

Reading:

Chapter 1.4 - 1.6 pages 26-49

Terminology:

Object-oriented programming	Machine language	IDE
Java API (standard class library)	Assembly language	Debugger
Documentation	High-level languages	SDK / JDK
Inline documentation (comments)	Fourth-generation languages	Syntax
Class	Editor	Semantics
Method	Compiler	Compile-time error
Identifiers	Interpreter	Run-time error
Reserved words	Source code	Logic error
White space	Java bytecode	

A **coding standard** consists of guidelines that are used by developers when writing source code to ensure that it follows a consistent format. Rules will be added to the COMP 1210 coding standard each week as more complex programs are developed.

COMP 1210 Coding Standard:

- The COMP 1210 coding standard is supported by Checkstyle, which is a tool that works with jGRASP to automatically detect style errors in a program. We will talk about Checkstyle in more details later.
- Lines of code must be less than 80 characters (including indentation). You can continue string literals on a new line using string concatenation. For example:

```
System.out.println("This is a long output string.");
```

Can be rewritten as:

```
System.out.println("This is a long "  
    + "output string.");
```

Note that the concatenation does not change the output of the program.

- All classes and methods require descriptive Javadoc comments. See pages 1-4 of the documentation guidelines for more information.

Due:

Part A: (50%) Monday, August 20, 2012 by the end of lab

Part B: (50%) Wednesday, August 22, 2012 by 11:59 PM

Goals:

By the end of this activity you should be able to do the following:

- Understand what happens when a program is compiled and run
- Add Javadoc comments to your program
- Generate documentation for your program
- Correct your source code structure using Checkstyle
- Submit your completed program to Web-CAT

Directions:**Part A:**

The two programs that you will need on your computer to compile and run a Java program are `java.exe` and `javac.exe`. In this activity Part A you will first create a program without the use of an IDE that will print out a welcome message to standard output. You will then create a similar program in jGRASP.

1. Using `javac.exe` and `java.exe` with no IDE (30%)

- Go to your H drive by selecting Start -> My Computer and then selecting the H: drive (H:). Create a new folder called COMP1210. This is where you will store all of your work for COMP 1210 for the rest of the semester. In the COMP1210 folder, create a folder called Lab1.

NOTE: it is important to save files to the H: drive so that they will be accessible from any computer that you use on campus rather than just on the local machine.

- Open Notepad and add the following text to create a Java class called `Welcome` and add a main method to your file as follows:

```
public class welcome
{
    public static void main(String[] args)
    {
    }
}
```

- Finally, add code that will print a welcome message to standard output:

```
public class welcome
{
    public static void main(String[] args)
    {
        System.out.println("welcome to COMP 1210!");
    }
}
```

- Now you will need to save your file as Java source code by adding the .java extension (instead of Notepad's default .txt extension). Select File -> Save As... and type the filename Welcome.java. Next to "Save as type", select "All Files" so that Notepad does not add the .txt extension to your file.

- Open a command prompt by clicking Start -> All Programs -> Accessories -> Command Prompt. Using the command prompt is very similar to navigating through folders using icons, but everything is done with text commands. The following are basic commands for the command prompt:

Command	Function
dir	"Directory" – displays the contents of the current directory
cd <i>foldername</i>	"Change directory" – Navigates to a folder called foldername
cd..	Goes back one folder
cls	"Clear screen" – Clears the command prompt screen
H:	Moves to the H: drive (or any other specified drive)

- First, use the H: command to navigate to the H drive. Now type dir to view all files and folders. You should see the comp1210 folder; use the cd command to navigate to the COMP1210 folder. Now use the cd command to navigate to the Lab1 folder. Perform the dir command once again and make sure that the file Welcome.java is present. **DO NOT CLOSE THE COMMAND PROMPT.**
- You will now need to compile your program using the javac.exe program. The command prompt does not know where this file is located, so you will have to find it on your computer.
 - Open a folder on your computer and then navigate to C:\Program Files\Java. Open the folder that starts with jdk, open the bin folder, and make sure that javac is present. Copy the link of the bin folder by clicking the address bar and then pressing Ctrl-C.
 - In the command prompt, type path= (include the opening double quotes). Now copy in the link to the bin folder by right clicking the command prompt and selecting paste. Type a double quote " to end the command and then press enter.
 - Type javac into the command prompt. If you get information concerning the javac program, then you have done the previous steps correctly. If not, then ask your lab instructor for help. Again, do not close the command prompt.
- Now you will need to compile your program. Make sure that you are in the correct folder by using the dir command and finding the Welcome.java file. Use the cls command to clear the screen.
- Run the javac program on your source code by typing the following command:

```
javac Welcome.java
```

If you do not see any errors, then your program has been compiled successfully. Use the `dir` command to determine whether the `Welcome.class` file is present. This is the **Java bytecode** file that is created when your program is compiled.

- Now, use the `java.exe` program to run the program that you created. In order to run your program, type the following command into the command prompt:

```
java Welcome
```

You should see your welcome message printed to the command prompt.


- **Keep the command prompt open. At the end of lab, you will need to show your lab instructor that your program was compiled and run in the command prompt.**

2. Using jGRASP (20%)

- Open jGRASP and then open a new Java window (File > New > Java). Enter the following Java statements to create a class called `StudentInfo`:

```
public class StudentInfo
{
}

```




- Save your file (`StudentInfo.java`) in the Lab1 folder.
- Generate your CSD by either pressing F2 or the  button in jGRASP. You should do this each time you write a line of code ensure that it is in the proper format. Alternatively, you can go to View then check the Auto Generate CSD box, and the CSD will be generated each time you Load, Save or Compile a file.
- Add a main method to the class (don't forget to re-generate the CSD):

```
public class StudentInfo
{
    public static void _____(String[] args)
    {
    }
}

```


- Create two `println` statements that print your name (first and last) and class (freshman, sophomore, junior, or senior) to the screen:

```
public class StudentInfo
{
    public static void _____(String[] args)
    {
        System.out.println(_____);
        System.out.println(_____);
    }
}
```

- Save your program.
- Click the  button on the toolbar. This opens the jGRASP Browse tab on the directory of the file in CSD in focus. You should see StudentInfo.java underlined in the list.
- Outside of jGRASP, if you open your COMP1210 and Lab1 folder, you should see this same list of files.
- Return to jGRASP, click the compile  button, and fix any compile-time errors. Return to the Lab1 folder outside of jGRASP and notice that there is a new file with the same name as your program but a different extension called Welcome.class. This is your compiled program and will be used to run your program in the next step.
- Return to jGRASP and click the run  button. Ensure that your output is correct.
- **At the end of lab, you will need to demonstrate compiling and running your completed program in jGRASP to receive credit for this portion of the activity. You will also need to show the program from part 1A being compiled and executed in the command prompt.**

Part B: (50%)

1. Checkstyle

- Download the Part B. zip file from the class website and extract the CourseInfo.java and the WelcomeCheckstyle.java program.
- Open WelcomeCheckstyle.java. Add Javadoc comments to the WelcomeCheckstyle program. These comments should describe what the program does. And use Checkstyle to assess the layout of your code. Click the Checkstyle  button in jGRASP and correct any issues identified by Checkstyle. If you are working at home, you will need to download and install Checkstyle. You may also need to configure Checkstyle in jGRASP (Tools > Checkstyle > Configure) so that jGRASP can find the folder where you installed Checkstyle.
- Open CourseInfo.java. You will be responsible for correcting style errors and logic errors that are present in the program. The first step is to modify the program to adhere to the COMP 1210 coding standard. Run Checkstyle and correct all of the formatting issues that appear.

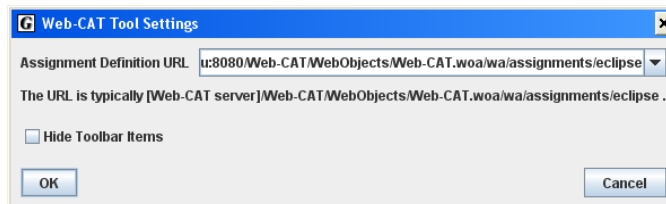
- HINT: Checkstyle states that the main method is missing a Javadoc comment, and it is correct. Make sure that you know the difference between a `//` single line comment, a `/*` multiple line comment, and a `/**` Javadoc comment.
- The program's **expected output** (the correct output) is as follows, with line numbers written on the left and expected output on the right:



1	Course Name: COMP 1210
2	Semester: Fall 2012
3	Instructor: Dr. Cross
4	
5	Description:
6	COMP 1210 uses the Java programming language to cover the basics of software development.

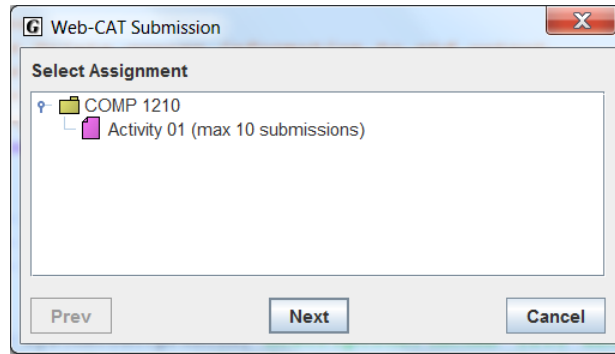
The original author of the program claims the program is correct because it "gets the point across," but you know that she is incorrect because the **actual output** of the program does not match the **expected output** above during testing. Correct the output errors in the program.
Hint: Look for issues in formatting/spacing, spelling, capitalization, etc.

2. Web-CAT

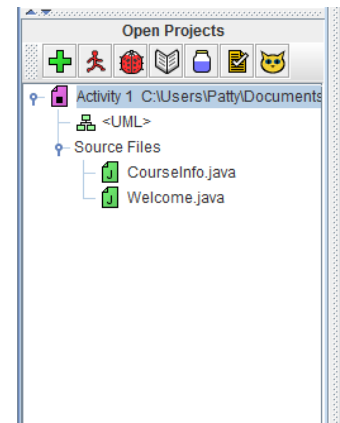
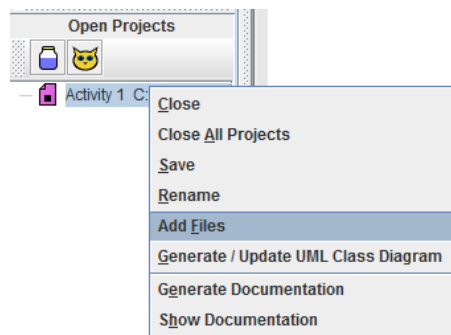
- In jGRASP, navigate to Tools > Web-Cat > Configure. In the dialog box enter <http://webcat.eng.auburn.edu:8080/Web-CAT/WebObjects/Web-CAT.woa/wa/assignments/eclipse> as the Assignment Definition URL (select the URL above and press Ctrl-C to copy, and then go to Assignment Definition URL and press Ctrl-V to paste). See the Web-CAT documentation on the class website for more information.



- Click OK. You should now have a Web-CAT  button on the toolbar.
- Open the Java file that you want to submit then you can submit your program via Web-CAT. Click the Web-CAT toolbar  button (or go to Tools → Web-Cat → Submit File). For example, select the COMP 1210 folder and the assignment labeled **Activity 01** and click Next; click Add to add the second file, click Next, login to Web-CAT, and click Submit.



- You will need to submit both `WelcomeCheckstyle.java` and the corrected version of `CourseInfo.java` to the Activity 01 in Web-CAT. You will only have 10 tries, so make sure that you submit both programs and they both have passed Checkstyle before submission.
- Your assignment Part B will be graded as follows. Web-CAT will automatically assess 50% of your score.
 - Program correctness: 90%
 - Checkstyle: 10%
- In order to submit both files, you can press the Web-CAT 🐱 button on one file and then add the other prior to submission (as described above) OR you can create a project, add both Java files to the project, and then submit the project. This will simplify the submission process and save time.
 - To create a project in jGRASP, go to **Project > New** and make sure that you are in the same directory as your source code files. Under Project Name, enter **Activity 1** and click Next. Click Next again to create the project.
 - On the left-hand side of jGRASP, right-click the project name and select Add Files. Add your two source code files to the project.



- Once both files are added, click the Web-CAT symbol on the project tab.
- Select the **Activity 01** assignment, click Next, make sure that both files are included, click Next, login to Web-CAT, and click Submit.

