Recent Programming Languages의 문제점

- Class, Subclass, Inheritance에 대해 가볍게 대처
- Always taking an easy and simple way
- Multi Paradigm Programming Language를 주장

- Programming Language Theory를 대폭 무시
  - Type에 대한 이론들에 무지와 무시
  - Correctness of Program의 중요성을 간과
  - Abstract Data Type의 기본 취지의 퇴색

본수업에서 강조하는 점

- The Object-Oriented Paradigm의 올바른 이해
  - Historical Background
  - Major Motivation of OOP features

- Class, Class Hierarchy, Inheritance의 Theoretical Famework 의 정확한 이해

Object-Orientations in Computer Science

서울대학교 컴퓨터공학부
Internet Data Base LAB
교수 김형주

Contents

THE OBJECT-ORIENTED PARADIGM

- Historical Background
- Object-Oriented Concepts
  - Core Elements
  - Encapsulation
  - Message Passing
  - Inheritance
  - Late Binding
  - Method Combination
- Benefits

HISTORICAL BACKGROUND (1)

- The Root of OO ➔ SIMULA (1966)
  - Dahl & Nygarrd 66 CACM paper
  - Simulation Language
  - Notion of Class

- SMALLTALK (1970 - 1980)
  - Xerox PARC, Adele Goldberg
  - The first substantial, interactive, display-based implementation
  - An integrated programming language environment
  - User interface
  - Class library

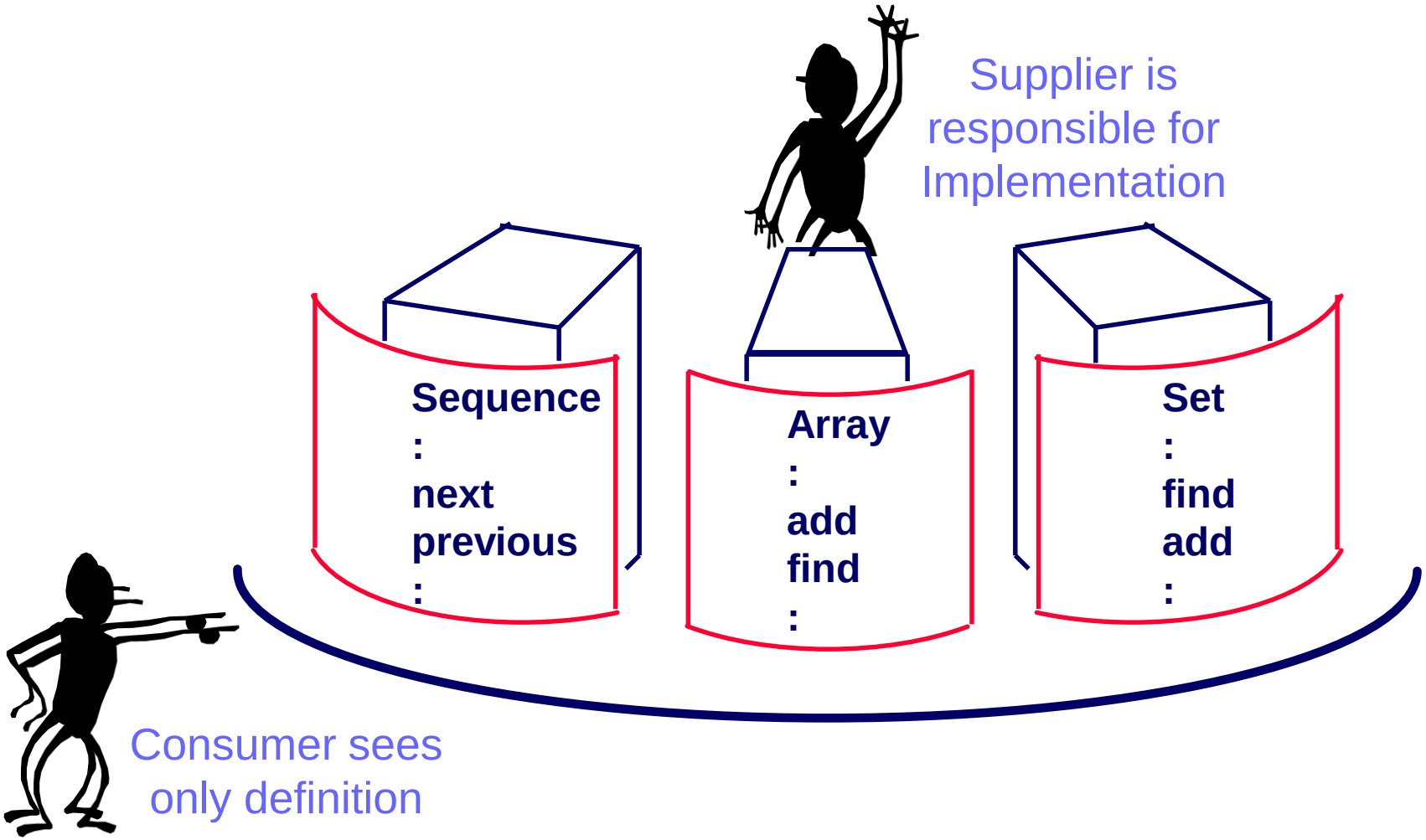
HISTORICAL BACKGROUND (2)

- AI Community (1980's)
  - KR/1, Frame: Knowledge Representation
  - Actor: Concurrent Objects (Carl Hewitt at MIT)
  - Flavors (MIT), Loops (XEROX): OO-Programming in Lisp
- Programming Language Community (1980's)
  - Abstract Data Type, Encapsulation
  - Alphard, CLU, ADA

HISTORICAL BACKGROUND (3)

- Observations
  - 1970's: Age of Structured Programming
  - 1980's: Transition Age from Structured Programming to OOP
    - C ➔ C++
  - 1990's: Object Technology for WWW
    - C++ ➔ Java
  - 2000's: OOP as commodity
    - Most software products are following the OO paradigm

Objects are a new way of packaging software (1985)



SNU Internet DataBase Lab.

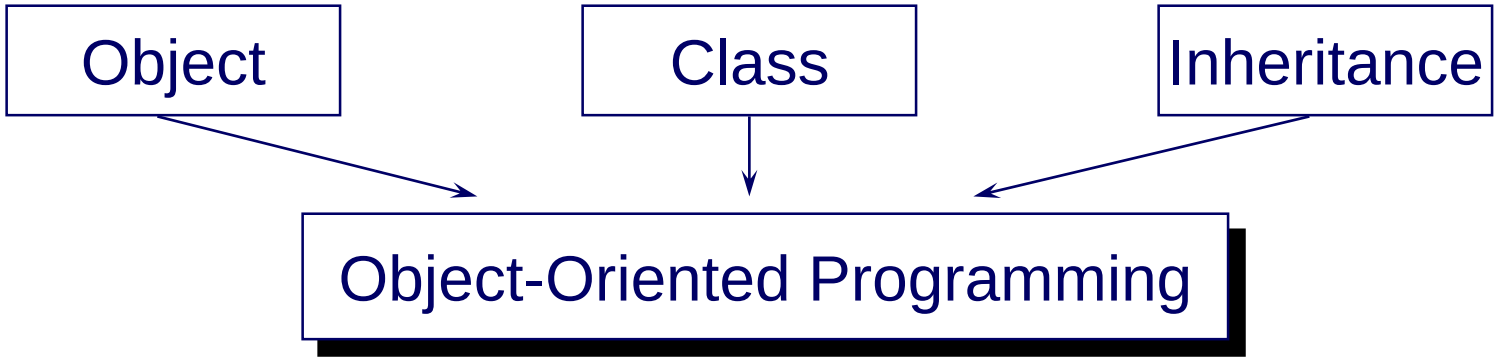
Definitions of The OO Paradigm

- Object-Oriented Technology:
  - *A way to develop* and *package* Software that draws heavily from the common experience and the manner in which real world objects relate to each other

- Object-Oriented Systems:
  - All programming languages, systems, tools and methodologies that support the Object-Oriented Technology

Peter Wegner in “Dimensions of OOPLs”

A programming style which relies on the notion of *object, class and inheritance.*



SNU Internet DataBase Lab.

Core Features of Object-Oriented Technology

1. Object
2. Message Passing
3. Class / Class Hierarchy
4. Object Identity
5. Complex object
6. Polymorphism

…….

Major Advantages of OOP

- OO provides better paradigms and tools
  - Modeling the real world correctly and easily
  - Interacting easily with a computer
  - Constructing reusable SW components and easily extensible libraries of SW modules
  - Easily modifying and extending implementations of components

- 3 focuses
  - Abstract Data Type, Inheritance, Object Identity

Roles of Object Orientation?

- *Software modeling and development (engineering) disciplines that make it easy to construct complex systems from individual components*

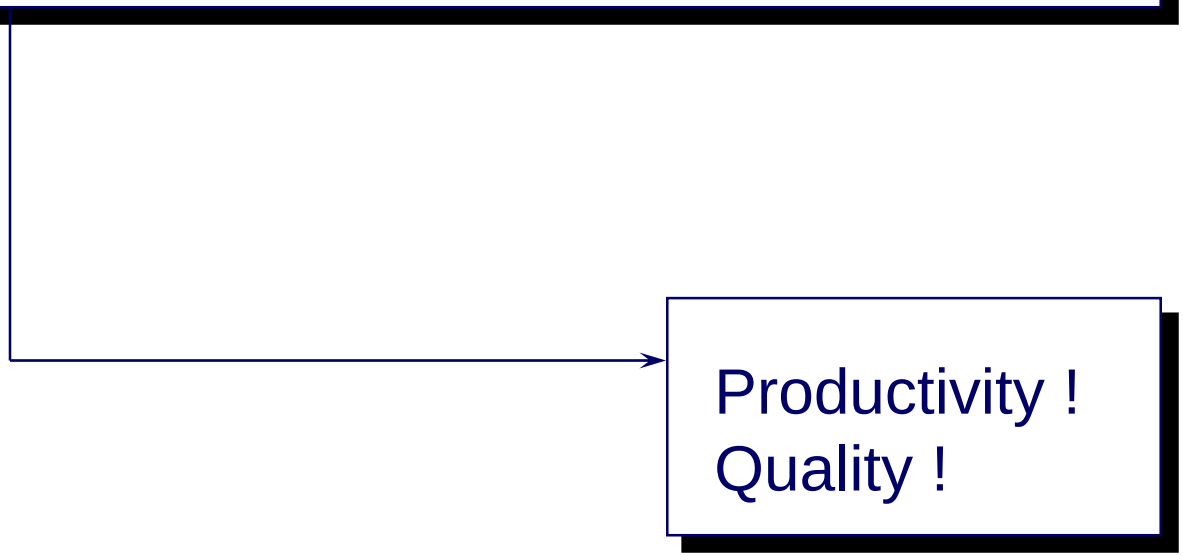
- Provide better concepts and tools to model and represent the real world ➔ modeling

- Provide better methodologies to construct complex SW systems out of modularized reusable SW units ➔ development

SNU Internet DataBase Lab.

Object-Oriented Software

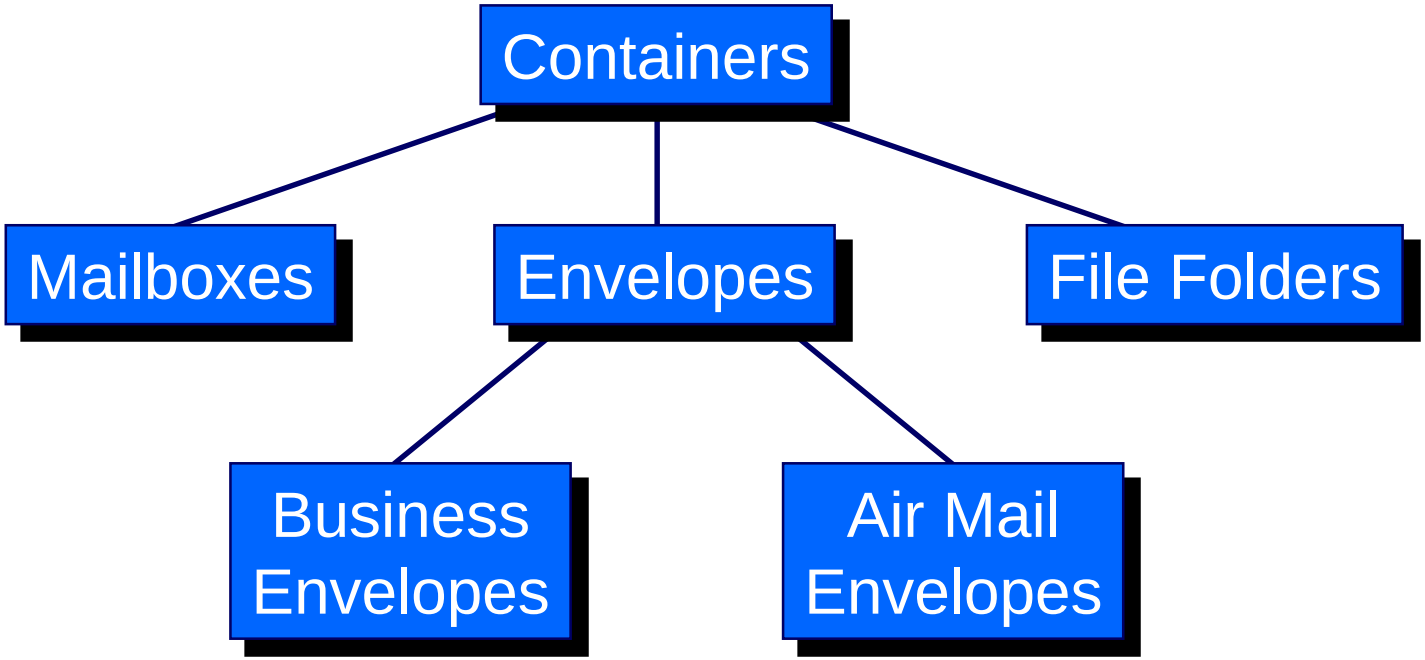
- **Can model problems better.**
- **Is more understandable.**
- **Is more repairable.**
- **Is more reusable.**



SNU Internet DataBase Lab.

Object-Oriented Systems: Class Hierarchy

- Class hierarchy of Envelope



Computation Model of OOP

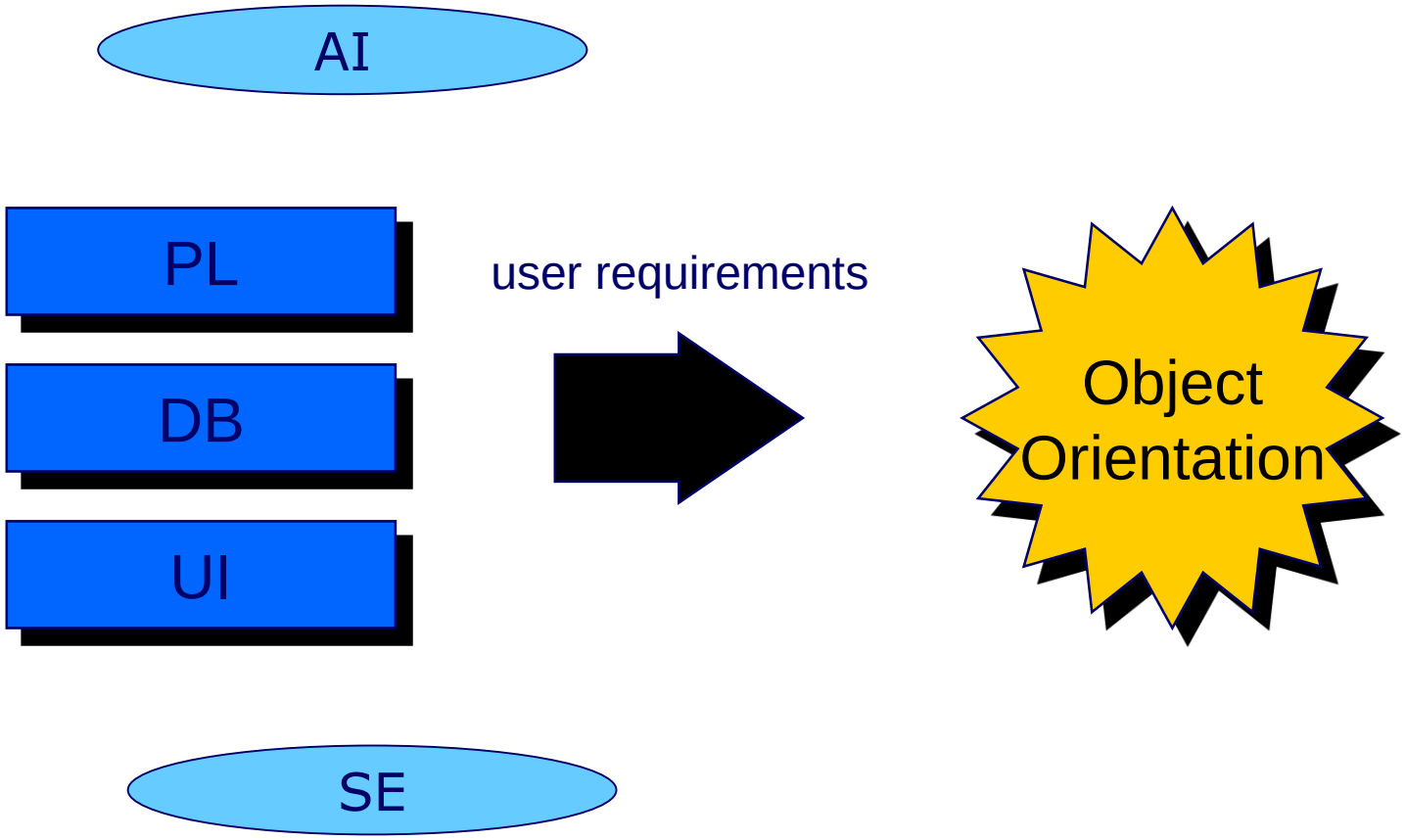
Computation

Computation is viewed as **message passing** among
a collection of self-contained, autonomous objects

Programming

Programming is a process of defining objects and message.
Objects are classified according to their behaviors.
Existing object definitions may be reused when new objects are defined.

Object Orientation everywhere



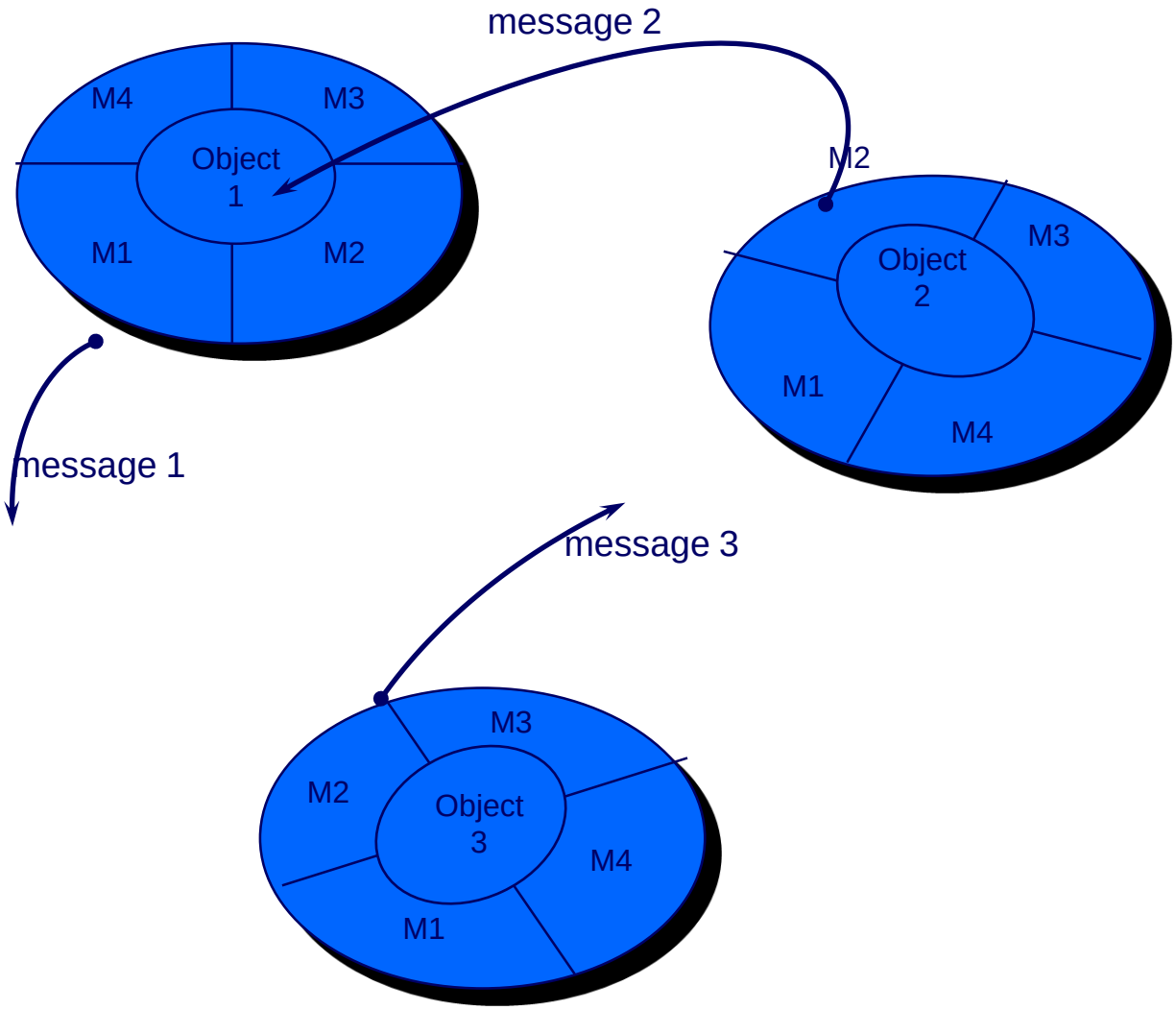
SNU Internet DataBase Lab.

Abstract Data Types (ADTs)

- ADTs extend the notion of a data type through hiding the implementation of the user-defined operations associated with the data type.

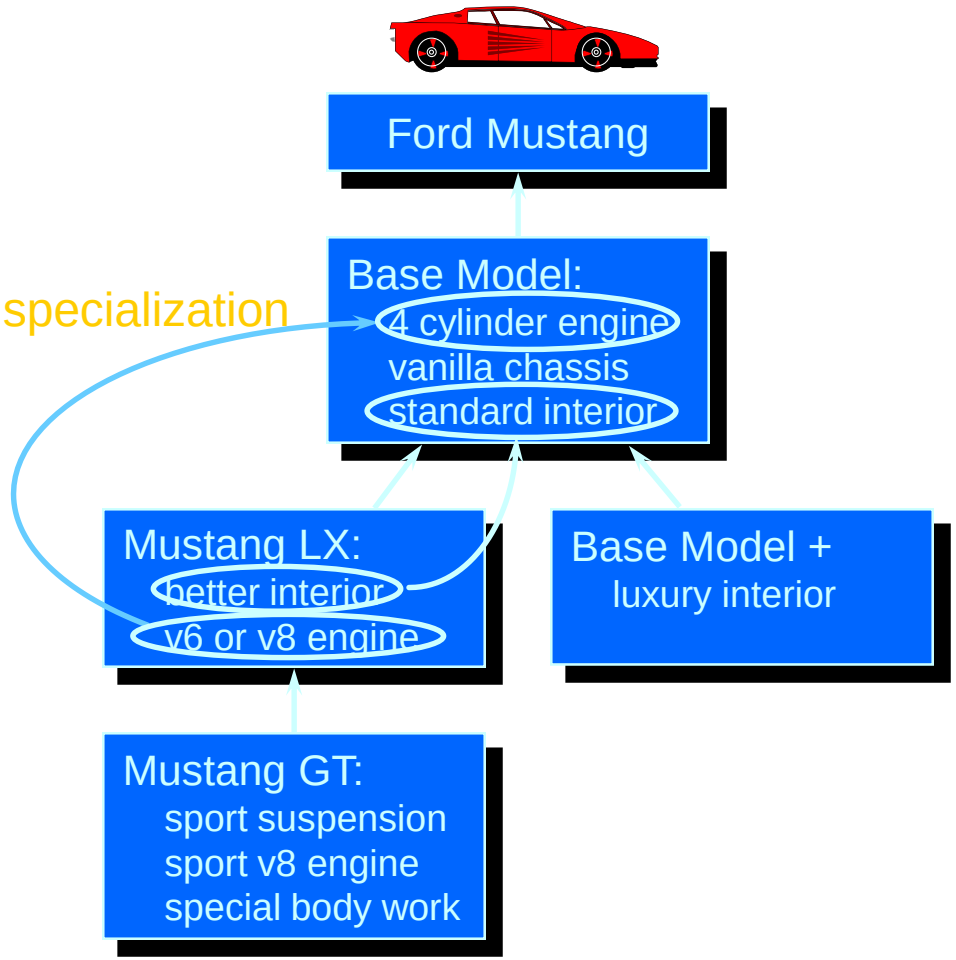
- All manipulations of instances of the data type are done exclusively through *operations* associated with the data type.

Message Passing Paradigm



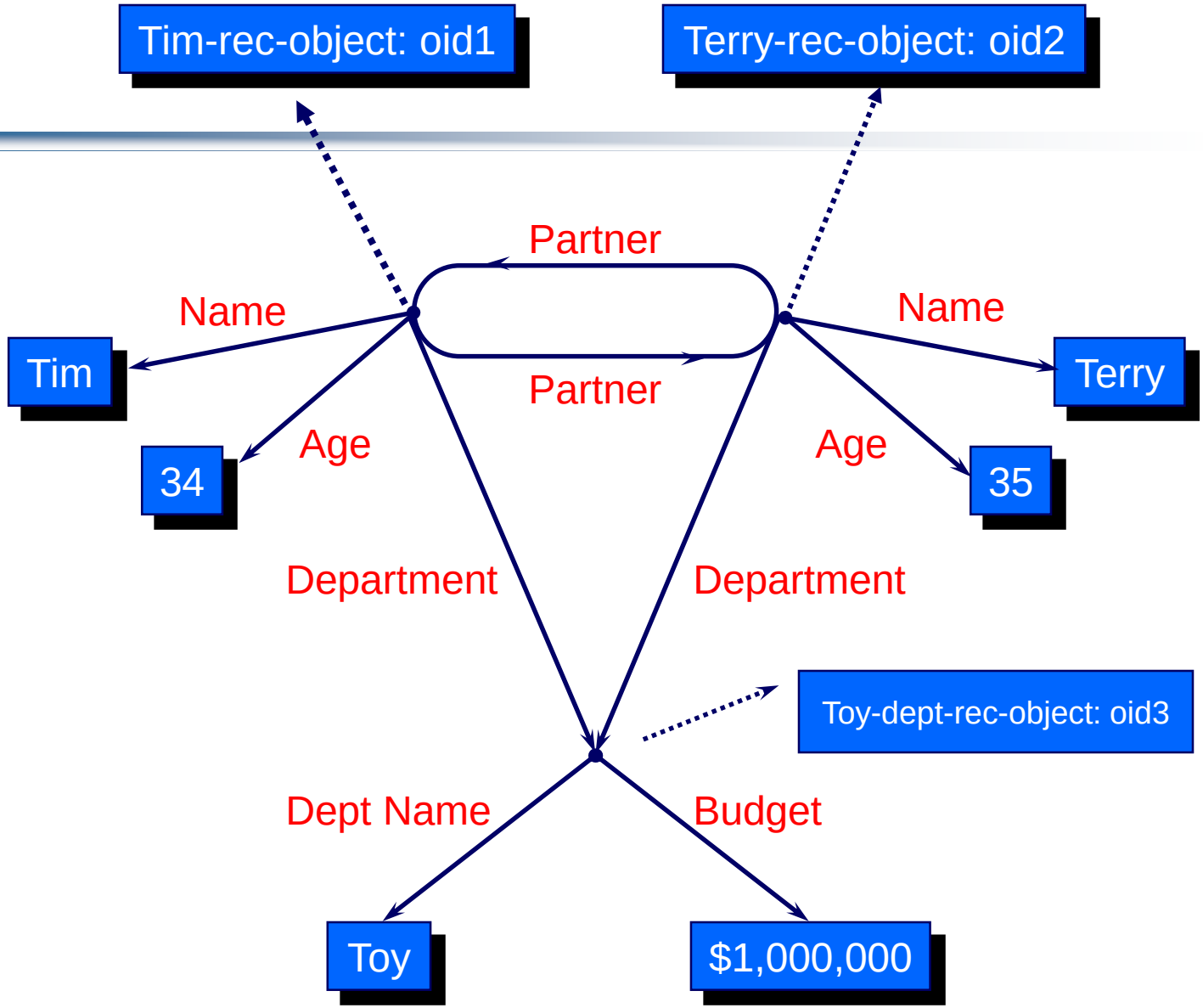
Inheritance

- *Inheriting behavior* enables *code sharing* among software modules.
- *Inheriting representation* enables *structure sharing* among data objects.

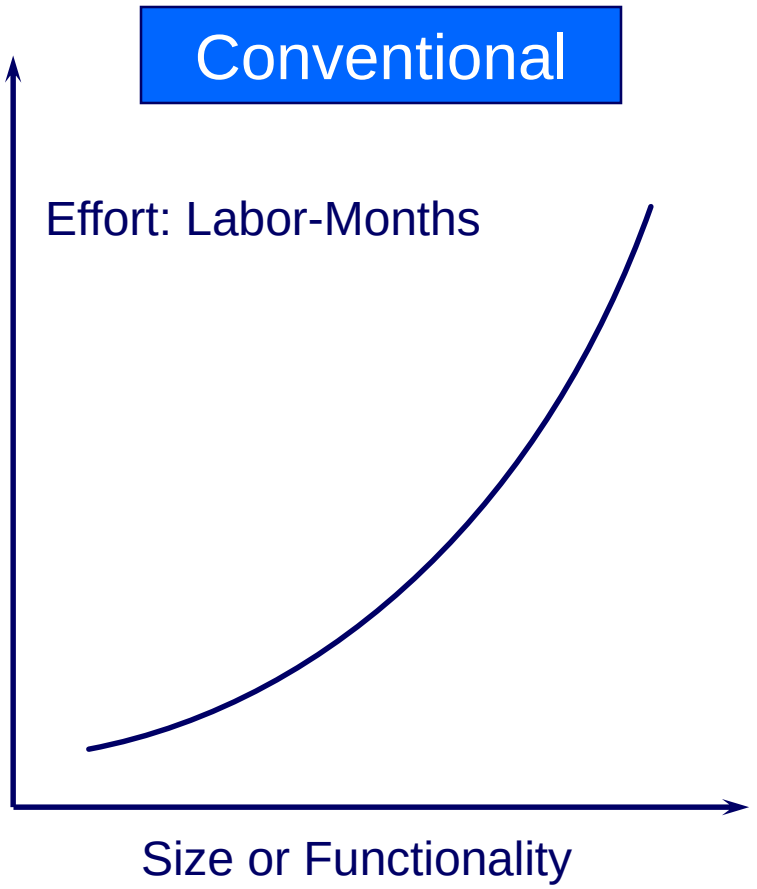


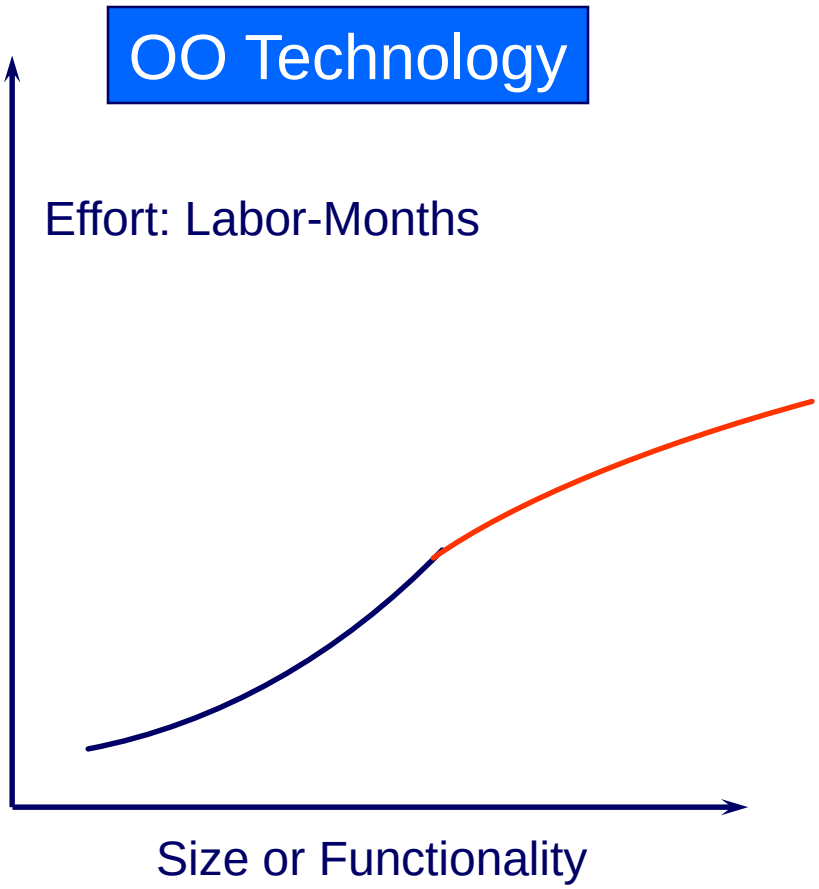
Object Identity

- *Object identity* distinguishes each object from all others.
- With *object identity*, objects can *contain or refer* to other objects.
- *Object identity* organizes objects of object space.
- *ADT* and *inheritance* organizes the classes of objects.


Tim-rec-object: oid1
Terry-rec-object: oid2
Partner
Name
Name
Tim
Partner
Terry
Age
Age
34
35
Department
Department
Toy-dept-rec-object: oid3
Dept Name
Budget
Toy
$1,000,000

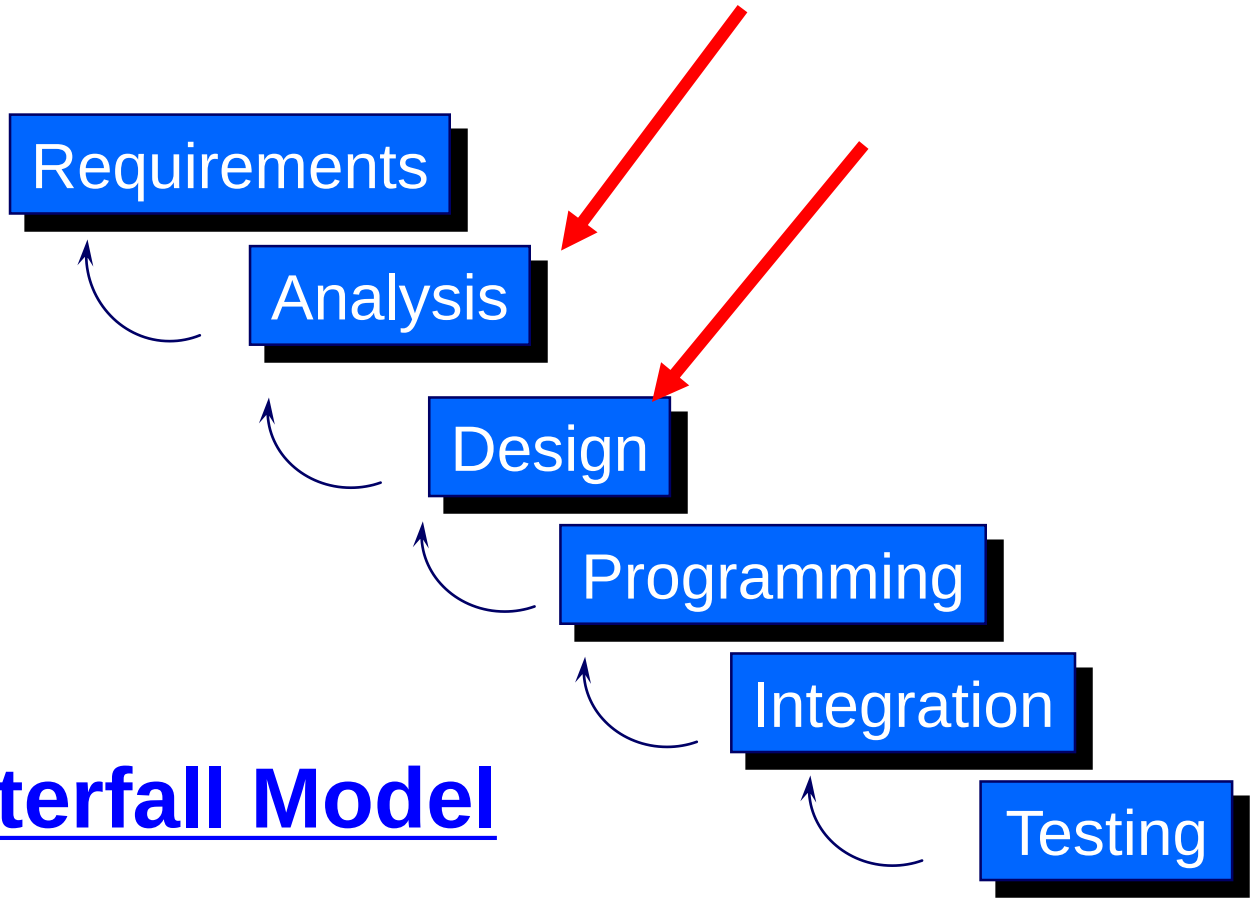
The SW Development Effort





SNU Internet DataBase Lab.

OOP from SW Engineering Viewpoint



Waterfall Model

Object-Oriented Analysis (OOA)

- Provide a detailed description of a system
- Identify WHAT
  - Identify objects, classes, operations
  - Identify object relationships, object interations
- Build real-world models using OO view of the world
- Goals of OOA
  - understand the problem domain
  - increase correctness, consistency, completeness

Object-Oriented Design (OOD)

- Provide the blueprint for implementation
- Specify HOW
  - Specify class definitions, class categories
  - Specify subsystems, system architectures
- OOA + Implementation details

- Goals of OO Design
  - Optimize maintainability, reusability, enhancibility and reliability

SUMMARY
Benefits of Object-Oriented Systems

1. Improve Productivity

2. Model Complexity Efficiently
   - The use of objects as basic modules assists the designer to model complex real-world systems

3. Designed for Change
   - The flexibility of object-oriented code allows a rapid response to changes in their requirements

4. Code Reusablility
   - The reuse of standard components reduces both the development time for new applications and the volume of code generated

5. Enhanced Maintainability
   - The increased maintainability of software makes it more reliable and reduces maintenance costs

Contents

The Evolution of OO in PL [1/4]

- *Assemblers (early 1950s)*: IBM 650, IBM 704
- *FORTRAN (mid 1950s), PL/1, COBOL (late 1950s)*
- *Algol (1960)*
  - Block structure → encapsulation
  - Algol60 (Randll and Russell): Begin-End blocks
- *Simula-67 (Dahl and Nygaard 1966)*
  - Introduced the concepts of *object / class / inheritance*
  - Intended as a simulation language
  - Laid the foundation of OO language
  - A "strongly typed" language

The Evolution of OO in PL [2/4]

- *LISP (MaCarthy 1965)*
  - Functional programming language for many AI applications
- Data Abstraction (Parnas,1972)
  - Grouping of structure and operations
  - Information hiding
  - Alphard (1976), CLU (1977)
- *Ada (late 1970, DoD)*
  - The most important PL supporting ADT

The Evolution of OO in PL [3/4]

- Smalltalk (XEROX PARC 1972)
  - One of the most influential OO language
  - Not just a language
    - A programming environment
    - A menu-based interactive user-interface
    - An initial class hierarchy
  - Not a typed language
  - Everything is an object

The Evolution of OO in PL [4/4]

- Concurrent OOPL (Primary Concern: Concurrency)
  - Actor (Hewitt 1977)
  - ABCL/1 (Yonezawa 1990)
  - Concurrnt C++ (Gehani 1988)
  - Eiffel II (Camerol 1993)
- Extensions, dialects, and versions of *Smalltalk*
  - Smalltalk/V, Smalltalk80
- OO Extentions of conventional PLs
  - C++, Objective-C, Object-Pascal
- OO Extensions of *LISP*
  - *CommonLoops, Common Lisp Object System(CLOS)*

Summary of Evolution of Programming Paradigms

- 1970s: Structured programming
- 1980s: The decade that launched the OO era
  - C → C++
- 1990s: The decade of the full-fledged OOP
  - Moving toward Internet and Web Technology
  - C++ and relatives, Java
  - Various OO extensions of existing PLs

2.2 The Evolution of Data Base Models

The Network and Hierarchical Models

- File management systems
- In the 1950s and 1960s
- Network data model
  - record types and one-to-many relationships
  - *CODASYL-DBTG:* COBOL + DB
- Hierarchical data model
  - A tree-structured hierarchical relationship among record types
  - IBM IMS

The Relational Model

- In 1970, by *Ted Codd* *"The Relational Data Model"*
- Simple and elegant
- *SQL*(Structure Query Language) by Don Chamberlain
  - Declarative
  - Specify what is to be accessed from DBs
  - Based on relational algebra and first-order predicate calculus
- Early Relational DB System (1976)
  - *System/R (IBM), INGRES (U.C. Berkeley)*
- SQL standard: *SQL92, SQL3*
- Major vendor: Oracle, *Informix*, *Sybase*, etc.

2.2 The Evolution of Data Base Models

Trials for Going Beyond RDB

- Semantic data model (early 1980)
  - Motivation: Model the real world as closely as possible
  - ER data model

- Functional data model (early 1980)
  - *Data manipulation capability*
  - Attributes are treated as functions
  - Values are retrieved through applying functions to entities

The Extended and Object-Relational Models

- *Extended* (or *object*) relational model (mid 1980)
  - Incorporated some object-oriented features as incremental extension to relational systems

- *Illustra*, *UniSQL*, *Matisse*

- *SQL3* (ISO-ANSI, 1993)
  - SQL + OO(ADT,inheritance..)

2.2 The Evolution of Data Base Models

The Nested Relational and Complex Object Models

- Nested relational model
  - Have a relation valued attribute
- Complex object model
  - More general
  - Arbitrary composition hierarchies with sets and tuples
    - tree structured object spaces

DBMS Market Trend

- 1990 이후 OODBMS 와 ORDBMS 등장
  - OODBMS: Versant, Object Store, Objectivity
  - ORDBMS: Illustra, UniSQL, Mattisse
  - 3 Big RDBMS: Oracle, Informix, Sybase
- 1997년 Major Big 3 의 ORDBMS 출시
  - Even small guys proposed ORDBMSs
- 2000년 이후: Internet and Web 지원 ORDBMS 체재

The Evolution of OO in User Interfaces

- Graphical User Interface
  - In early 1970s ➔ *mouse*
  - *Menubar, Pulldown menu, Dialog box*
  - Desktop metaphor and direct manipulation
    - *End user data object (ex. icon)*
      - data + procedure required to modify it
  - Automatizing user-interface design
    - Class hierarchy + new screen object design

- The concept of Metaphor
  - Alan Kay: Smalltalk User Interface
  - The physical metaphor is a way of saying that the visual displays of a computer system should present the images of real physical objects with some degree of abstraction
  - UI should have the physical metaphor paradigm
  - Ex. Wastebasket icon, folder icon, ...

References

- Randell, b., and Russel, L. (1964) *ALGOL 60 Implementation*
- Dahl and Nygaard (1966) *Simula - an Alogol-based simulation language*. CACM, 9
- Goldberg and Robson (1983) *Smalltalk-80:The Language and its Implementation*
- Shipman(1981) *The functional data model and the data language DAPLEX*, ACM TODS, 6(1)
- Keene(1988), *Object-Oriented Programming In COMMONLISP*
- Astrahan(1976), *System-R:A relational approach to data management*, ACM TODS, 1(2)
- Stonebraker(1976), *The design and implementation of INGRES*, ACM TODS, 1