

Lecture 2

Intro. To Software Engineering and Object-Oriented Programming (1/2)

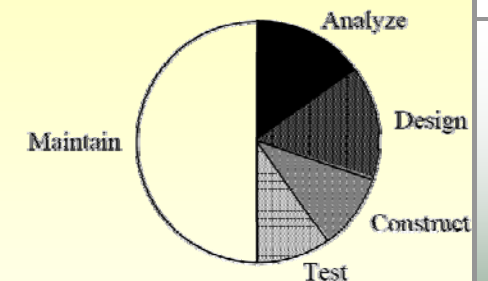
What Is Good Software(Program)?

Software Crisis

- rapid increases in computer power and the complexity of the problems (SW).
- Software technology can not follow the complexity.
- the difficulty of writing **correct, understandable, and verifiable** computer programs
- SW Symptoms
 - Unmanageable
 - difficult to maintain
 - Over-budget.
 - Over-time (late)
 - Poor quality
 - Software often did not meet requirements.
- Cause : **complexity**

Software Life Cycle

1. Requirement Analysis
 - Problem specification
2. Design
 - Program structure
 - Module specification
3. Implementation
4. Test
 - unit test
 - integration test
5. Maintenance



Fred Brooks states that over 90% of the costs of a typical system arise in the maintenance phase, and that any successful piece of software will inevitably be maintained.

Software Engineering

■ Definition

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
- The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently

■ Goals

- produce a correct, efficient, and reliable software
- Minimize the cost of software development and maintenance

Goals of SE

■ Maintainability

- the ability to easily make changes, enhancements, or improvements.

■ Dependability

- the ability to rely on the software to function properly when needed.

■ Efficiency

- the ability for software to use computing resources effectively (mainly space and time).

■ Usability

- the ability for the end user to easily and effectively put the software to proper use.

Software Properties

- **Modularity** – divide and conquer.
- **Encapsulation** – hide the implementation.
- **Localization** – collect similar things together.
- **Abstraction** – simplifying complex reality.
- **Uniformity** – make everything look similar.
- **Completeness** – do everything required.
- **Confirmability** – be able to prove that the software works properly.

Programming Methodology

- Unstructured Programming
- Procedural Programming
- Modular(Structured) Programming
- Object Oriented Programming

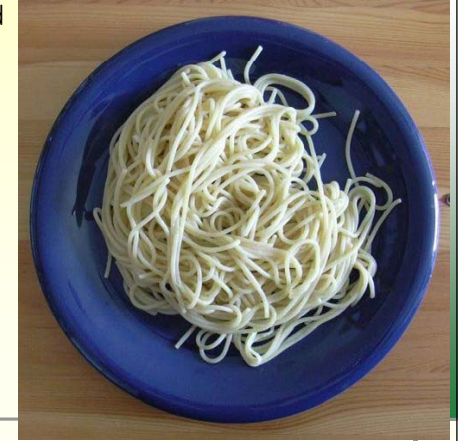
Unstructured Programming

- Consists of only one main program.
- a sequence of commands or statements
- Only global data
- Use goto statement

- Disadvantage
 - If there is the same statement sequence?

Spaghetti Code

- Code with complex and tangled structure
 - With **goto**'s jumping all over the place
 - Almost impossible to understand
 - Hard to follow the flow
 - Difficult to modify



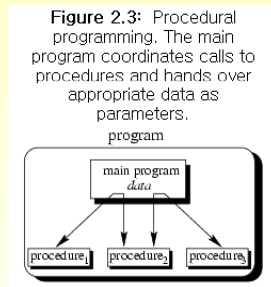
Procedural Programming

- The abuses of spaghetti code led to structured programming languages to support procedural programming

- Program
 - A sequence of procedure calls
- Procedure
 - the same sequence of statements

- More structured : reduce repetition
- More error-free

- Structured Programming
 - Top-down approach
 - Divide and conquer
 - Functional decomposition

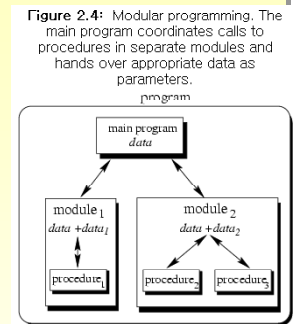


Modular Programming

- SW is composed of separate parts (module)

- procedures of a common functionality are grouped together into separate *modules*

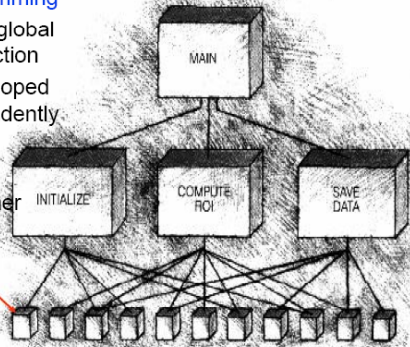
- module
 - Program uses modules through interface.
 - Interface : visible, implementation : hidden
 - Improve maintainability



Don't use Global Variables

- Using global variables : unsafe!

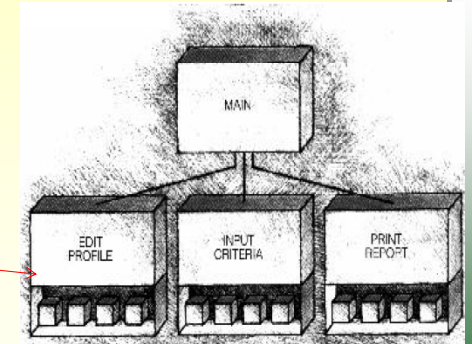
- Sharing data (global variables) is a violation of modular programming
- All modules can access all global variables without any restriction
 - No module can be developed and understood independently.
- Global Data are dangerous
 - This makes all modules dependent on one another



Copyright: OOT A Managers' perspective, Dr. Taylor

Modularized Data

- Localize data inside the modules
- This makes modules more independent of one another
- Safe!



Copyright: OOT A Managers' perspective, Dr. Taylor

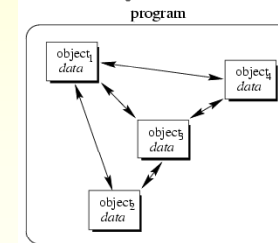
Problems of Procedural/Modular Programming

- Decoupled (separation of) data and operations
- Doesn't work well in large systems
 - Difficult to debug, extend, and maintain
 - Code reuse limited
- Explicit creation and destruction
- Many others compared to OOP

OOP (Object Oriented Programming)

- Program consists of a set of interacting objects.
- Each object keeps its own state.
- Combine data and operation into an object

Figure 2.6: Object-oriented programming. Objects of the program interact by sending messages to each other.



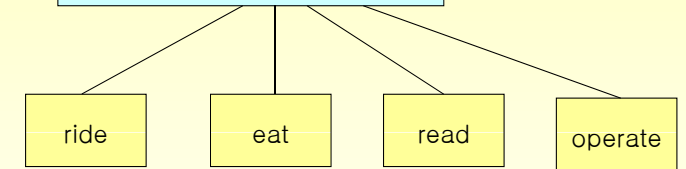
What is an Object?

- Object is an instance of a class
- Objects have
 - identity
 - attribute
 - behavior (method)
 - relationships with other objects

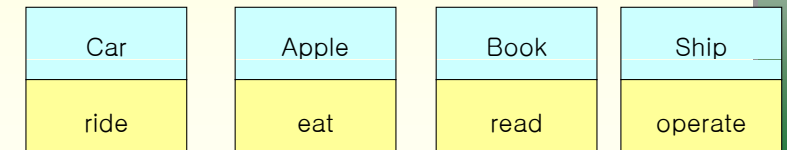
Structured Programming vs OOP

Structured Programming

Car, Book, Apple, Ship



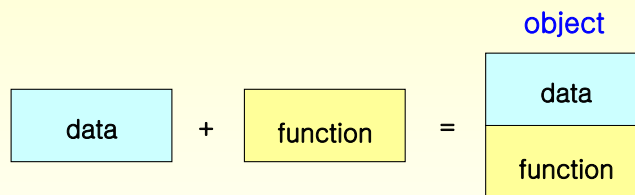
OOP



Structured Programming vs OOP

Structured Programming

data + function = program



OOP

object + object = program

Views of OOP

- OOP does not consider the problem domain as just a flow of processings.
- OOP defines objects related to the problem domain and regards these objects communicate information each other (through message passing).
- On the other hand, structured programming treats data and operations independently even though they are related.

Software Components(부품화)

- In OO programs, an object works as a component(부품) of software
- Interface of an object should be well defined.



H/W 부품으로 컴퓨터 만들기



S/W 부품으로 프로그램 만들기

Why Object Oriented?

- A new way to think of software
- Reuse software, not rewrite
- Increase
 - Programmer productivity
 - Quality of software
 - Understandability of software
 - Scalability (extensibility, reusability) of software
 - Lifetime (maintainability) of software