

Class 19: Basics of Object-Oriented Programming

Held: Monday, March 5, 2007

Summary: Today we begin our exploration of object-oriented programming.

Related Pages:

- EBoard.
- Reading: CS302 Students: Answers to Questions on Polymorphism and Overloading.

Due

- HW5.

Notes:

- The Mid-Semester Exam is ready.
- Wednesday and Friday, you will be watching videos in class. In lieu of reading response, please send me one-paragraph responses to those videos *after* you watch the videos.
- The reading for Monday the 12th is an online document, available at <http://java.sun.com/docs/books/tutorial/reflect/index.html>.

Overview:

- What is object-oriented programming?
- Three key issues: Encapsulation, Inheritance, Polymorphism.
- Why OOP?
- A biased history.
- Polymorphism vs. Overloading.

What is object-oriented programming?

- As most of you know, object-oriented programming (and, similarly, object-oriented programming) is a strategy that emphasizes *objects*.
- Objects group data and functionality. That is, an object stores information and provides methods that use or modify that information.

Three key issues: Encapsulation, Inheritance, Polymorphism

- To most object-oriented programmers, object-oriented programming is more than just the idea of objects. It also incorporates three related ideas:
 - Objects *encapsulate* data.
 - An object (or class) may *inherit* methods and fields from other objects (or classes)

- Objects can be used *polymorphically* in place of each other.

Why OOP?

- Why not?
- Strives to be a *silver bullet* for software design.
- More reuse with inheritance and polymorphism.
- Well-designed objects can be reused in many contexts.
- Natural for some problems (e.g., modeling).
- Natural for modern application design (i.e., user interfaces and event-driven programming).

A Biased History

Sam's grounding quote (emphasis mine) from Nygaard and Dahl

The principle extensions which convert ___ to ___ provide the ability to:

1. Declare a *class*
2. Generate *objects* of the declared class
3. Name the generated *objects*
4. Form a hierarchical structure of *class declarations*

- Sketchpad.
- Simula and Simula 67: Modeling.
- Smalltalk (and its variants): Purity; Language as environment.
- CLOS (early 1980's): Objects in LISP
- Many years of oop as primarily research topic.
- C++: Add some object-oriented features to C.
- Java: Do C++ right.

Polymorphism vs. Overloading.

- Both polymorphism and overloading permit a method to behave differently based on its parameters.
- In overloading, you must write different versions of the method for each set of parameters.
- In polymorphism, you write one method, and it behaves differently by using methods of different objects.
 - Of course, those different objects needed to implement different versions of the same called method (or methods).
- Consider the simple example of square roots.