Waukesha County Technical Institute
152-135 Advanced Java

Class 3 Lesson Plan

Discussion Activities:

- **Due Today**
    1. You have continued your study of OOA/D and memorization of the 5 steps. There will be a real quiz today, offering you the chance at earning your first positive points.
    2. You have researched "Java Inheritance" and have posted the required comments to the "Inheritance" Discussion Board on Blackboard.
    3. You have continued to study your Java fundamentals if necessary
    4. You have practiced using Git and GitHub
    5. You have studied the notes on Encapsulation from the Class 2 Plan, including the "EncapCheckList.pdf"

- **General Q&A**

- **Daily Code Puzzle and Java Fundamentals Review**
    - Your review questions

- **Misc. Git and GitHub Notes**
    - How to Delete a GitHub repository. Click on the link for your GitHub project in your web browser. Look for the "Settings" link at the top right of the screen. Click on it and scroll to the bottom. There will be a button or link to delete this repository.
    - Don't forget to use Git by always performing the following tasks in order:
        1. Save your changes
        2. Make sure the project folder is selected (not a sub-folder or individual file)
        3. Perform an Add followed by a Commit
        4. When you're ready to push your work to GitHub make sure the above steps have been completed. Then click the Push button or select Git > Remote > Push from the popup menu for your project. Pay attention to the URL address of the project you are pushing to and make sure it is correct, then follow the wizard instructions.

- **Introduction to Encapsulation**
    - See the "EncapCheckList.pdf" handout on Blackboard
    - See the sample project "Encapsulation.zip" on Blackboard. Note: the version provided with the Class 2 Plan on Blackboard has been replaced with a new and improved version attached to Class 3 Plan. Make sure you

are using the correct one. See the comments at the top of each class and look for Version 1.02.
    - Encapsulation is all about hiding data and complexity
    - Motivation: control access and validation of data, and minimize complexity so that code use and maintenance can be done easily and reliably, resulting in higher quality code. Also, make programs more modular so that individual components can be reused instead of duplicated.

    - How to do it:
        - **Data hiding** via private properties
        - **Procedure hiding** via methods (Single Responsibility)
        - **Method hiding** (and complexity) via public or private helper methods called by public methods. Try to have as few public methods as possible.
        - **Class hiding** (and complexity) via composition and delegation. A Car class, e.g., may have an engine property declared as an Engine type of variable. The Engine class is hidden inside the Car. You access the engine by commanding the Car to start the engine. You never interact directly with the Engine.

- **Architectural Best practices:**
    - Single Responsibility Principle: every class should have at most one responsibility. This does not necessarily mean one method. But all methods should be involved in the one responsibility. Methods should also have only a single responsibility. If a method is doing too much work, divide it up into many smaller methods.
    - Principle of Least Knowledge: every class should know as little as possible about other classes in a system. Encapsulation through composition is a key to making this happen.
    - All class (static) and instance (non-static) properties should be declared private and have public getter and setter methods to provide access.
    - Constants (final) may be public since they cannot be changed. However, consider whether you really want to or need to expose these to other classes. That may be confusing if not needed.

- **Naming Best Practices:**
    - All class names must start with a capital letter and be camel case thereafter. Names should be nouns. Example: MessagerService
    - All class and instance property names should begin with a lower-case letter and be camel case thereafter (no spaces or weird punctuation; no cryptic names). Example: yAxisCoordinate
    - All method names should begin with a lower-case letter and be camel case thereafter (no spaces or weird punctuation; no cryptic names). Example: setLastName(String value)

- No cryptic names! In general it's better to have names that are unusually long vs. unusually short if it helps readability.

Lab Activities:

- **Activity #1**

  The "Encapsulation" sample project on Blackboard will be used for this lab. Please open it in Netbeans. This project is NOT Git-enabled. You must do this when you first start using it so that your work can be version controlled and pushed to GitHub. Enable it for Git and then create an empty project on GitHub and push your local copy to GitHub before proceeding. Remember, add and commit often and push your changes to GitHub when done working for the day.

- **Activity #2**

  In the Encapsulation project you will find several packages. Those NOT labeled "Lab1, Lab2, etc." are for discussion purposes only. These will be discussed now as examples of good and poor encapsulation.

- **Activity #3**

  There are four packages: lab1, lab2, lab3 and lab4. You will work on each of these in order, practicing Property, Method and Class Encapsulation. Each lab gets progressively harder. Perform them in order. Instructions for each lab are at the top of the "Employee" class in each lab. Read these instructions carefully. For lab4 a solution is also provided. Please do not look at the solution until you have made your best effort to solve lab4 on your own. Only peek if you have to.

  Please note that much of this lab work involves Critical Thinking and lots of trial and error, where you consider what you have learned and try to find ways to apply those skills to this new problem. This is the best way to learn but it may feel like you have been set adrift, alone in the sea, without any help at all. Not true! You can get help – from your peers and from your instructor. But don't ask for help until you have tried your best. And don't let anyone write the code for you.

  When you have done the very best you can (remember you aren't required to be 100% correct or complete – just your best effort) push your final changes to GitHub. This lab is due by start of next class.

Resources:

- OOA/D: Object-oriented Analysis and Design Study Guide (Blackboard)
- Introduction to Git Version Control (Blackboard)
- Encapsulation Check List ("EncapCheckList.pdf") on Blackboard
- The "Ecapsulation.zip" sample project on Blackboard
- Using Git in Netbeans: http://netbeans.org/kb/docs/ide/git.html
- Netbeans IDE web site: http://www.netbeans.org

- Netbeans Help Menu in the program; plus, Netbeans tutorials on their web site
- Java SE API: http://docs.oracle.com/javase/8/docs/api/

Preparation Work for Next Class: Independent Research and Practice

1. Complete the Git and Encapsulation lab by start of next class
2. Continue contributing the “Inheritance” Discussion Board