COP 3330, Spring 2013

Intro to Object Oriented Programming - I

Instructor : Arup Ghosh

01-16-13

School of Electrical Engineering and Computer Science
University of Central Florida

Programming

- A *program* is a set of instructions to perform a given task. Typically the task solves some problem or part of a larger problem.
- In your CS classes, so far, you've learned the fundamental programming concepts, such as loops, assignments, conditional statements, and so on. Hopefully, you learned one or more ways to implement each of those constructs.
- You also learned that computer science is not just programming, but rather that programming is just a tool of a computer scientist.

Programming

- What you might not have learned yet, are the advantages and disadvantages of the different ways to write a program to solve the same problem.
- For example, at a low level, there are many different ways to implement a structure such as a loop, e.g., you could use a for loop, a while loop, or recursion. At a higher level, there are many different ways to perform a task, e.g., a merge sort versus a selection sort. At even higher levels there are many different ways to divide a program into modules such as classes and methods.

Programming

- At this point you might reasonably ask whether this discussion really matter. If two designs, one with few classes and methods and one with many, both solve a problem, does it matter which one you use?
- Similarly, if all versions of a loop correctly perform a computation, does it matter which one you use? If two algorithms both work correctly, does it matter that one is faster, especially given the increasing speeds of processors?
- The answer is:

YES, it really does matter!

Programming

- Before we write a computer program to solve a problem, we should organize its solution. (*problem solving*)
- Normally computer scientists are good at problem solving, but we should apply certain methods to solve problems (especially when we solve large problems) elegantly.
- Good problem solving steps make life easier when we write a computer program to solve a given problem. We will talk about top-down approach (divide and conquer) when we organize solutions for problems.
- We will also talk about object-orient software development techniques.

Programming

- Addressing the pressures of cost and time are beyond the scope of this course. Our focus concerns the skills, knowledge, and experience that developers need in order to do high quality work.
- In this course, you'll learn some of the things a software developer should know in order to design and implement high quality software systems and be able to redesign existing software systems in a way that fight the forces of mud.

Software Engineering

- To create high quality software, you first need to know what it means for software to have high quality.
- Unfortunately, such quality is not easily measured.
- The field of software engineering was created with the purpose of understanding and devising ways of measuring the quality and reliability of software. One of its tools has been the application of engineering principles to software development.
- Software engineering traditionally divides the software process into stages, including specification and analysis, design, implementation, and maintenance.

Software Engineering

1. *Specification and Analysis*
   - Determine the precise behavior the final software system is to have. Exhaustively determine what the system should do in all possible cases. This stage also determines what the user interface will be. This stage is performed in close cooperation with the clients of the system and some users of the system. You must understand the problem completely and determine what is required for its solution.
2. *Design*
   - Determine the components of the system, what each component is responsible for doing, and how the components will interact. For example, this stage includes determining the data structures that will be used and the kind of data that will be stored in those structures.
   - Develop a list of steps (algorithm) to solve the problem
   - Refine steps of this algorithm. (Divide and Conquer)
   - Verify that the algorithm solves the problem, i.e. the algorithm is correct

Software Engineering

3. *Implementation*
   - Programmers develop the code of the software system components using an appropriate (or several appropriate) programming languages.
   - You have to know a specific programming language (java).
   - Extensive testing is part of this stage.
4. *Maintenance*
   - Once the product has shipped or been placed into service, programmers repair defects, update or enhance the software to extend its usefulness.
   - Testing is also crucial in this stage.
   - Use different test cases (not one) including critical test cases.

Criteria For Elegant Software

1. Usability
   - Is it easy for the client to use?
2. Completeness
   - Does it satisfy all of the client's needs?
3. Robustness
   - Will it deal with unusual situations gracefully and avoid crashing?
4. Efficiency
   - Will it perform the necessary computations in a reasonable amount of time and using a reasonable amount of memory and other resources?

Criteria For Elegant Software

5. Scalability
   - Will it still perform correctly and efficiently when the problems grow in size by several orders of magnitude?
6. Readability
   - Is it easy for another programmer to read and understand the design and code?
7. Reusability
   - Can it be reused in another completely different setting?
8. Simplicity
   - Is the design and/or implementation unnecessarily complex?

Criteria For Elegant Software

9. Maintainability
   - Can defects be found and fixed easily without adding new defects?
10. Extensibility
   - Can it be easily enhanced or restricted by adding new features or removing old features without breaking the code?

- The first four properties mostly relate to the functional requirements of the software, i.e., does it do what the requirements documents say it is supposed to do?
- The last six properties are the ones that we will be most concerned with in this course. They address software style and how easily the software can be changed.