Software Developments Toolbox

9466A | 6:00-7:00 MTH

GROUP 1

Topic 1 - Advanced Code Editing/Formatting

Topic 2 - Code Refactoring

Topic 3 - Testing

Topic 4 - Debugging

Topic 5 - Miscellaneous (Documentation)

Submitted by:

Bernardo, Laurence M.

Fongkot, Reden F.

Mahmood, Imran D.

Panopio, Russel Jacob L.

Siababa, Carlos Joshua A.

Submitted to:

Mr. Roderick Makil

TOPIC 1 - ADVANCED CODE EDITING/FORMATTING

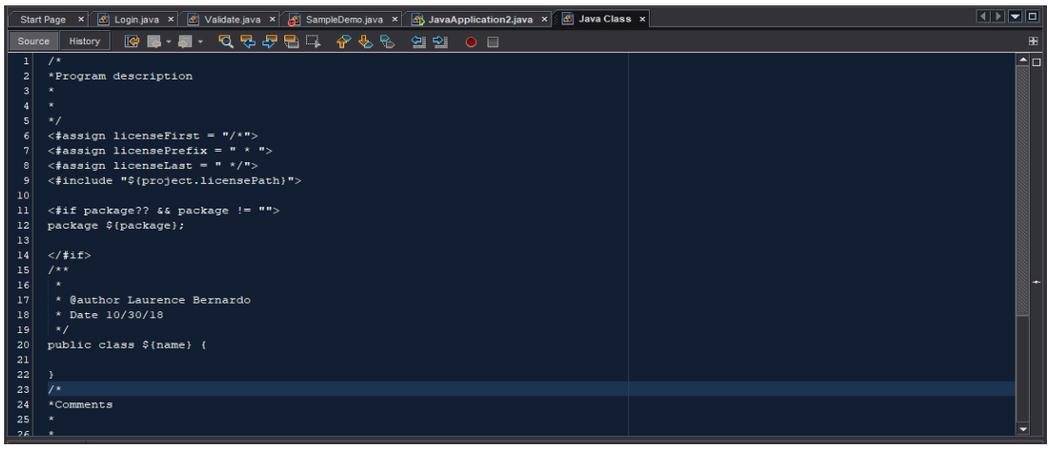
Advanced Code Editing

Helps the developers in advance coding by providing tools and techniques to make their life a lot easier.

Template changing using Java Class

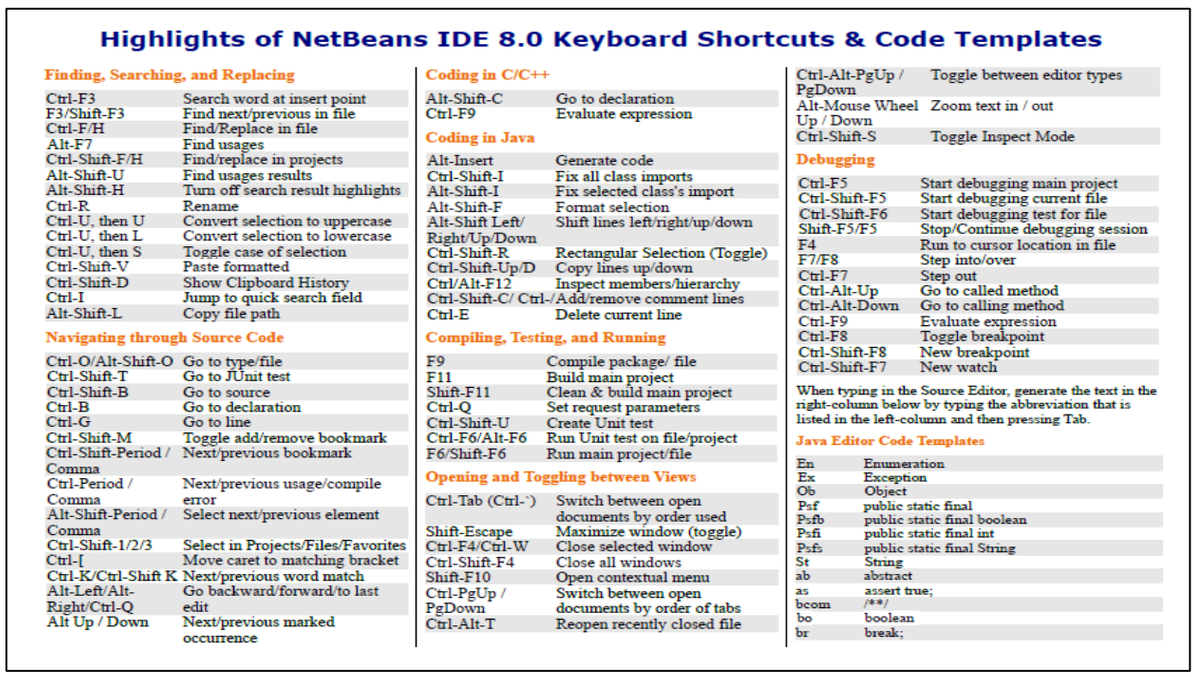
You can add you the name of the author of the program at the top also the date and some comments before the coding starts**.**

1. You go to the tools templates.
2. Find Java then choose Java Class and click Open editor.
3. Now you can insert the author of the program and also the description, date and comment of the instructor.
4. Don't forget to save, then open a new Java class.



Shortcuts Code from keyboard template.

There are a lot advantage using NetBeans IDE keyboards one of this is Keyboard Shortcuts.



Highlights of NetBeans IDE 8.0 Keyboard Shortcuts & Code Templates

Finding, Searching, and Replacing

Ctrl-F3	Search word at insert point
F3/Shift-F3	Find next/previous in file
Ctrl-F/H	Find/Replace in file
Alt-F7	Find usages
Ctrl-Shift-F/H	Find/replace in projects
Alt-Shift-U	Find usages results
Alt-Shift-H	Turn off search result highlights
Ctrl-R	Rename
Ctrl-U, then U	Convert selection to uppercase
Ctrl-U, then L	Convert selection to lowercase
Ctrl-U, then S	Toggle case of selection
Ctrl-Shift-V	Paste formatted
Ctrl-Shift-D	Show Clipboard History
Ctrl-I	Jump to quick search field
Alt-Shift-L	Copy file path

Navigating through Source Code

Ctrl-O/Alt-Shift-O	Go to type/file
Ctrl-Shift-T	Go to JUnit test
Ctrl-Shift-B	Go to source
Ctrl-B	Go to declaration
Ctrl-G	Go to line
Ctrl-Shift-M	Toggle add/remove bookmark
Ctrl-Shift-Period / Comma	Next/previous bookmark
Ctrl-Period / Comma	Next/previous usage/compile error
Alt-Shift-Period / Comma	Select next/previous element
Ctrl-Shift-1/2/3	Select in Projects/Files/Favorites
Ctrl-[	Move caret to matching bracket
Ctrl-K/Ctrl-Shift K	Next/previous word match
Alt-Left/Alt-Right/Ctrl-Q	Go backward/forward/to last edit
Alt Up / Down	Next/previous marked occurrence

Coding in C/C++

Alt-Shift-C	Go to declaration
Ctrl-F9	Evaluate expression

Coding in Java

Alt-Insert	Generate code
Ctrl-Shift-I	Fix all class imports
Alt-Shift-I	Fix selected class's import
Alt-Shift-F	Format selection
Alt-Shift Left/Right/Up/Down	Shift lines left/right/up/down
Ctrl-Shift-R	Rectangular Selection (Toggle)
Ctrl-Shift-Up/D	Copy lines up/down
Ctrl/Alt-F12	Inspect members/hierarchy
Ctrl-Shift-C/ Ctrl-/	Add/remove comment lines
Ctrl-E	Delete current line

Compiling, Testing, and Running

F9	Compile package/ file
F11	Build main project
Shift-F11	Clean & build main project
Ctrl-Q	Set request parameters
Ctrl-Shift-U	Create Unit test
Ctrl-F6/Alt-F6	Run Unit test on file/project
F6/Shift-F6	Run main project/file

Opening and Toggling between Views

Ctrl-Tab (Ctrl-`)	Switch between open documents by order used
Shift-Escape	Maximize window (toggle)
Ctrl-F4/Ctrl-W	Close selected window
Ctrl-Shift-F4	Close all windows
Shift-F10	Open contextual menu
Ctrl-PgUp / PgDown	Switch between open documents by order of tabs
Ctrl-Alt-T	Reopen recently closed file
Ctrl-Alt-PgUp / PgDown	Toggle between editor types
Alt-Mouse Wheel Up / Down	Zoom text in / out
Ctrl-Shift-S	Toggle Inspect Mode

Debugging

Ctrl-F5	Start debugging main project
Ctrl-Shift-F5	Start debugging current file
Ctrl-Shift-F6	Start debugging test for file
Shift-F5/F5	Stop/Continue debugging session
F4	Run to cursor location in file
F7/F8	Step into/over
Ctrl-F7	Step out
Ctrl-Alt-Up	Go to called method
Ctrl-Alt-Down	Go to calling method
Ctrl-F9	Evaluate expression
Ctrl-F8	Toggle breakpoint
Ctrl-Shift-F8	New breakpoint
Ctrl-Shift-F7	New watch

When typing in the Source Editor, generate the text in the right-column below by typing the abbreviation that is listed in the left-column and then pressing Tab.

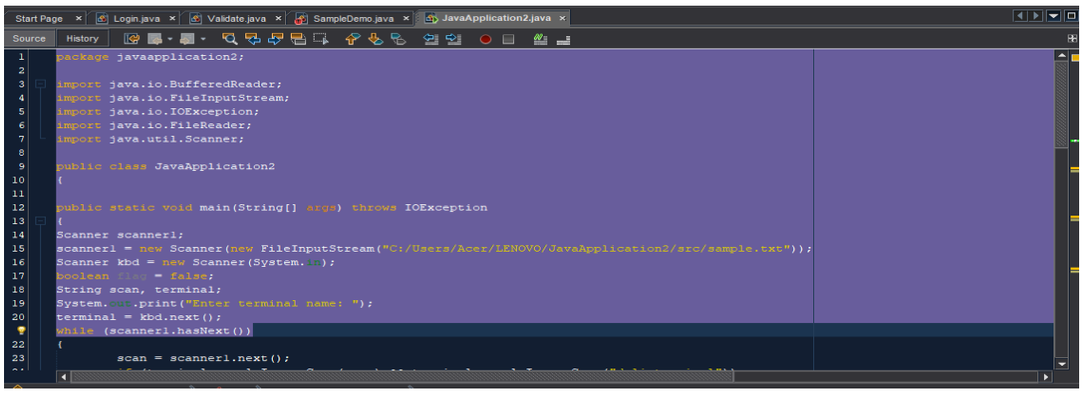
Java Editor Code Templates

En	Enumeration
Ex	Exception
Ob	Object
Psf	public static final
Psfb	public static final boolean
Psfi	public static final int
Psfs	public static final String
St	String
ab	abstract
as	assert true;
bcom	/**/
bo	boolean
br	break;

This Shortcut Keyboard is not available to other IDEs.

Formatting Code in NetBeans

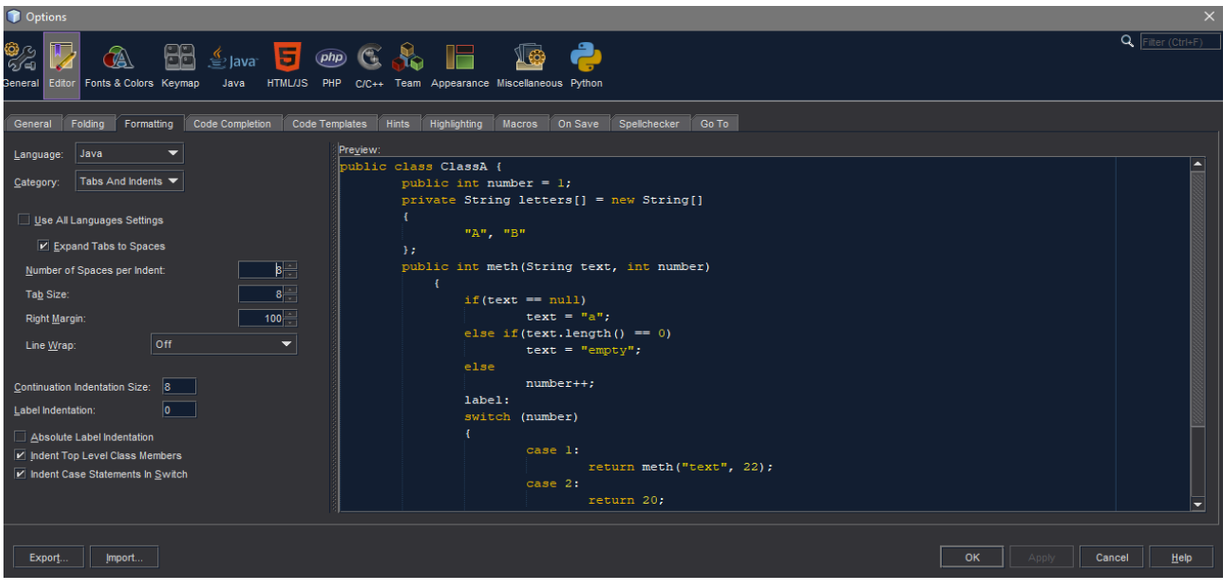
You can indent selected multiple lines of codes by pressing Tab or Alt+Shift+Right, and you can also reverse the indentation by pressing Shift+Tab or Alt+Shift+Left.
The editor toolbar has two buttons that can use for indenting and unindenting the selected multiple lines of codes.



Changing Format Rules

For the file types that you created, you can adjust the files in the formatting settings, such as placement of curly braces, number of spaces per tab, and so on. To adjust formatting rules for your Java files:

1. You go to the tools menu, find the options and click the editor button in the left panel.
2. Click editor button.
3. You can choose in the different categories.
4. For my demo I will choose Tabs and Indents.
5. Adjust the properties for the indention tab engine to your taste then click Ok to App.
6. You must reformat each file to the new rules by opening the file that you create and pressing Ctrl+Shift+F.



TOPIC 2 - CODE REFACTORING

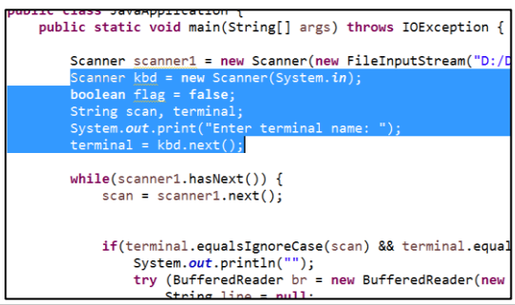
Refactoring is the process of altering the design and structure of a code, but without changing its functionalities. Factoring is done “little by little”. With every small step, excess and unneeded codes are removed or eliminated, and in return, readability is enhanced and memory demand is reduced. Enhanced readability does not only allow for a better understanding of the code, but results to easier modification as well.

Refactoring Using Eclipse IDE

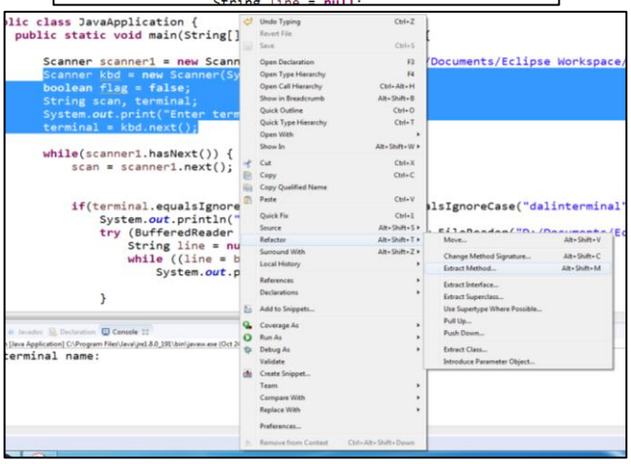
Refactoring: Extracting a Method

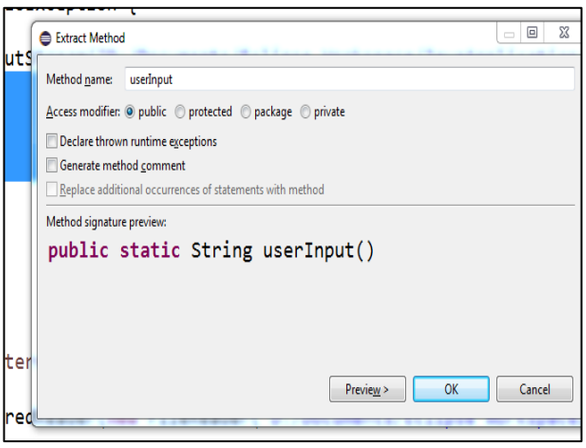
This refactoring technique transforms a block of code into a method. This will prevent code duplication, making your program shorter and allows better readability, understanding, and maintainability.

Highlight the block of code which you want to transform into a method. In this demonstration, the block of code that asks/takes input from the user is selected.



Right-click while the codes are highlighted to see available options. Go to “Refactor”, or simply do the keyboard shortcut *Alt+Shift+T*, and choose “Extract Method”. Alternatively, you can directly go to this refactoring technique by typing the keyboard shortcut *Alt+Shift+M*. You will then be prompted to type a name for the method.





Clicking on “OK” will now extract the block of code that was chosen into a method.

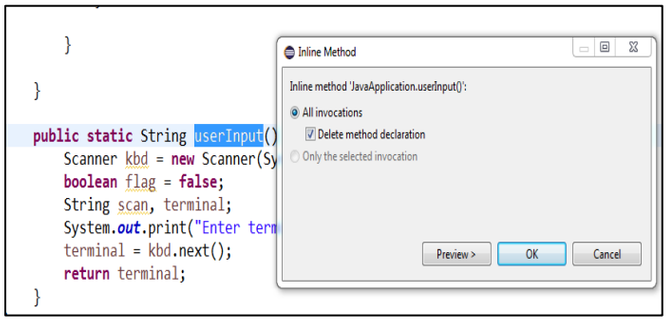
```
public static String userInput() {
    Scanner kbd = new Scanner(System.in);
    boolean flag = false;
    String scan, terminal;
    System.out.print("Enter terminal name: ");
    terminal = kbd.next();
    return terminal;
}
```

Refactoring: Inline Method

The opposite of extract method is inline method. You may want to use this technique if you have a method that is very short, or a method that is not really necessary.

Highlight the name of the method that you wish to refactor. Go to the refactoring options, then choose “Inline…”. In Eclipse IDE, the keyboard shortcut for inline method is *Alt+Shift+I*.

Using inline method, the userInput() method that was created using the extract method technique was more or less reverted back to its original form.




```
boolean flag;
String scan;
Scanner kbd = new Scanner(System.in);
boolean flag1 = false;
String scan1, terminal1;
System.out.print("Enter terminal name: ");
terminal1 = kbd.next();
String terminal = terminal1;
```

Refactoring: Renaming

Renaming is probably the most used refactoring technique. This technique can be very useful in a large program where variables and methods are used multiple times, and renaming them one by one can be tedious.

Highlight the name of the variable/method which you wish to rename. Right-click and choose “Rename…” in the refactoring options, or directly go to this refactoring technique using the keyboard shortcut *Alt+Shift+R*.

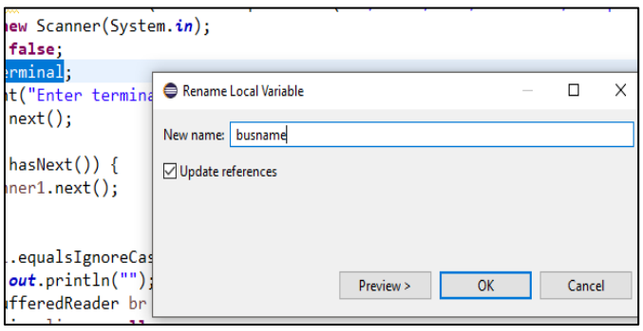
```
Scanner scanner1 = new Scanner(new FileInputStream("C:/Users/Acer/Documents/Eclipse Workspace/JavaApplication,
Scanner kbd = new Scanner(System.in);
boolean flag = false;
String scan, terminal;
System.out.print("Enter terminal name: ");
terminal = kbd.next();

while(scanner1.hasNext()) {
    scan = scanner1.next();


    if(terminal.equalsIgnoreCase(scan) && terminal.equalsIgnoreCase("dalinterminal") || terminal.equalsIgnore(
        System.out.println("");
        try (BufferedReader br = new BufferedReader(new FileReader("C:/Users/Acer/Documents/Eclipse Workspace,
            String line = null;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
                                }
        }

        flag = true;
        break;
    }
```

Choose a new name for the variable/method that you wish to rename. This will also automatically update all instances in which that certain variable/method has been called within your program.




```
Scanner scanner1 = new Scanner(new FileInputStream("C:/Users/Acer/Documents/Eclipse Workspace/JavaApplic
Scanner kbd = new Scanner(System.in);
boolean flag = false;
String scan, busname;
System.out.print("Enter terminal name: ");
busname = kbd.next();

while(scanner1.hasNext()) {
    scan = scanner1.next();


    if(busname.equalsIgnoreCase(scan) && busname.equalsIgnoreCase("dalinterminal") || busname.equalsIgnc
        System.out.println("");
        try (BufferedReader br = new BufferedReader(new FileReader("C:/Users/Acer/Documents/Eclipse Work
            String line = null;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
                                }
        }
```

TOPIC 3 - TESTING

Testing on Netbeans using JUNIT

JUNIT is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

Click the 'Tools' on the menu bar and Select the "Create/Update Tests" (for netbeans IDE 8.0).

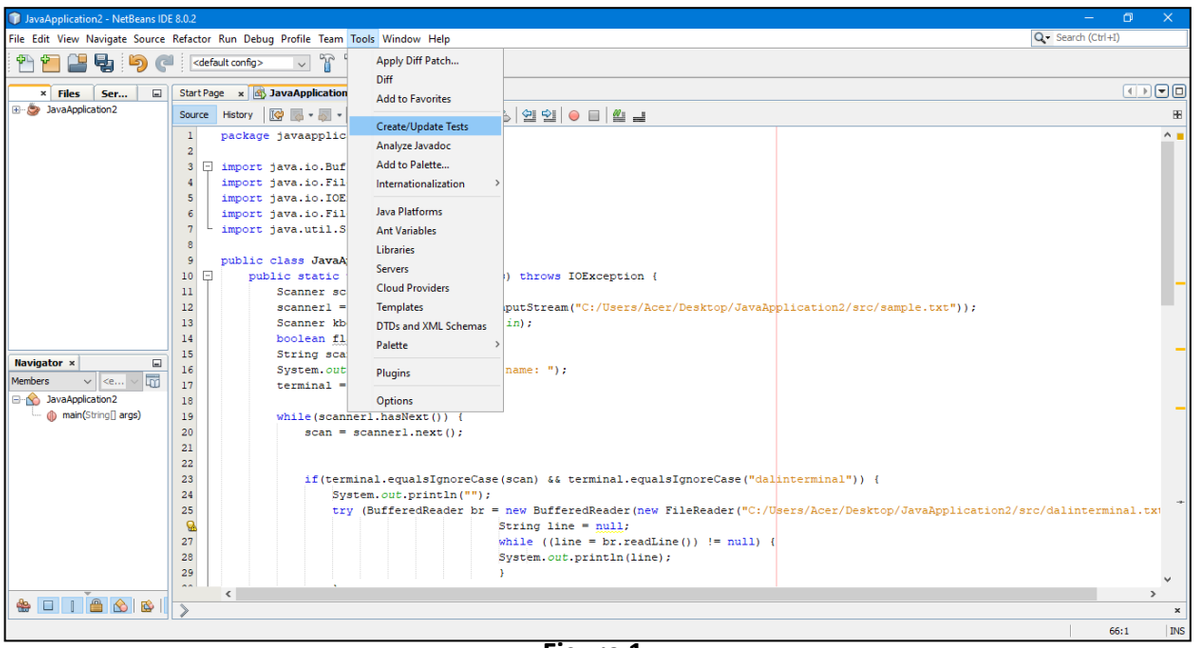


Figure 1.

Click the 'Create\Update Tests'.

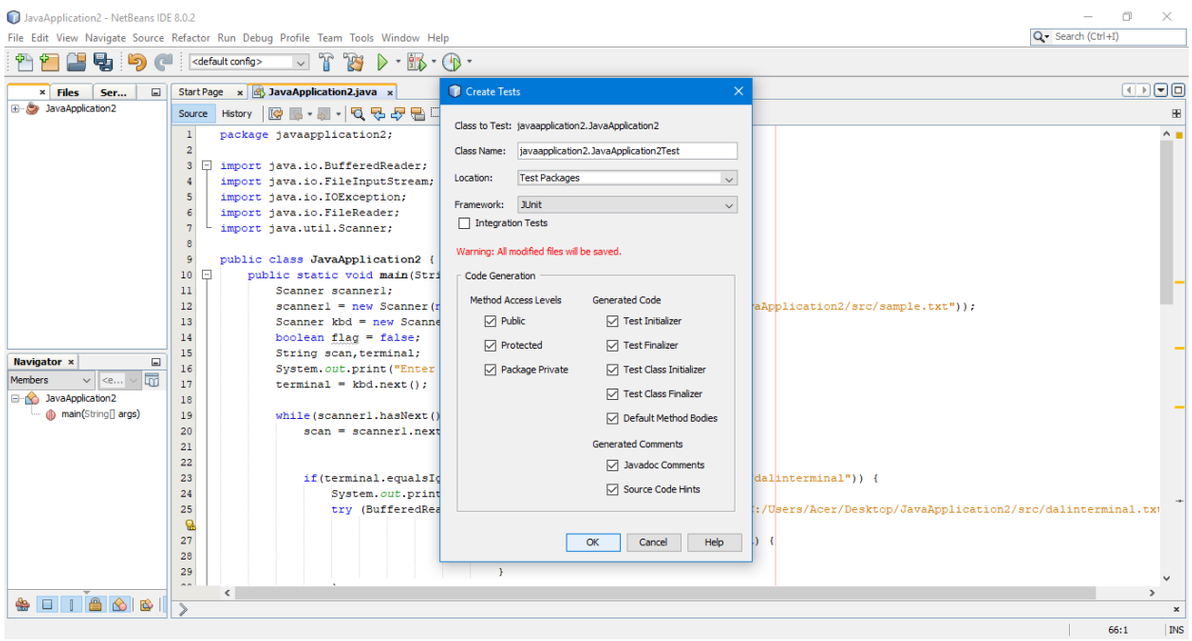


Figure 2.

Click the 'OK', and by clicking the 'OK' it will make a separate class based on name of your class was.

The output of the Test that has been made.

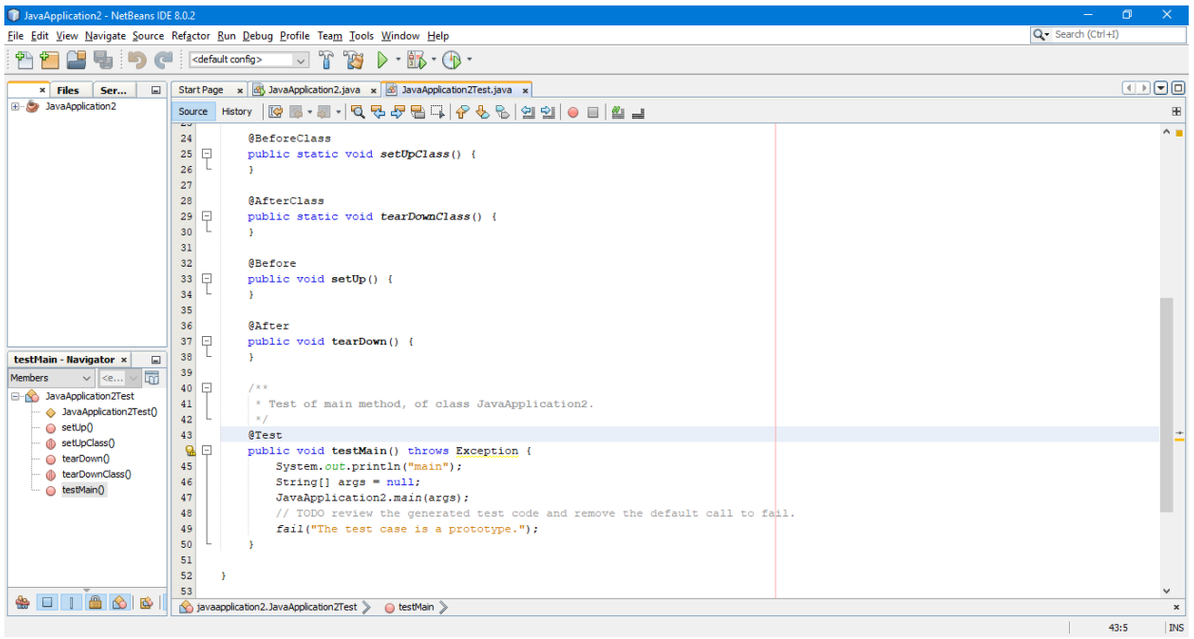


Figure 3.

For every method that was in the previous class will generate a test function.
Then select ‘run file’ option on the ‘Run’ menu bar to see if the test has a failures.

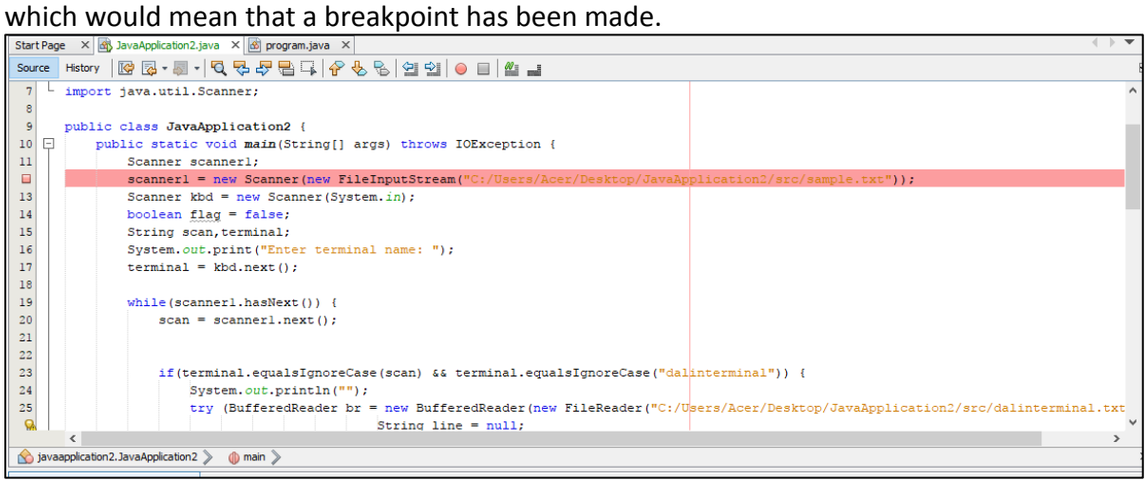
TOPIC 4 - DEBUGGING

Debugging using NetBeans

Programmers use debugging tools like NetBeans to help identify problems in a program and to easily fix the programs' issue.

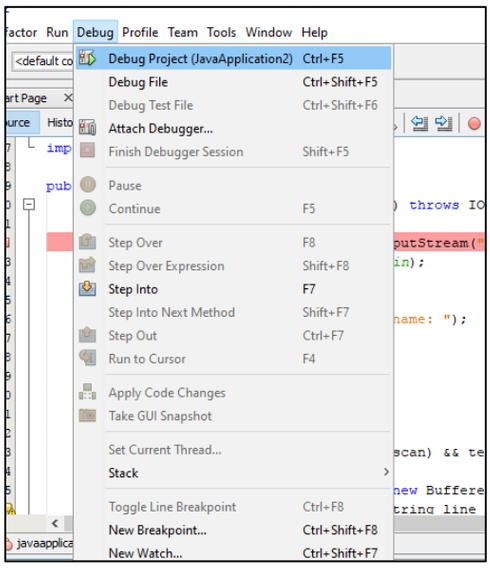
Breakpoints

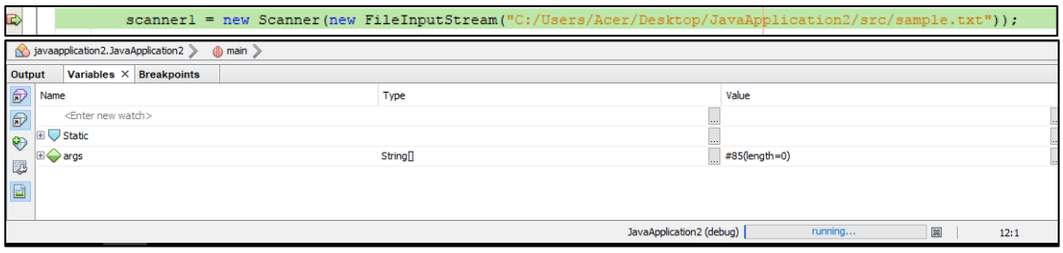
• A point where the code will stop temporarily for user to analyze the code up till that point.
• To make a break point, click on the left side of the line. Upon doing so, a pink box will appear which would mean that a breakpoint has been made.



Start debugging session

• Click on the menu tab and then choose Debug and click on Debug file
• Once clicked on debug button its color will turn to green
• At the lower part a debugger console will appear where it will show the line breakpoint, the variables declared and other information to be used during the debugging session



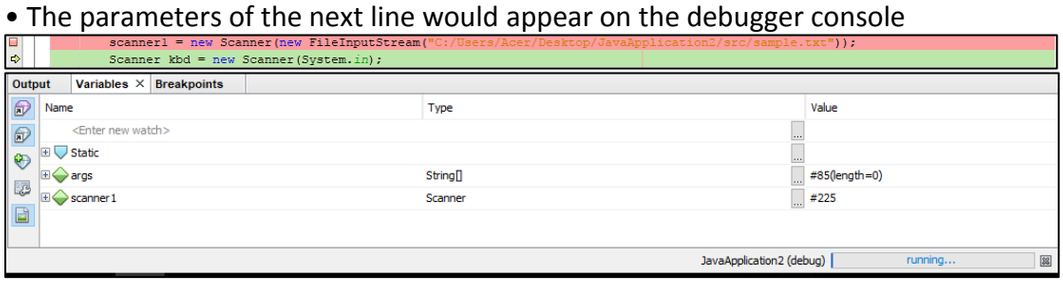


Step over(F8)

• Step over will view the input parameters and the resulting parameters then proceed to the next method

• Clicking on step over would change the color of the breakpoint and the next line would turn green

• The parameters of the next line would appear on the debugger console



TOPIC 5 - JAVA DOCUMENTATION

- Documentation is where we put every detailed functional information for the read to learn and guidelines for the user. In programming we used documentation to document our program for easier to understand and for future purposes. Documentation is also written to support any activity. Javadoc is a tool that NetBeans uses for documenting java. Javadoc documentations are being generated to HTML form. Under Javadoc we have a “doc comment”, the doc comment is where we place the description, function, explanation for the block or lines of codes.
- Documentation generator made by Sun Microsystems for the Java dialect (now claimed by Oracle Corporation) for creating API documentation in HTML arrange from Java source code. Javadoc does not influence execution in Java as all remarks are evacuated at accumulation time.

JAVA Documentation Using NetBeans IDE

Sample of doc comment

JavadocComment

- ***it is the place we locate the general portrayal, label marks with information.***

```
/*
*
*@Author
*/
```

Block Comment

- ***this remark we can discover the depiction for the class and others.***

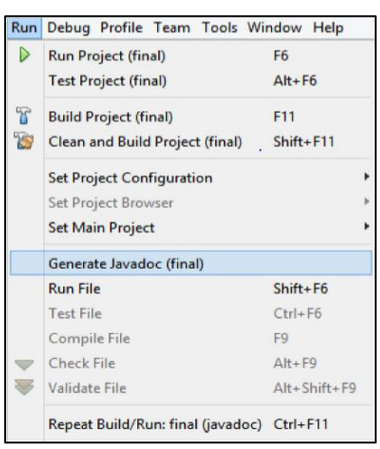
```
/*
* To change this license header, choose License Headers in Project Properti
* To change this template file, choose tools | Templates
*and open the template in the editor
*/
```

Inline comment- ignores everything to the end of the line

```
// TODO code application logic here
//
//
```

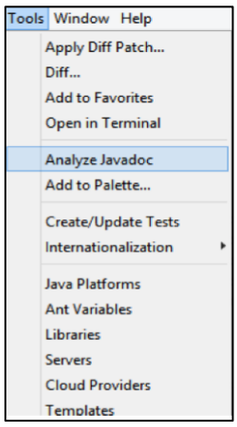
Generate Javadoc documentation-

- creates a Javadoc to the Javadoc organizer in your catalog at that point opens the documentation in the assigned program.

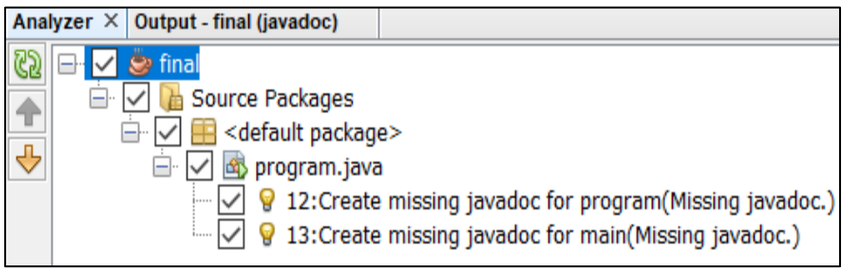


Auto generate comments in the file

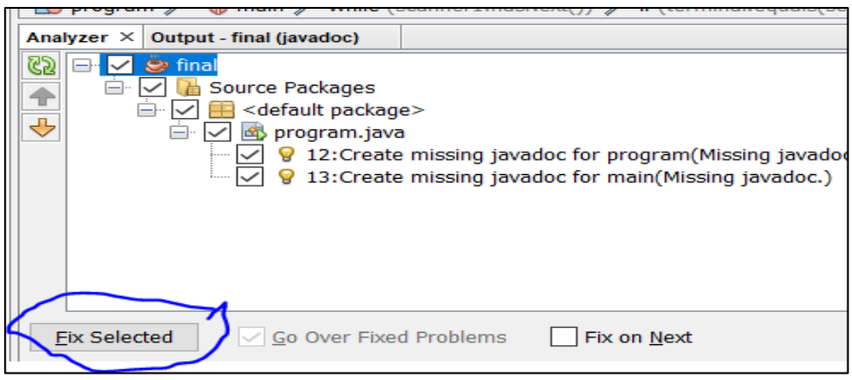
Step 1: Click on tools in the navbar and select “Analyze Javadoc”

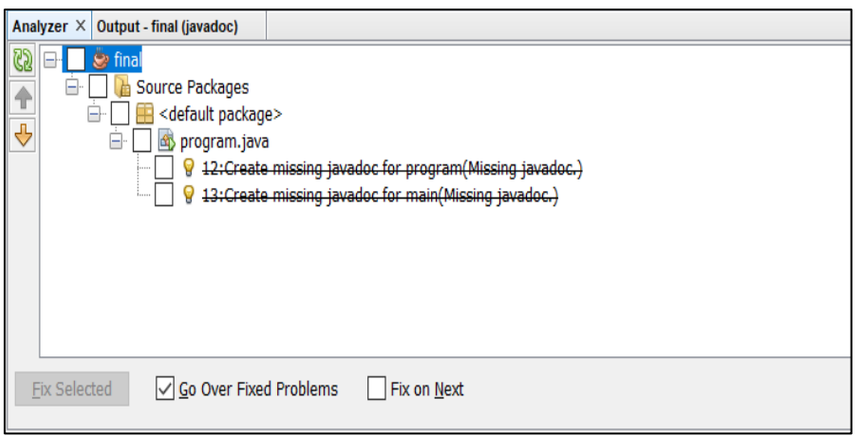


Step 2: Check all the files



Step 3: Click on Fix selected to apply the changes





References:

Shiholude. (2010, Jun 28). *Changing the template for java class (CTU Standards)*. Retrieved from https://www.youtube.com/watch?v=IZASEKx_20Q

[Telusko]. (2015, Feb 24). *Netbeans IDE KeyBoard Shortcuts Code Template for Java*[Video File]. Retrieved from https://youtu.be/R038Q1asI80

What Is Refactoring. (n.d.). Retrieved from http://wiki.c2.com/?WhatIsRefactoring&fbclid=IwAR1jMzQkiHS41hHafJLthmSnKHVX6IM2WqD0eiWhKaaZtdN1-1BGrteTqs8

Refactoring Inline Method. (n.d.). Retrieved from https://refactoring.guru/inline-method?fbclid=IwAR3Az7pnHiuB0g0zYPBAI6Bw6La7K-dhg7-0LzswYHN2Sd8_rlp40k_ECzQ

Design Patterns and Refactoring. (n.d.). Retrieved from https://sourcemaking.com/refactoring/rename-method?fbclid=IwAR1iaFNHEhPAv1mHH-YsXg_CRfaXsDJn1j2BINK3ID-zwx_TBLaIdpir-jc

[in28minutes]. (2012, Oct 11). *Eclipse Java Tutorial 11 - Refactoring* [Video File]. Retrived from https://youtu.be/sPq5fKweIhY

[luv2code]. (2014, Aug 12). *Java Eclipse Tutorial - Part 6.1: Refactoring Code (Extract Constants and Variables)* [Video File]. Retrieved from https://youtu.be/OrsdvCndArY

[luv2code]. (2014, Aug 12). *Java Eclipse Tutorial - Part 6.2: Refactoring Code (Extract Methods, Rename Methods and Variables)* [Video File]. Retrieved from https://youtu.be/JZn4hlMvP60

Junit Tutorial. (n.d). Retrieved from https://www.tutorialspoint.com/junit/

Debugging in Netbeans. (n.d.). Retrieved from http://www.itk.ilstu.edu/faculty/bllim/itk168/Labs/NetBeans_DebuggingW4/NetBeans_DebuggingW4_New.htm

Bouras, A. (n.d.). *Debugging Java Programs with NetBeans IDE*. Retrieved from https://www.bouraspage.com/repository/articles-java/debugging-java-programs-with-netbeans-ide

Javadoc Comment. (n.d.). Retrieved from https://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-files-in-eclipse

Java Documentation.(n.d.). Retrieved from https://www.tutorialspoint.com/java/java_documentation.htm

How to generate Javadoc HTML files. (n.d.). Retrieved from https://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-files-in-eclipse