

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 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 detail later.
- Lines of code must be less than 80 characters (including indentation). You can continue string literals on a new line using string concatenation (make a string from two or more other strings). 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.

- For project assignments, 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 22, 2016 by the end of lab

Part B: (50%) Friday, August 26, 2016 by 11:59 PM

(You should try to complete Part B in lab on Wed; otherwise during the Help session on Fri.)

Goals



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

- Create a program then compile and run it
- 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



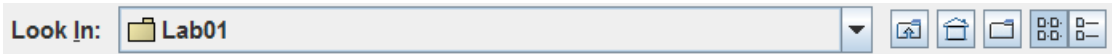








Part A: (50%) – Due Monday, August 22, 2016 by the end of lab



Using jGRASP (50%)

- If you are working on a lab computer, go to your H drive by clicking the **Start**  button in the lower left corner of the screen and click **File Browser** to open a file browser. Then select and open (double-click) the H: drive (H:) and create a new folder called COMP1210 (right-click and select **New > Folder**). This is where you will store all of your work for COMP 1210 for the rest of the semester. Open the COMP1210 folder and create a folder called Lab01. *Files on the H: drive will be accessible from any computer in an Engineering lab.* **If you are using your own machine, you should create the same folders described above in an appropriate location (e.g., C:\...\Documents\COMP1210\Lab01).**
- **Open jGRASP** – on lab computers, click the Windows **Start**  button in the lower left corner of the screen, then in the Search window, enter **jgrasp** then click **jGRASP** in the list.
- Using the top menu, open a new Java window (**File > New > Java**). Enter the following Java statements to create a class called StudentInfo:


```
public class StudentInfo
{
}

```






- Click the Save  button on the top menu, then in the Save As dialog, use the buttons to the far right of “Look In” near the top of the dialog to navigate up  to the folder where you 
Look In:  **Lab01**      
- want to save your program. If you are using a lab machine, you should select the H: drive. If you have not yet created a COMP1210 folder (and within it a Lab01 folder) you can do that now by clicking the folder  button to create a new folder. Click once on the name “New folder” then click on it again and change the name to COMP1210. Double-click the COMP1210 folder to open it, then (as above) create a folder named Lab01 and open it. If your program code is correct, you should see StudentInfo.java as the file name. If this is not the case, enter the file name manually. Now click the Save button at the bottom of the dialog.

- Click the Compile button , and fix any compile-time errors.
- Generate your CSD by either pressing F2 or by clicking the Generate CSD button  on the toolbar at the top of the jGRASP desktop. Now turn on Auto Generate CSD (View > then check the **Auto Generate CSD** box), so that the CSD will be generated each time you Load or Compile a file.
- Add a main method to the class. Be sure to replace the blank in the code below with the method name (don't forget to re-generate the CSD):

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


- Click the Compile button , and fix any compile-time errors.
- Create two statements that print your name (first and last) on the first line and then class (freshman, sophomore, junior, or senior) on the second line. Don't forget to put double quotes around the String literals in the println statements.

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

- Click the Compile button , and fix any compile-time errors. Note that if Auto Save is on, which is the default, your program is saved each time you compile.
- Toggle line numbers on/off by clicking  on the toolbar. Most users leave line numbers on.
- Click the Browse button  on the toolbar. This opens the jGRASP Browse tab on the folder that contains the file in the CSD window. You should see StudentInfo.java underlined in the Browse tab.
- Outside of jGRASP (e.g., in a Windows file browser), if you open your COMP1210\Lab01 folder, you should see this same set of files.
- Return to jGRASP, click the Compile button , and fix any compile-time errors. Once your program compiles successfully, you should see the corresponding .class file (StudentInfo.class) in the Browse tab. This file contains the bytecode for your program that will be used to run your program in the next step.
- In jGRASP and click the Run button . Ensure that your output is correct.
- **Before 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.**

Part B: (50%) – Due Friday, August 26, 2016 by 11:59 PM (You should try to complete Part B in lab on Wednesday. Otherwise, you should plan to attend the Help session on Friday for assistance, or you should make arrangements with your lab instructor for help with this part of the assignment.)

1. Checkstyle

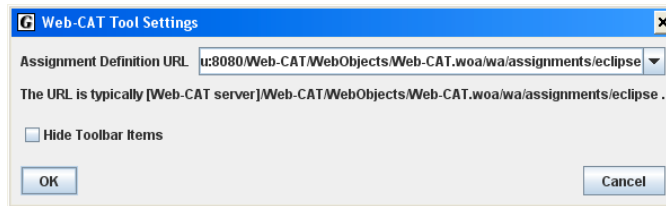
- Download the Part B zip file from the class website and save it in your Lab01 folder. Then extract the CourseInfo.java and the WelcomeCheckstyle.java files (Windows: right-click on the zip file then select **Extract All ...**; Mac OS X: just open the zip file).
- Open WelcomeCheckstyle.java in jGRASP. 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  on the jGRASP toolbar and correct any issues identified by Checkstyle. If you are working at home, you will need to download and install (unzip) Checkstyle. You may also need to configure Checkstyle in jGRASP (**Tools > Checkstyle > Configure**) so that jGRASP can find the folder containing the Checkstyle JAR file.
- Open CourseInfo.java by double-clicking on the file in the Browse tab. 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 this is true. Hint: Make sure that you know the difference between a // single line comment, a /* multiple line comment */, and a /** Javadoc comment */.
 - HINT: For dealing with a source line over 80 characters, see page 1 of this activity.
- The program's **expected output** is shown below (the line numbers on the left are **not** part of the expected output):

1	Course Name: COMP 1210
2	Semester: Fall 2016
3	Instructor: Dr. Cross
4	
5	Description:
6	COMP 1210 uses the Java programming language to cover the basics of software development.

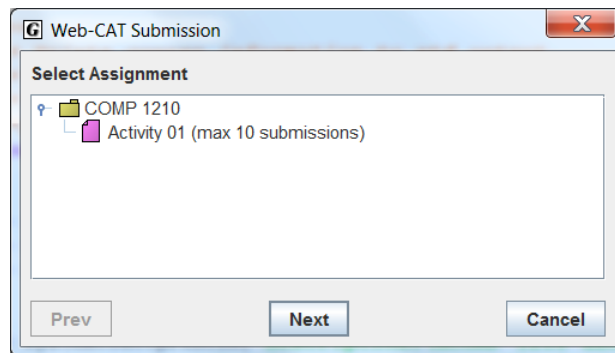
The original author of the program claims the program is correct because it "gets the point across," but you know that it is incorrect because the **actual output** of the program does not match the **expected output** above during testing. Correct the output errors by modifying the program. *Hint: Look for issues in formatting/spacing, spelling, capitalization, etc. One way to skip a line is print an empty line with the statement: `System.out.println();`*

2. Web-CAT

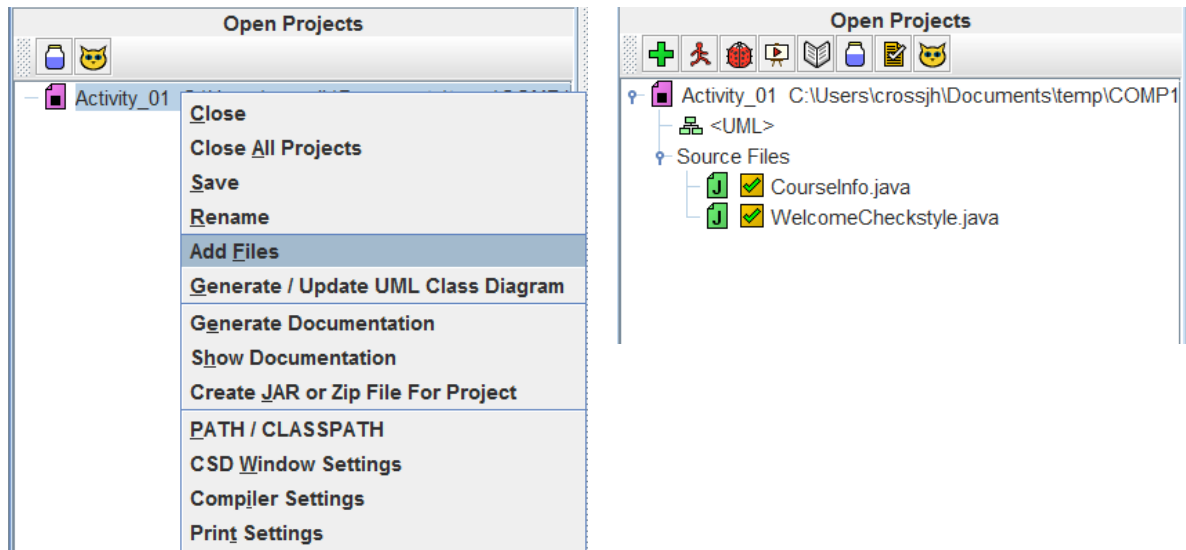
- In jGRASP, navigate to **Tools > Web-Cat > Configure**. In the dialog box, copy/paste <http://webcat.eng.auburn.edu:8080/Web-CAT/WebObjects/Web-CAT.woa/wa/assignments/eclipse> as the Assignment Definition URL (select the URL above then right-click and select **Copy**, and then in the Web-CAT Tool Settings dialog, replace the current entry (if any) for Assignment Definition URL by selecting the text, right click and select **Paste**. See the Web-CAT documentation on the class website for more information.



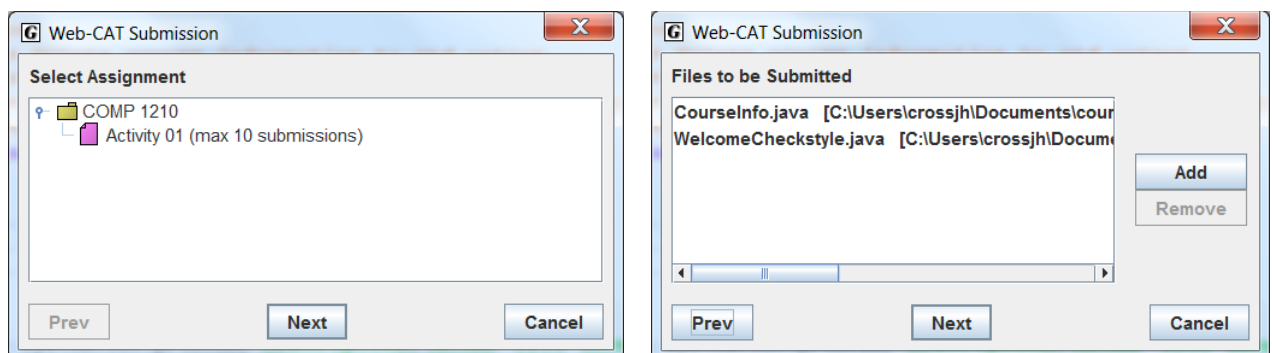
- Click OK. You should now have a Web-CAT button 🐱 on the toolbar.
- Open the Java file that you want to submit then you can submit your program via Web-CAT. Click the Web-CAT toolbar button 🐱 (or go to **Tools > Web-Cat > Submit File**). For example, select the COMP 1210 folder and the assignment labeled **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 assignment in Web-CAT. You will only have 10 tries, so make sure that both files have passed Checkstyle before submission.
- Your assignment Part B will be automatically be graded by Web-CAT as follows.
 - 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 file prior to submission (as described above) OR you can create a jGRASP project, add both Java files to the project, and then submit the project to Web-CAT. 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_01** and click **Next**. Click **Next** again to create the project.
 - Now in the lower part of the Browse tab you should see an **Open Projects** section. 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 (if not already logged in), and click **Submit**.



NOTE: Future activity files will not be submitted to Web-CAT. The purpose of the submission this week was simply to ensure that you are able to submit files to the Web-CAT. After this week, only your project files will be submitted to Web-CAT.