Introduction to Advanced Object Oriented Programming

Why Object-Oriented?

*"The "software crisis" came about when people realized the major problems in software development were not algorithmic, but were caused by **communication** difficulties and the management of **complexity**" [Budd]*

The Whorfian Hypothesis:

The language in which an idea is thought or expressed colors or directs in a very emphatic manner the nature of the thought

What kind of language can alleviate difficulties with communication & complexity well?

Why Object-Oriented?
– Consider Human Growth & Concept Formation

Communication & complexity about the problem and the solution, all expressed in terms of ***concepts*** in a language!

But then, What is CONCEPT? [Martin & Odell]

Consider Human Growth & Concept Formation

stage	concepts
infant	*the world is a buzzing confusion*
very young age	*"blue" "sky" (individual concepts)* *"blue sky" (more complex concept)* *hypothesis: humans possess an innate capacity for perception*
getting older	*-> increased meaning, precision, subtlety, ...* *the sky is blue only on cloudless days* *the sky is not really blue* *it only looks blue from our planet Earth* *because of atmospheric effects* *elaborate conceptual constructs*

Concept formation: from chaos to order!

Why Object-Oriented?
- concepts and objects

So, concepts are needed to bring order into the problem and the solution

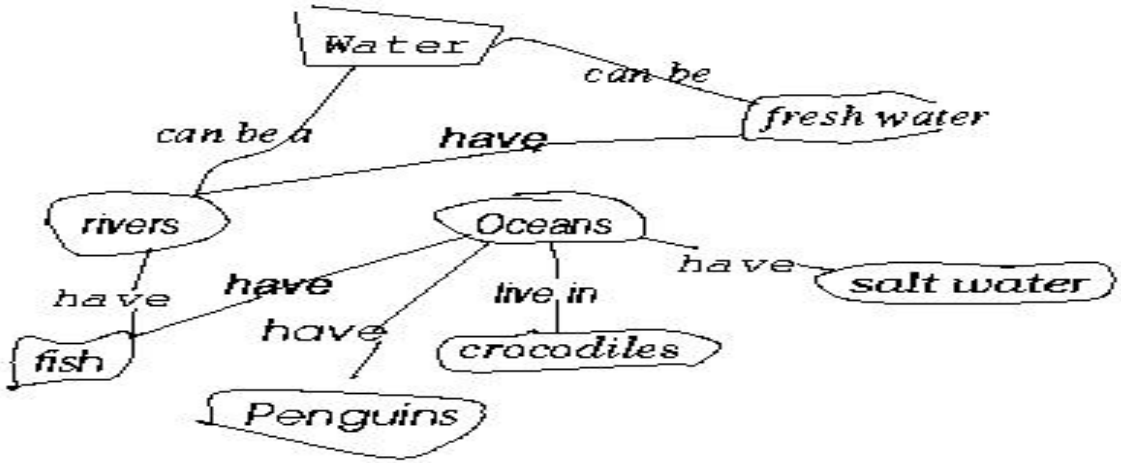
---

But, What is CONCEPT? [Martin & Odell] [Novak, 1984, Cambridge University Press]

Study of a first grade class

When given a list of concepts (water, salt water, Oceans, Penguins,…),

Harry constructed a concept diagram through which he understands his world and communicates meaning



A "concept" is an idea or notion that we apply to the things, or objects, in our awareness

Why Object-Oriented?

… for Conceptual Modeling Reasons

What kind of language is used to create this concept diagram, or Harry's mental image? What are the building blocks of Harry's perception of this piece of reality, represented in his mind/brain?

Object-Orientation (OO) = Being "conceptual"
OO analysis & design = "Conceptual" analysis & design

But for what?

for Modeling!

Analysis for Model of the problem
design for Model of the solution

*"**conceptual**" analysis for "**conceptual**" model of the problem*
*"**conceptual**" design for "**conceptual**" model of the solution*

Why Object-Oriented ->

What is a model?

A model is a simplification of reality.

E.g., a miniature bridge for a real bridge to be built

Well...sort of….but not quite

A model is *our* simplification of *our perception* of reality (that is, if it exists, otherwise it could be a mere illusion).

Your perception, my perception, his, hers, …, => about reality but about perception of reality => verification hard but needed

Your communication is not your/my/his/her validation and

A model is an *abstraction* *(omitting tremendous amount of details)* of something for the purpose of *understanding*, be it the problem or a solution.

*A **model** (like Harry's) is expressed in terms of **concepts** in a **language!***

What is Object-Orientation
- Abstraction and Encapsulation

Abstraction

Focus on the essential

Focus on what an object “is and does”

Omits tremendous amount of details

Encapsulation

a.k.a. information hiding

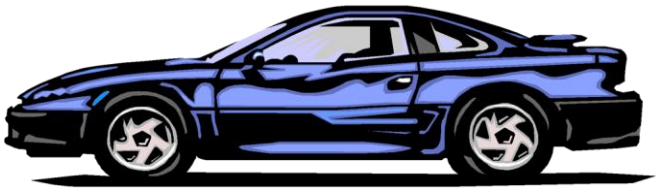
Objects encapsulate:

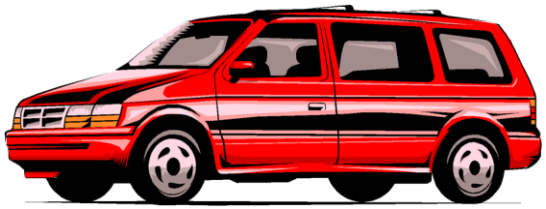
state as a collection of instance variables

behavior as a collection of methods invoked by messages

What is Object-Orientation

- Example of Abstraction and Encapsulation





Class Car

Attributes

- ❑ Model
- ❑ Location

Operations

- ❑ Start
- ❑ Accelerate

What is Object-Orientation?
- Object

What is *OBJECT*?

A "concept" is an idea or notion that we apply to the things, or objects, in our awareness

An "object" is anything to which a concept applies.

Thing drawn from the problem domain or solution space.

- E.g., a living person in the problem domain, a software component in the solution space.

A structure that

- has identity (i.e., discrete and distinguishable), and

- bundles together attributes (the data part, or state) and behavior (the function/code part).

It is an instance of a collective concept, i.e., a class.

What is Object-Orientation?

- Class

What is *CLASS*?

- a collection of objects that share common properties, attributes, behavior and semantics, in general.
- A collection of objects with the same data structure (attributes, state variables) and behavior (function/code/operations) in the solution space.
- A blueprint or definition of objects.
- A factory for instantiating objects.
- The description of a collection of related components.

Classification

- Grouping of common objects into a class

Instance.

- An object created by a class.

Instantiation.

- The act of creating an instance.

Cf. Containment.

- Objects that contain other objects as components.

What is Object-Orientation
- Subclass vs. Superclass

- Specialization

  The act of defining one class as a refinement of another.

- Subclass

  A class defined in terms of a specialization of a superclass using inheritance.

- Superclass

  A class serving as a base for inheritance in a class hierarchy

- Inheritance

  Automatic duplication of superclass attribute and behavior definitions in subclass.

What is Module?

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations.

Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application.

Software applications include many different tasks and processes that cohesively serve all paradigms within a complete business solution. Early software versions were gradually built from an original and basic level, and development teams did not yet have the ability to use prewritten code.

The introduction of modularity allowed programmers to reuse prewritten code with new applications. Modules were created and bundled with compilers, in which each module performed a business or routine operation within the program.

For example, Systems, Applications and Products in Data Processing (SAP) - an enterprise resource planning (ERP) software - is comprised of several large modules (for example, finance, supply chain and payroll, etc.), which may be implemented with little or no customization. A classic example of a module-based application is Microsoft Word, which contains modules incorporated from Microsoft Paint that help users create drawings or figures.

What is OOAD?

Analysis — understanding, finding and describing concepts in the problem domain.

Design — understanding and defining software solution/objects that represent the analysis concepts and will eventually be implemented in code.

OOAD — Analysis is object-oriented and design is object-oriented. A software development approach that emphasizes a logical solution based on objects.

What is Analysis

- Emphasis an investigation of the problem and requirements, rather than a solution.

What is Design

- Emphasizes a conceptual solution that fulfills the requirements rather its implementation

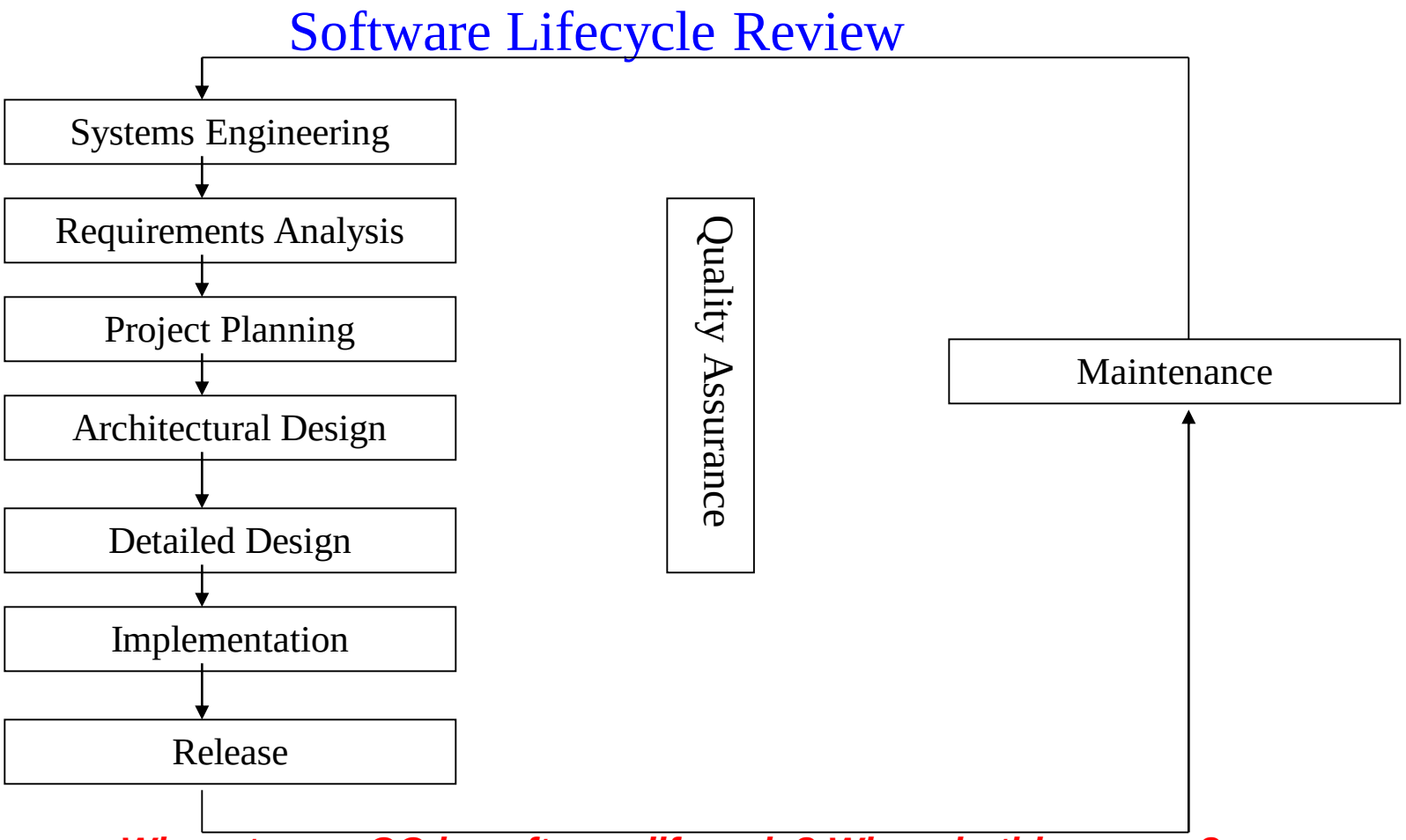
Do the right thing(analysis), and do the thing right(design). The purpose of Analysis and Design:

Transform the requirements into a system design.

- What is Object-Oriented Analysis?
  - Finding and describing the concepts in the problem domain.
  - For example:
    - Flight Information System, some of the concepts: **Plane**, **Flight**, and **Pilot**.
    - Auto Rental Company, some of the concepts: **Vehicle**, **Customer**, and **Reservation**.
    - Payroll System, some of the concepts: **Employee**, **PayCheck**, and **Timecard**.
    - Course Registration System, some of the concepts: **Student**, **Course**, **Registration**, and **Professor**.

- What is Object Oriented Design?
  - defining software objects, their responsibilities and how they collaborate to fulfill the requirements.
  - For Example:
    - **Plane** software object may have a **getFlightHistory** operation.
    - **Vehicle** software object may have a **getStatus** operation.
    - **Employee** software object may have a **payCalculate** operation.
    - **Student** software object may have a **hasPrerequisites** operation.
- A critical ability in OO design is to skillfully assign responsibilities to software objects.

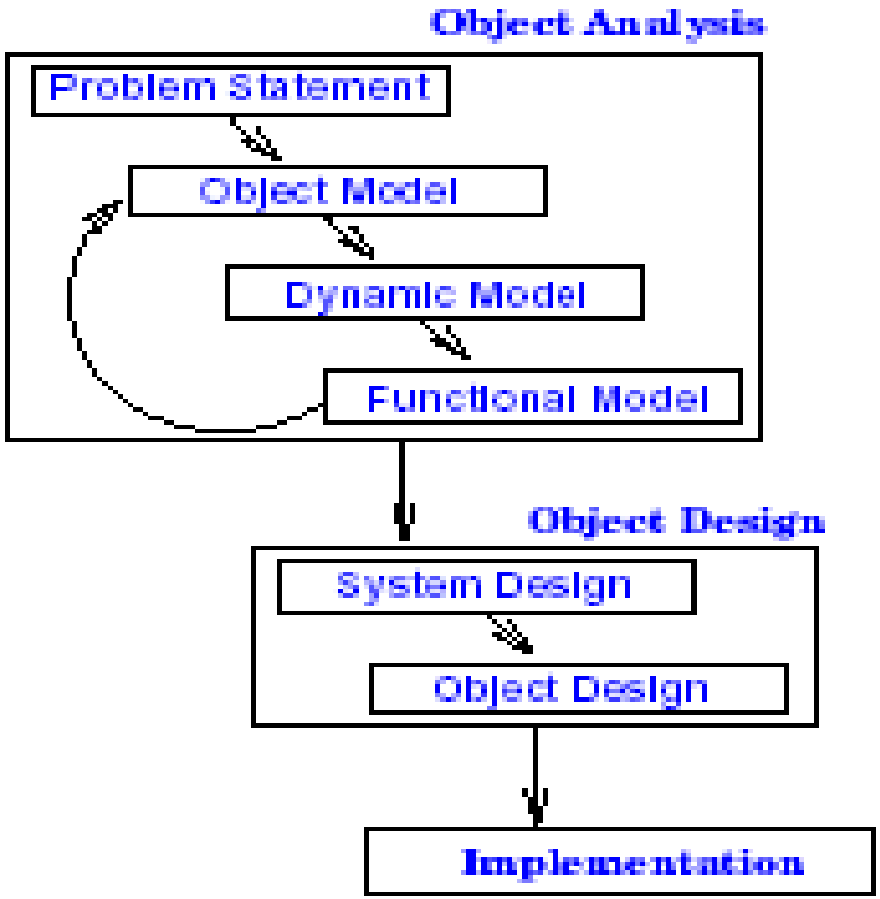
How to Do OOAD
– Where to Use OO?



Where to use OO in software lifecycle? Where in this course?

How to Do OOAD
– OMT as Object-Oriented Methodology

- OMT Methodology



Object Model: describes the static structure of the objects in the system and their relationships
-> Object Diagrams.

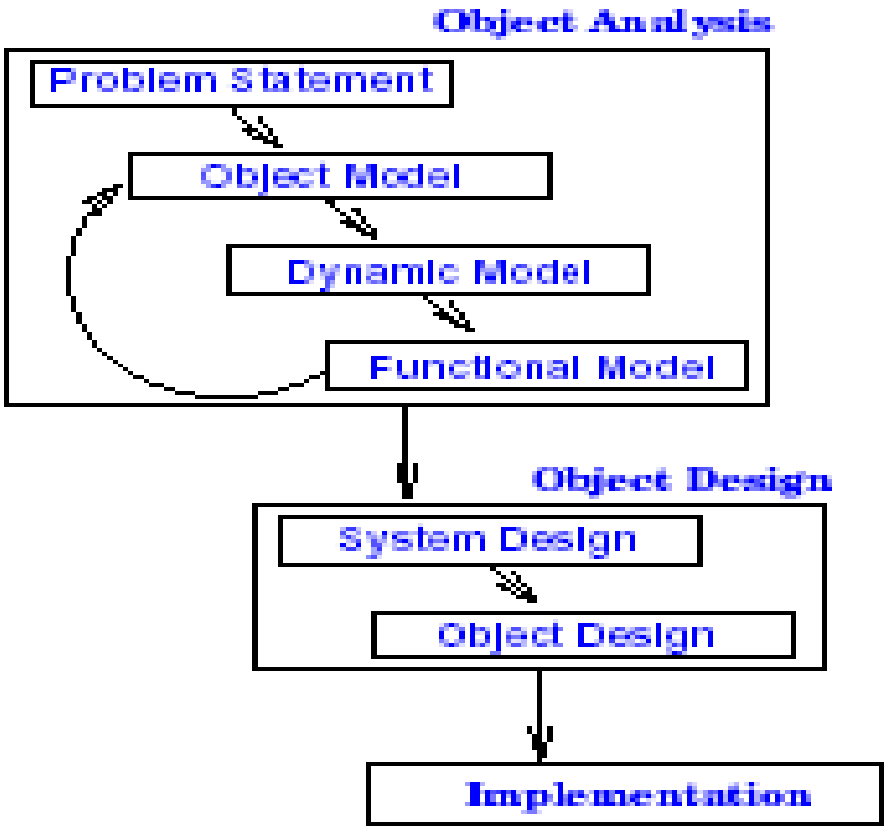
Dynamic Model: describes the interactions among objects in the system
-> State Diagrams.

Functional Model: describes the data transformation of the system
-> DataFlow Diagrams.

How to Do OOAD
– OMT as Object-Oriented Methodology

OMT (Object Modeling Technique) by James Rumbaugh

- OMT Methodology



Analysis: Model the real world showing its important properties; Concise model of what the system will do

System Design: Organize into subsystems based on analysis structure and propose architecture

Object Design: Based on analysis model but with implementation details; Focus on data structures and algorithms to implement each class; Computer and domain objects

Implementation: Translate the object classes and relationships into a programming language

Why Object-Oriented
– Why Shift in Modeling Paradigm

From Functional

- Functions as building blocks (fn: Input -> Output)
- Algorithmic perspective
- E.g., Lisp

To Object-Oriented

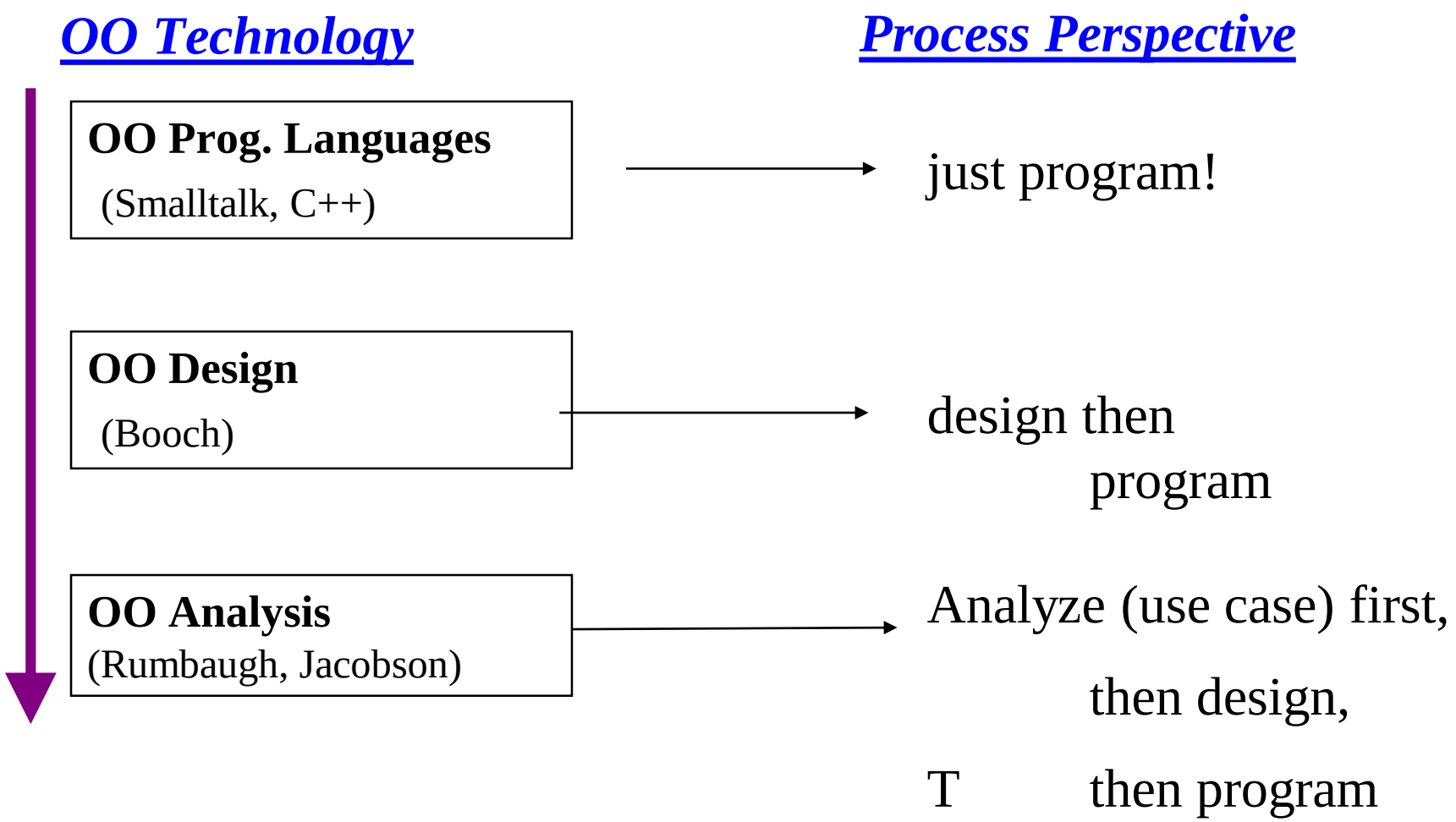
- Objects as building blocks.
- Conceptual perspective

  from the vocabulary of the problem space for analysis

  from the vocabulary of the solution space for design

How to Do OOAD

- Historical Perspective



Why Object-Oriented
- Programming Language Perspective

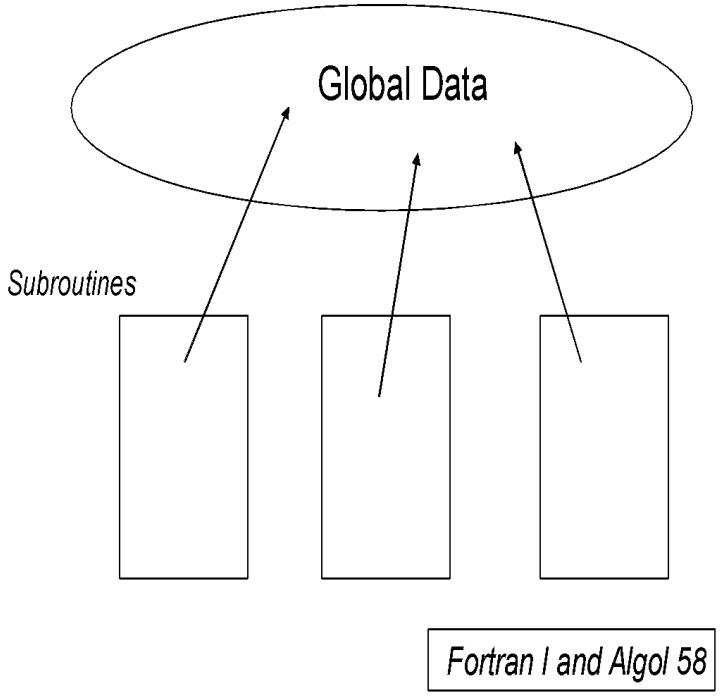
First Generation (1954-1958)
- Fortran I

Second Generation (1959-1961)
- Fortran II, Algol, Cobol

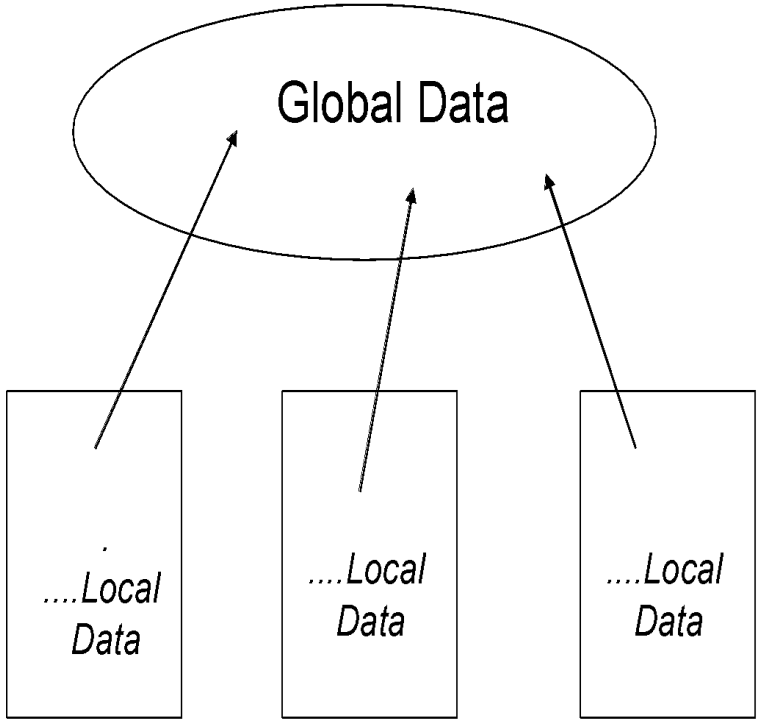
Third Generation (1962-1970)
- PL/I, Pascal

Object-Oriented Languages
- Smalltalk, C++, Java

1st Generation

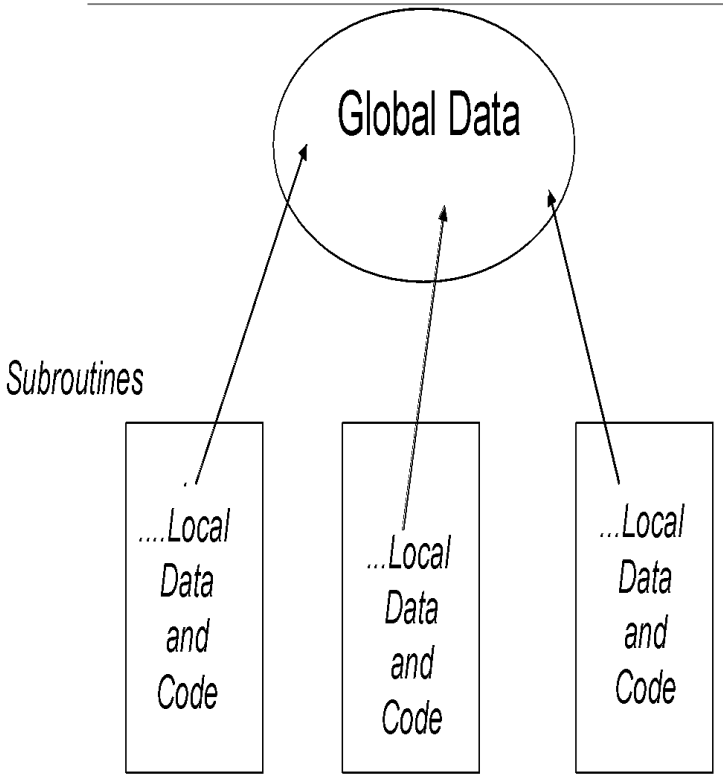


2nd Gen



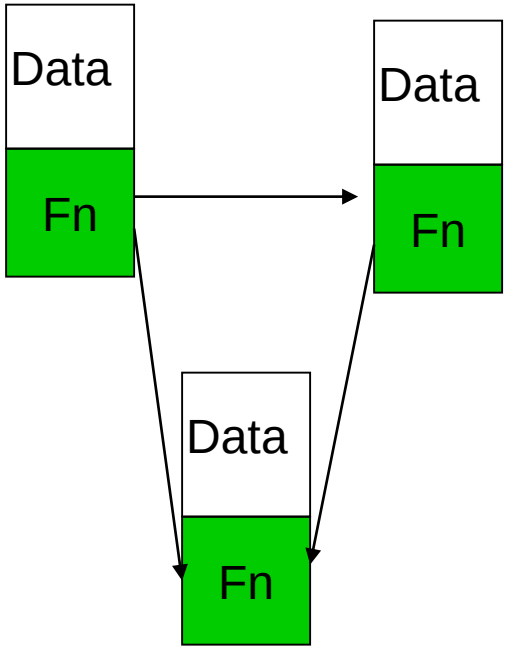
Fortran II, Algol, Cobol

3rd Generation



PL/I, Pascal

Object Oriented



Allocation of functionality to objects
Shift from monolithic to decentralized control
Object-Oriented Programming Languages (OOPL)

What is Object-Orientation
- Abstract Class vs. Concrete Class

Abstract Class.

- An *incomplete* superclass that defines common parts.
- Not instantiated.

Concrete class.

- Is a *complete* class.
- Describes a concept completely.
- Is intended to be instantiated.