard for me is the new class wizard. Thus I assigned it the CTRL+SHIFT+N keys.

Let me demonstrate a quick way to create new class now.

Hit **CTRL + SHIFT + N** (or the combination you assigned in the previous step). This should bring up new class wizard. Type in the name and press **ALT+E**. You can now select a class which will be a superclass for the newly created class. Hit ALT+A and select all implemented interfaces . Now hit **ALT+F** and your class will be generated. Eclipse will also provide the default implementation for all abstract and interface methods you inherited.



Java Class

Create a new Java class.

Source folder: Encyklopedia/src

Package: sk.fiit.icp.heimdall.gui

□ Enclosing type: sk.fiit.icp.heimdall.gui.foo

Name:

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to

□ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments as configured in the properties of the current project?

□ Generate comments

CTRL+ALT+H Open Call Heirarchy. Find out where that method is used.

F4 Open Type Heirarchy. See the subclasses and/or superclasses. This one can be tricky sometimes. If there is a class under the cursor it will open that class's heirarchy, if not it will use the class that contains the cursor (including inner classes).

F3 Open Declaration. Jump to the declaration of the variable, method or class. It will try to find the most specific declaration it can for methods, but depending on your code you might end up in an Interface which is probably not what you want. Then just press **F4** to get the type heirarchy and find the implementation you are looking for.

ALT+SHIFT+R Refactor->Rename. Works in the editor and in the package explorer. If you haven't learned the power of refactoring, do so now. Think of this one as a super-smart Find-Replace. The new 3.3 feature of inline rename is awesome.

ALT+SHIFT+V Refactor->Move. Again, works in both the edtor and package explorer. Move the method or field to another class.

ALT+SHIFT+M Extract to method. Break up that monolithic code, takes the selection and trys to make it into a method. Will optionally find duplicate code blocks and use the new method there as well. It might take a few trys to get this to work the way you want. Just try it, rearrange thee code a little and try again.

Edit

Ctrl + Space	Content Assist (also Templates invoking)
Ctrl + Shift + Space	Context Information/Show Parameter
Ctrl + C	Copy the selection to the clipboard
Ctrl + X	Cut the selection to the clipboard
Ctrl + F	Find and Replace
Ctrl + K	Find next
Ctrl + Shift + K	Find previous
Ctrl + J	Incremental Find
Ctrl + Shift + J	Incremental Find reverse
Ctrl + V	Paste from the clipboard
Ctrl + 1	Quick Fix
Ctrl + Y	Redo the last operation
Alt + Shift + ↓	Restore Last Selection
Ctrl + A	Select All
Alt + Shift + ↑	Expand selection to include enclosing element
Alt + Shift + →	Expand selection to include next sibling
Alt + Shift + ←	Expand selection to include previous sibling
F2	Show tooltip/javadoc description for the element at the cursor
Ctrl + Z	Undo the last operation

File

Ctrl + F4	Close the active editor
Ctrl + Shift + F4	Close all editors
Ctrl + N	Open the New item wizard
Ctrl + P	Print
Alt + Enter	Display the properties of the selected item
Ctrl + S	Save the current contents
Ctrl + Shift + S	Save all current contents

Navigate

Alt + ←	Move backward in the editor navigation history
Alt + →	Move forward in the editor navigation history
Ctrl + Q	Go to last edit location
Ctrl + L	Go to Line
Ctrl + Shift + P	Moves the cursor to the matching bracket
Ctrl + Shift + ↓	Go to Next Member
Ctrl + Shift + ↑	Go to Previous Member
Ctrl + .	Navigate to the next item
Ctrl + Alt + H	Open Call Hierarchy
F3	Open Declaration (Open an editor on the selected element)
Shift + F2	Open External Javadoc
Alt + F3	Opens an implementation of a method call on an interface
Ctrl + Alt + F3	Opens an interface declaration of a method call
Ctrl + F3	Show the structure of the selected element
Ctrl + Shift + T	Open a type in a Java editor
F4	Open a type hierarchy on the selected element
Ctrl + Shift + H	Open a type in the type hierarchy view
Ctrl + ,	Navigate to the previous item
Ctrl + O	Show Outline

Project

Ctrl + B	Build all projects

Refactoring

Alt + Shift + L	Extracts an expression into a new local variable
Alt + Shift + M	Extracts a set of statements or an expression into a new method
Alt + Shit + I	Inline a constant, local variable or method
Alt + Shift + V	Move – Refactoring
Alt + Shit + Y	Redo – Refactoring
Alt + Shift + R	Rename – Refactoring
Alt + Shift + Z	Undo – Refactoring

Run/Debug

Ctrl + Shift + B	Add/Remove Breakpoint
F11	Debug Last Launched
Ctrl + D	Display result of evaluating selected text
Ctrl + U	Execute: evaluate selected text
F8	Resume
Ctrl + F11	Run Last Launched
Ctrl + R	Run to Line
F5	Step Into

Run/Debug

Ctrl + F5	Step Into Selection
F6	Step over
F7	Step Return
Shift + F5	Step with filters

Search

Ctrl + G	Declaration in Workspace
Ctrl + Shift + U	Occurrences in File
Ctrl + H	Open Search Dialog
Ctrl + Shift + G	References in Workspace

Source

Ctrl + Shift + M	Add Import (create on selection)
Ctrl + /	Comment
Ctrl + Shift + F	Format the selected text
Ctrl + Shift + O	Organize Imports
Ctrl + \	Uncomment

Window

F12	Activate Editor
Ctrl + F6	Next Editor
Ctrl + F8	Next Perspective
Ctrl + F7	Next View
Ctrl + Shift + F6	Previous Editor
Ctrl + Shift + F8	Previous Perspective
Ctrl + Shift + F7	Previous View
Ctrl + W	Show Ruler Context Menu
Alt + -	Show System Menu
Ctrl + F10	Show View Menu
Ctrl + Shift + W	Switch to Editor

Text Editing

Ctrl + E	Delete Line
Ctrl + Shift + E	Delete to End of Line
Ctrl + ↓	Scroll Line Down
Ctrl + ↑	Scroll Line Up
Insert	Toggle Overwrite