CS 200 Sections 02 & 04 Spring 2013

Week #2: Top 3 Lessons Learned

Day Two: Personal Lessons

The top personal lessons that I learned from this lecture were from both parts of chapter one (1.3) were (1) what an algorithm was, (2) the Software Development Cycle and its phases (Design, Implementation, Testing, Debugging, and Documentation), (3) the types of errors in programming (compile, run-time, logic), (4) the Scanner Class, (5) the Java API Hierarchy, (6) what inheritance is in programming and how it is used, (7) what polymorphism is and what it does in programming, (8) what flowcharting is and why it is important to programming, and (9) what pseudo-codes are and how they are important to programming. Overall, those are my personal "take away" lessons that I learned from the second day of this class.

- D. Mc Manus

Here are my top 3 lessons learned from week 2:

1. An algorithm is a set of directions for solving a problem that must be so precise that someone can follow them without having to fill in details or make any decisions not specified in the instructions.

2. The software development cycle consists of 5 phases: design, implementation, testing, debugging, and documentation.

3. Object features consist of attributes (Characteristics of an object) and behaviors (actions an object can take).

- J. Hoffman

1. always start with a flow chart

2. be specific when getting user input make sure its not "breakable".

3. always write code in small pieces and test. and /*always comment your code

-G. Martinez

1) there are different types of phases in ~~Java~~ software development, like for example, the designing phase, implementation phase, the testing phase and debugging phase too.
2) how there are different types of errors such as compile, run-time and logic and how it each of them worked
3) how the Java API Heirarchy worked in the programs and that uses different objects in them.

- J. Morales

1. Using “+” to concatenate text and java syntax.

- Java syntax- System.out.println("Hello " + name + "! Let’s do some math.");

- Screen output - Hello Mustafa! Let’s do some math.

2. To display a prompt for user to input data, the Java utilities API must be imported. The scanner object syntax should be inserted before the public class syntax.

3. Types of errors: Compile, Run-Time, Logic

• Compile – Syntax error caught by compiler
• Run-Time – Error caused by exceptions resulting in program terminating
• Logic – Program compiles and runs but produces inaccurate resultso

- M. Khan

1. The flowchart is the recipe for the code
2. There are 5 phases for writing an algo
   Design
   (Then a plan of attack is created using flowcharting or psuedocode)
   Implementation phase
   Testing phase
   Debugging phase
   Documentation
3. There are three types of bugs or errors
   Syntex
   Run-Time
   Logic

- K. Hanrahan

Three things that I learned in week 2 are that it is imperative to create a flowchart before you start to write the program. The flowchart will guide you through the steps that you need to go through. Second, when debugging a program, you should fix the first error because that error may cause the other errors that are occurring. Third, you should always comment your code so that you can remember what you did when you look back at it in the future. In addition, you should also comment your code so that other programmers can change or maintain your code.

- J. Gomez

1. Flowcharts are very important. They help lay that groundwork so once you get to coding you can focus on that and not have to worry so much about the logic and how the system will work since you've already worked it out.
2. Quizzes are a lot harder than I thought. Not used to being tested on actually applying the subject.
3. I learned that I can forward my Nmail to my Gmail which makes it a lot easier to check emails now.

- P. Zito

My top 3 lessons that I have learned are:
1. Java.lang is imported into every program by default.
2. Flowcharts are used to bypass the language barrier.
3. If you can't solve a problem, you should get some rest and try to solve it the next day.

- D. Starostka

1. For an algorithm to be qualified, the directions in the algorithm must be expressed so completely and precisely that anyone can follow the directions without having to fill in any details or make any decisions that are not fully specified in the instructions. An algorithm is like a recipe. I think of the "checkbook" example you gave.

2. Remember when using // for a single line comment, if the comment overflows to the next line the compiler will want to read it as a piece of code. If you notice your comment will be more than one line, use /* at the start of your comment and */ at the end of your comment.

3. Logic errors. Your program compiles and runs without any errors caught by the compiler, but your result/output may be inaccurate. Example: using a + sign instead of a - sign. Produces inaccurate results, but the compiler will not know that.

- T. Blanchard

I think the most valuable lesson I learned is if you think you studied enough and know the material, go back though it again and see if you can be more detailed because in some cases, you'll know part of the answer, but in all cases, you better be able to backup your answer. Another thing I learned is that an algorithm has step by step instructions that are so specific to solve a problem, that it can't be screwed up. Last and probably the most exciting for me was when the light bulb went off in my head about polymorphism. I've heard the term and tried to make sense of the definition, but something in the explanation in class clicked with me and it made sense how different data types can be accepted by a class and the correct method for that data type would be used to perform the calculations and return the result.

- E. Zacharias