Chapter 5

Object-Oriented Programming
Traditional vs. Object Oriented Programming Paradigm

• Traditional Paradigm
  Algorithm identification
  Functional decomposition
    Control Flow construction
    Data Structure required for functions

• Object Oriented Paradigm
  – Object identification
    » ADTs that represent the problem domain.
  – Object Characteristic definitions
    » Internal attributes of object.
  – Object Method (operation) definitions
    » Abstract operations (i.e., interface) for the ADTs.
  – Solution is a sequence of calls to methods.
Example: The *File* Object

- A set of **Characteristics**
- A set of **Methods** (operations).

![Diagram of a file object with properties and methods]

Copyright 1995 - 2000 C. Gotsman & 1995 - 2016 Y.M. Kimchi, Computer Science Dept. Technion
Why OOP?

• The **objects** are the **most stable** factor in the problem
  – Algorithms may be improved
  – Implementations may be modified
  – Interface may undergo major changes

• If the data structure for the objects is defined according to *algorithm / implementation / interface* it is very likely to backfire when one of them is changed.

• Objects are the appropriate level at which decisions about **encapsulation** (**information hiding**) are made.
Example: A Student

Characteristics:
- The **Person** parts: Name, ID, Address
- The **Student** parts: Faculty, program, Years records
- **Program** parts: Requirements, Courses records.

Methods (operations):
- **Enrole** to an **institution** (a person becomes a student)
  - Needs to connect to an **institution** and a **faculty**
- **Enrole** to a **course**, **Submit** HW-2, **Take an Exam** in a **course**
  - Needs to connect to the **course**, for each of these tasks
- **Change** a **program**
- **Print** yourself
Interface vs. Implementation

What to do

Message

Method

How to do
Examples of Messages

A Set Object can receive messages such as:

- **Add** a given element
- **Remove** a given element
- **Is** a given element a member?
- **What** is your size?
- **Unite** yourself with another set object

A Vehicle Object can receive messages such as:

- **Move** forward at a given speed (or to a certain distance)
- **Accelerate**
- **Stop**
Messages vs. Methods

In Object Oriented Programming, the message “rotate n degrees clockwise” would be implemented differently by different objects:

- **Shape**: Do nothing if $n = 360k$, otherwise: rotate . . .
- **Circle**: Do nothing.
- **Square**: Do nothing if $n = 90k$, otherwise: rotate . . .

The crucial part is that all may appear to the client as “shape”, but the correct operation will be invoked.
OOP Toolbox

Language

Methodology

Patterns

Mechanisms
- Encapsulation: Hide implementation
- Abstraction: Classes express concepts
- Inheritance: IS-A relation among abstractions
- Polymorphism: Same message invokes different methods

Principles
- Open Closed
- Liskov Substitution
- Dependency Inversion
- Interface Segregation
- Single Responsibility

Roles
- Builder
- Adapter
- Composite
- Command
- Iterator
- Observer
- State

References on last slide
OOP Will

• Encourage **use of ADTs and their extensions** (classes, modules, packages, …)

• Make **design decisions** easier

• Encourage **code reuse** (within same/other project(s))

• The above will result in
  – Making code predictable to use
  – Ease program implementation
  – Ease understanding of program structure
  – Ease program maintenance
    » Corrections
    » Modifications
    » Enhancements
  – Help in documentation
OOP Will NOT

- Design your program for you
- Prevent you from designing a bad program
- Provide algorithms
- Perform data management
A Short OOP Reference List

Language
Books & Web
Bjarne Stroustrup
Scott Meyers
Marshall Cline
Bertrand Meyer

Methodology
Web/book Example
Robert Martin (Uncle Bob) Principles and Patterns
The original website was lost: Find it using WayBackMachine, at http://archive.org/web/, look for www.objectmentor.com/resources/index 2006 => Published Articles => Robert Martin => Principles and Patterns (The references are there too)

Web/book Example
Gamma et. al (GoF), Design Patterns (1994)
http://www.oodesign.com/