Lecture 1: Introduction

Sunghyun Cho
Professor
Division of Computer Science
chopro@hanyang.ac.kr

HANYANG UNIVERSITY

Contents

- Functions of Modern Computer
- Data Structures
- Abstract Data Type
- OOD Goals (§2.1)
- OOD Principles (§2.1)
- Object-Oriented Programming (OOP)

Lecture Objectives

- To be able to explain what is a **Data Structure**
- To be able to explain what is meant by **Abstract Data Type (ADT)**
- To be able to explain the principles and goals of **Object-Oriented Design (OOD)**

Functions of Modern Computers

- First computer : Numeric processor
  - Quickly perform long sequences and mathematical calculations
- Modern computer : Information Management
  - Larger memory
  - Faster processor

Big Data

- Data stored in FaceBook everyday

  - 2.5B - content items shared
  - 2.7B - 'Likes'
  - 300M - photos uploaded
  - 100+PB - disk space in a single HDFS cluster
  - 105TB - data scanned via Hive (30min)
  - 70,000 - queries executed
  - 500+TB - new data ingested

Big Data & Apps. (cont'd)

- Store, analyze, search, transfer & update huge collections of complex data
  - Data must be well organized
    - ☞ Needs good data structures and algorithms
- A myriad of different applications
  - Creating S/W has become a complex enterprise
    - ➢ The interactions of pieces of S/W written by many different peoples in several different organizations
    - ➢ Upgrade old S/W
- Create Software that are
  - Conceptually simple enough to understand
  - Powerful enough to solve hard problems efficiently

Data Structures

- What Do We Mean by Data?
  - Function of a program : verbs in the programming language (e.g. add, write etc.)
  - Data : nouns of the programming world.
    - The objects to be manipulated
- Single integer : useful for a counter, a sum, or index in a program
- In general we must deal with data that have lots of parts (e.g. list)

Data Structures (cont'd)

- Data Structure :

  A collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data elements

e.g. Java array

allows individual items to be stored and retrieved based on an index ([0],[1],[2]…) that is assigned to each item

Data Structures (cont'd)

- Features of Data Structures (DS)
  - **Logical arrangement of data element**
    - Can be decomposed into their component elements
  - **The set of operations to access the elements**
    - The arrangement of the elements is a feature of the structure that affects how each element is accessed
- The arrangement of the elements and the way they are accessed can be encapsulated.

Data Structures (cont'd)

- A real-life example of DS : library
  - Component element : books
  - Accessing to books :
    - The physical arrangement of books on the shelves.

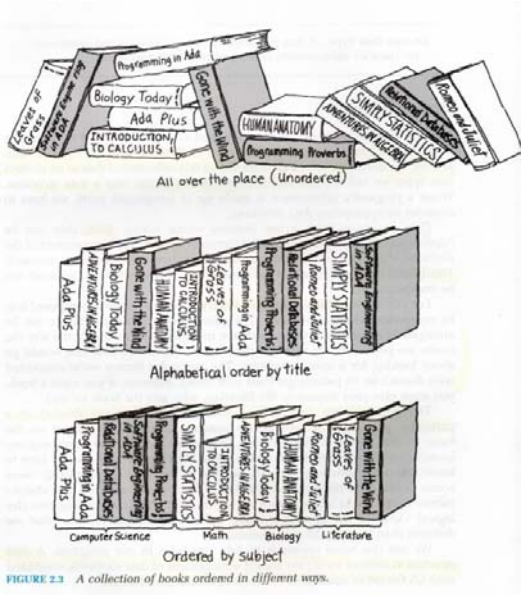

FIGURE 2.3 *A collection of books ordered in different ways.*

Abstract Data Types (ADT)

- Separate the specification for a data structure (what it does) from the implementation of a DS (how it does it)

- Abstract Data Types (ADT)
  - A well-specific collection of data and group of operations that can be performed upon data : specifies
    - The type (characteristics) of data stored
    - The operations supported in them
  - ADT specifies what each operations does, not specifies how it is implemented or represented in the program

ADT (cont'd)

- Data : Instances of an Abstract Data Type that includes a repertory of methods for performing operations on the data, not a collection of bytes and addresses.
  - *Example* : Java
    - Expression of an ADT : "***interface***"
    - Realization of an ADT : "***class***"
      - Java: Specify how the operations are performed
- One of the main ideas of the object-oriented approach : Data should be represented as being encapsulated with the methods that access and modify them.

Object-Oriented Design (OOD)

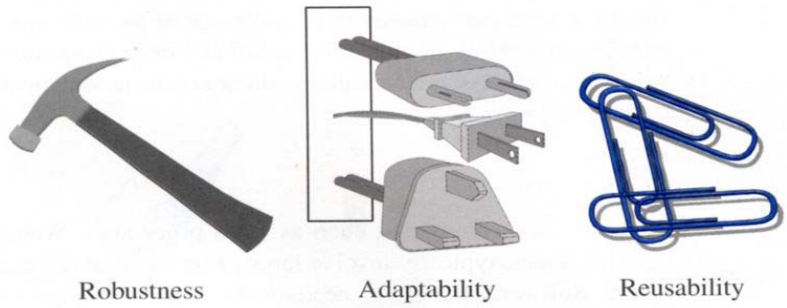

Figure 2.1: Goals of object-oriented design.

OOD (cont'd)

- Fundamental Goal
  - Production of quality software
- Sub-goals
  - Quality DS
  - Quality algorithm implementation

Goals of OOD

- Robustness
  - Software : Correct and Robust
  - Robust : capable of
    - Handling unexpected inputs that are not explicitly defined for its application
      - Critical in life-critical applications
      - "***Exceptions***"
    - Producing correct solutions, even given the well-known limitations of computer
      - *Ex*. Store more elements in an array that originally expected
      - "***java.util.Vector***" class : expandable array

Goals of OOD (cont'd)

- Adaptability (Portability)
  - Software needs to be able to evolve over time in response to changing conditions in its environment – cpu speed, network speed
    - Word processor
    - Web browsers
    - Internet search engines : last for many years
  - The ability of S/W to run with minimal change on different H/W & O.S. platforms

Goals of OOD (cont'd)

- Reusability
  - Code should be usable as a component of different systems in various applications
    - S/W is designed in a way that makes it easily reusable in future applications
    - Application level : Sorting Program
    - Language level
      - Java methods : "*Extends*", "*Override*"

Principles of OOD

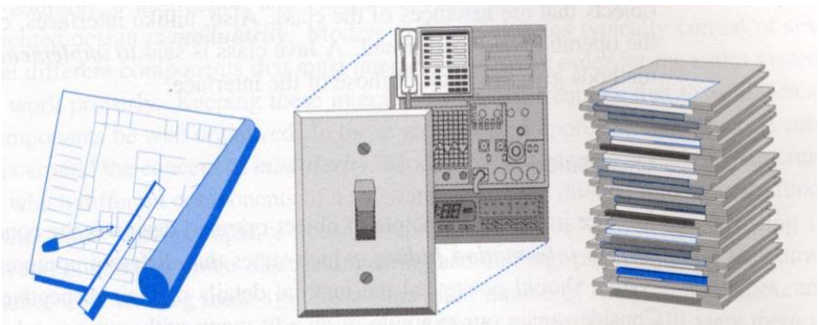

Figure 2.2: Principles of object-oriented design.

Principles of OOD (cont'd)

- Abstraction
  - Distill a complicated system down to its most fundamental parts
  - Describe these parts in
    simple (simplicity) and precise (Clarity) language
    ⇒ leads to understandable & correct implementations
  - *Example* : Text-editor GUI : abstraction of "edit" menu

Principles of OOD (cont'd)

- Encapsulation (Information Hiding)
  - Different components of a S/W system should not reveal the internal details of their respective implementations – separation of specification from implementation
  - One of the main advantages : Encapsulation gives the programmer freedom in implementing the details of a system
  - Principles of encapsulation : All the different components of a large S/W system should operate on a strictly need-to-know basis

Principles of OOD (cont'd)

- *Example* : GUI
  - ➢ We completely understand how to use it without understanding exactly how it is implemented.
  - ➢ Don't know
    - how the menu is drawn
    - How selected text to be cut or pasted is represented
  - ➢ The code associated with the edit menu should not depend on all of these details to work correctly
  - ⇒ The edit menu should provide an interface that is sufficiently specified for other software components to use its method
- Encapsulation yields adaptability : It allows the implementation details of parts of a program to change without adversely affecting other parts

Principles of OOD (cont'd)

- Modularity
  - An organizing structure in which different components of a software system are divided into separate functional units.
  - The structure imposed by modularity helps to enable software reusability : modules written in an abstract way to solve general problems can be reused when instances of these same general problems may arise in other contexts.
  - Organizing components in hierarchical fashion : Groups together common functionality at the most general level, and views specialized behavior as an extension of the general one : "is a" relationship

Principles of OOD (cont'd)

- Example : A house or Apartment
  - Consists of several interacting units :
    - Electrical, heating, cooling, plumbing, & structural
      - One giant jumble of wires, vents, pipes, and boards.

      ⇒View these system as separate modules that interact in well-defined ways.

      ⇒Bring a clarity of thought that provided a natural way of organizing functions into distinct manageable units.
    - Some parts might be different depending on the type of buildings

      ⇒organize the various structural components in a hierarchical fashion.

Object-Oriented Programming

- OOP : The implementation of an OOD.
  - An approach to programming in which data occurs in tidy package called objects
  - Manipulation of an object happens with function called methods
  - The Java mechanism to create objects and methods : class
- Three essential ingredients in OOP
  - Encapsulation
  - Inheritance
  - Polymorphism

OOP (cont'd)

- OOD results in a hierarchy of objects
  - A special relationship with each other objects
  - Objects get more specialized the lower in the hierarchy
  - Each new object inherits the characteristics of the parent object

- Polymorphism : The ability to determine which of several methods with the same name is to be invoked