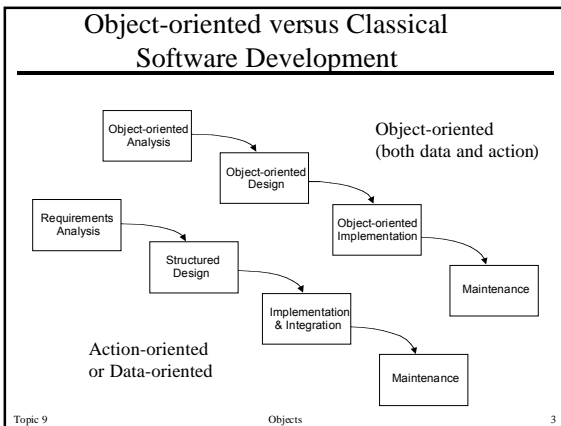


ICS 121 Topic 9: Object-Oriented Design

Introduction to Object-Oriented Design
Comparison to Structured Design
Background on Objects and
Object-Oriented Programming

Object-Oriented Software Development

- Object-Oriented Methodology
 - development approach used to build complex systems using the concepts of object, class, polymorphism, and inheritance with a view towards reusability
 - encourages software engineers to think of the problem in terms of the application domain early and apply a consistent approach throughout the entire life-cycle
- Object-Oriented Analysis and Design
 - analysis models the “real-world” requirements, independent of the implementation environment
 - design applies object-oriented concepts to develop and communicate the architecture and details of how to meet requirements



Object-Oriented vs. Structured Design

- Structured design approaches focus primarily on either actions or data
 - Action-oriented approaches (a.k.a. process-oriented, function-oriented)
 - Structured systems analysis [DeMarco, 1978]
 - Finite state machine modeling
 - Data-oriented approaches
 - Entity-relationship modeling [Chen, 1976]
 - Jackson system development [Jackson, 1983]
- Object-Oriented design is a more synergistic approach
 - Object = Data + Actions
 - data and actions are both treated as first class citizens

General Advantages

- Understandable
 - maps the “real-world” objects more directly
 - manages complexity via abstraction and encapsulation
- Practical
 - successful in real applications
 - suitable to many, but not all, domains
- Productive
 - experience shows increased productivity over life-cycle
 - encourages reuse of model, design, and code
- Stable
 - changes minimally perturb objects

Advantages *with respect to* “Principles”

- Separation of concerns
 - developers focus on common versus special properties of objects
- Modularity
 - specifically in terms of classification of objects
- Abstraction
 - allowing common versus special properties to be represented separately
- Anticipation of change
 - new modules (objects) systematically specialize existing objects
- Generality
 - a general object may be specialized in several ways
- Incrementality
 - specific requirements (e.g. performance) may be addressed in specializations

Object Model Notation: Introduction

Class Name

```
InstanceVariable1
InstanceVariable2: type

Method1()
Method2(arguments) return type
```

Classes are represented as rectangles;

The class name is at the top, followed by attributes (instance variables) and methods (operations)

Depending on context some information can be hidden such as types or method arguments

(Class Name)

```
InstanceVariable1 = value
InstanceVariable2: type

Method1()
Method2(arguments) return type
```

Objects are represented as rounded rectangles;

The object's name is its classname surrounded by parentheses

Instance variables can display the values that they have been assigned; pointer types will of ten point (not shown) to the object being referenced

Topic 9 Objects 13

Object Communication

- Objects communicate via method invocation
 - This is known as message passing
 - Legal messages are defined by the object's interface
 - This interface is the only legal way to access another object's state

Topic 9 Objects 14

Objects: Terminology (partial review)

- Class**
 - set of objects having the same methods and attributes
 - has a specification and an implementation
 - behavior is defined by the operations that can be performed on objects belonging to the class
- Method**
 - action that can be performed on any member of a class
- Encapsulation**
 - packaging the specification and implementation of a class so that the specification is visible and the implementation is hidden from clients
 - Instantiation
 - the creation of a new object belonging to a class

Topic 9 Objects 15

Objects: Terminology, *continued*

- Aggregation**
 - Objects representing components are associated with an object representing their assembly (e.g. consists-of)
 - A mechanism for structuring object models

Topic 9 Objects 16

Aggregation: example

Topic 9 Objects 17

Objects: Terminology, *continued*

- Generalization**
 - Allows a class, called a supertype, to be formed by factoring out the common state and methods of several classes, called subtypes (is-a)
 - Specialization is the converse case

Topic 9 Objects 18

