Introduction to Software Engineering (contd.)

Dealing with Complexity

1. Abstraction
2. Decomposition
3. Hierarchy

2. Decomposition

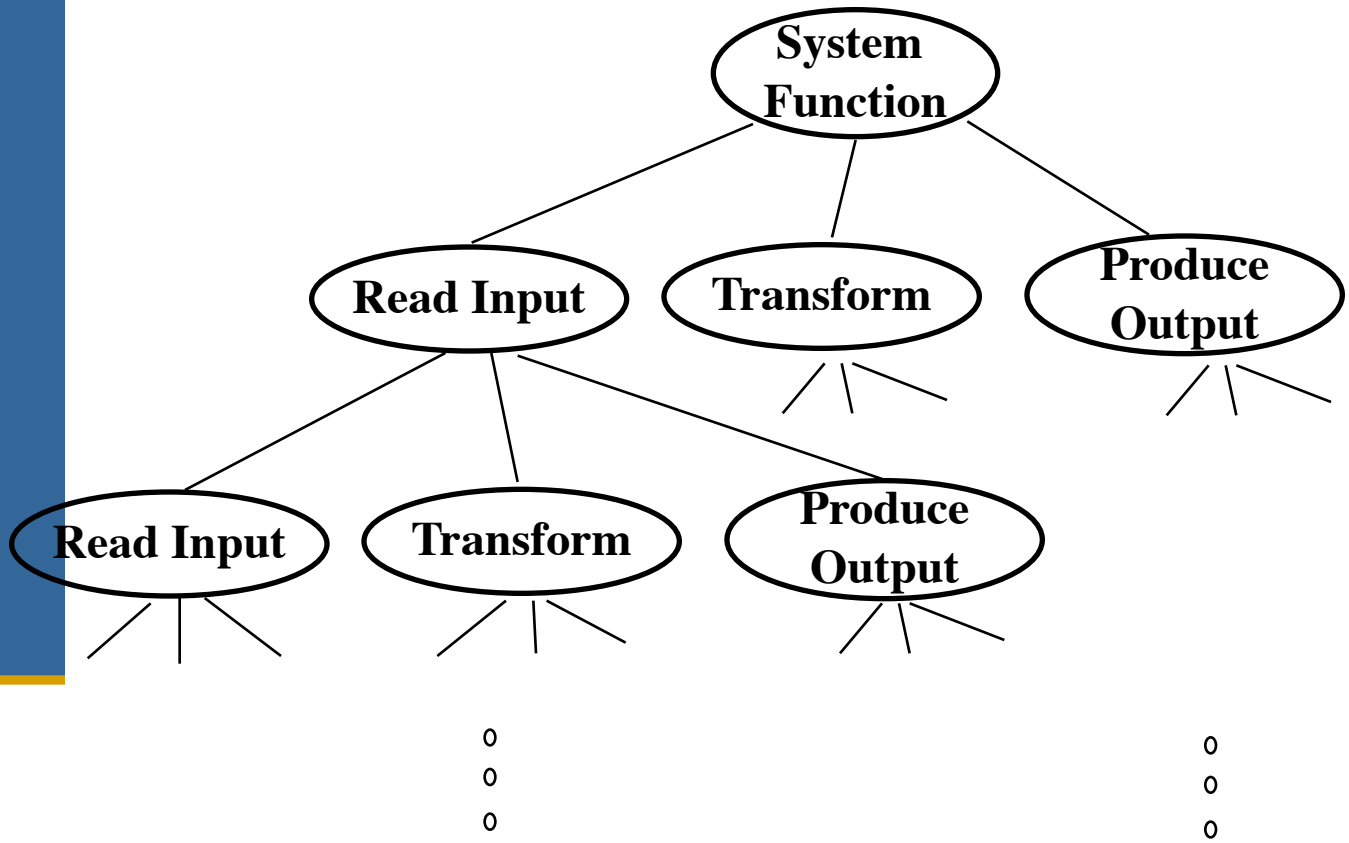
- A technique used to master complexity ("divide & conquer")

- Functional decomposition
  - The system is decomposed into modules
  - Each module is a major processing step (function) in the application domain
  - Modules can be decomposed into smaller modules

Which decomposition is the right one?

- Object-oriented decomposition
  - The system is decomposed into classes ("objects")
  - Each class is a major abstraction in the application domain
  - Classes can be decomposed into smaller classes

Functional Decomposition



Top Level functions

Level 1 functions

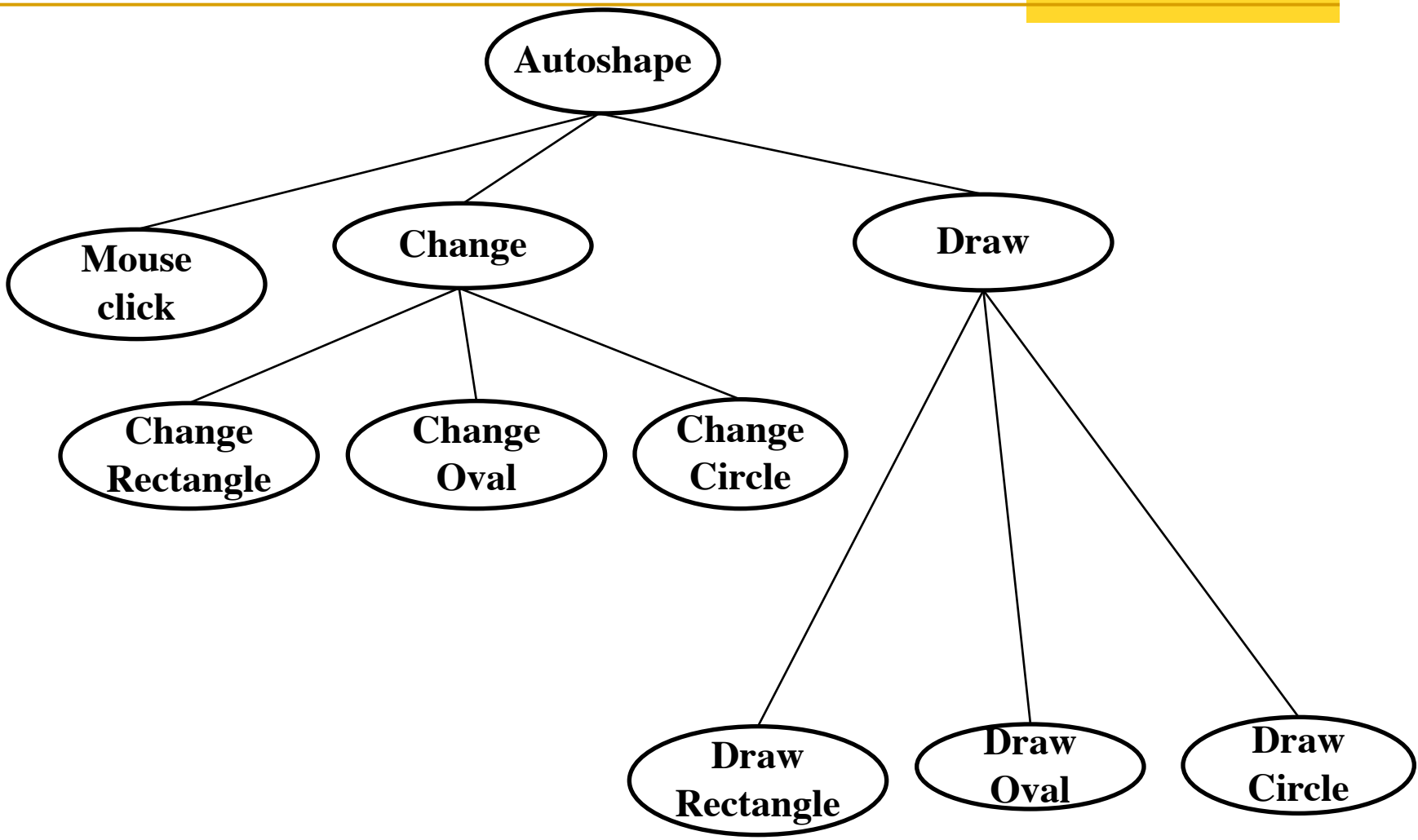
Level 2 functions

Machine Instructions

Functional Decomposition

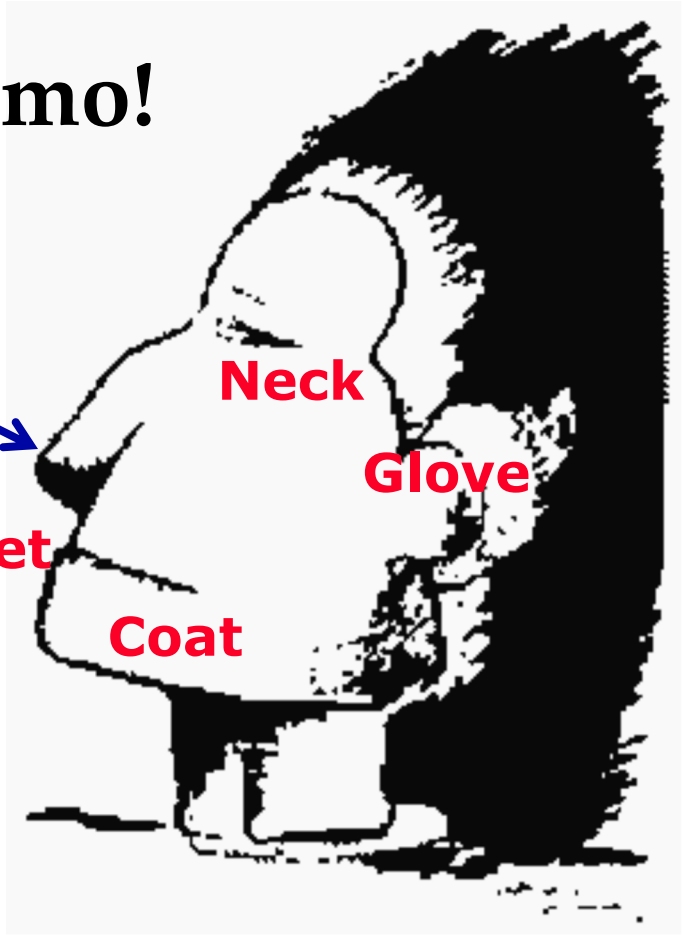
- Functionality is spread all over the system
- Maintainer must understand the whole system to make a single change to the system
- Consequence:
  - Code is hard to understand
  - Code that is complex and impossible to maintain
  - User interface is often awkward and non-intuitive
- Example: Microsoft Powerpoint's Autoshapes

Functional Decomposition: Autoshape



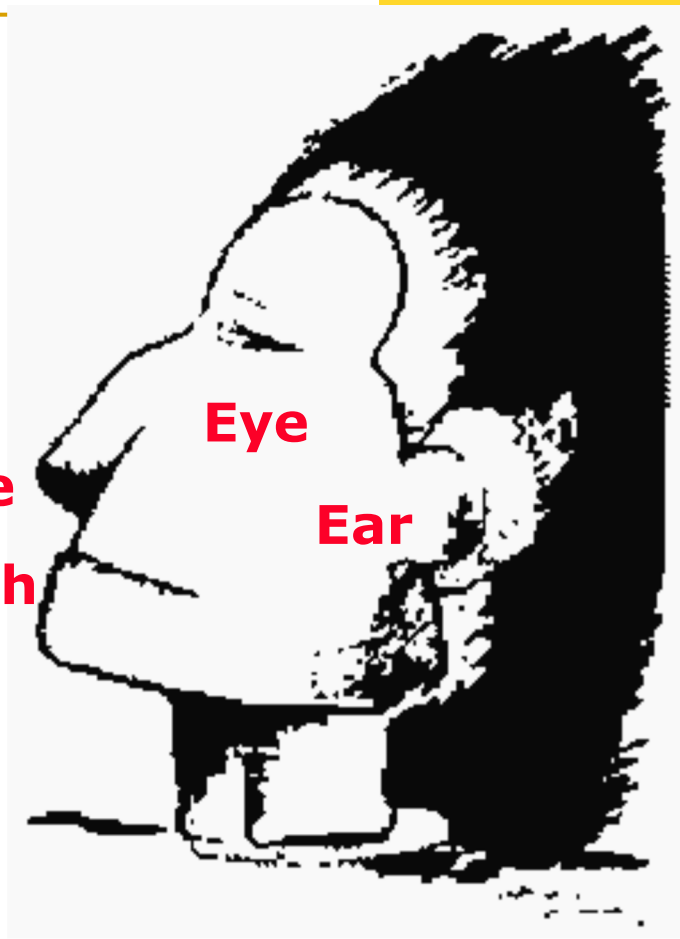
What is This?

What is This?



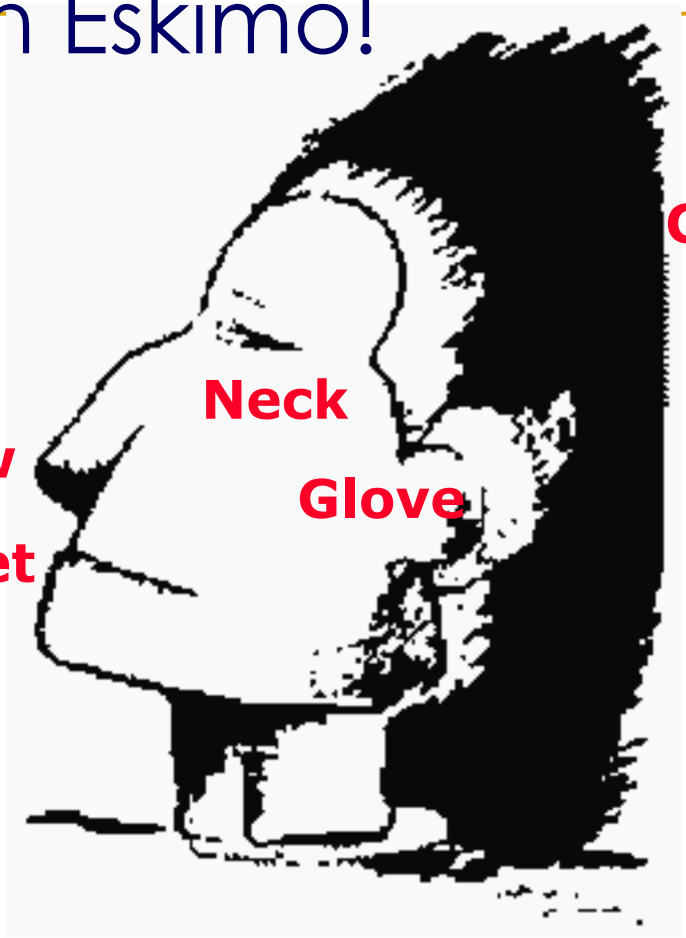
What is This?

A Face!

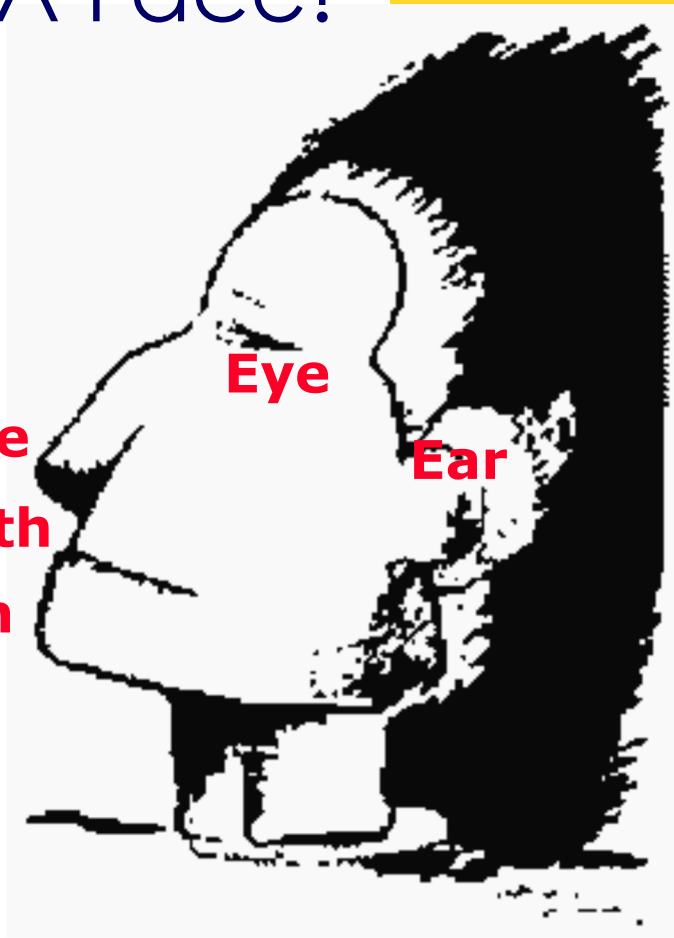


What is This?

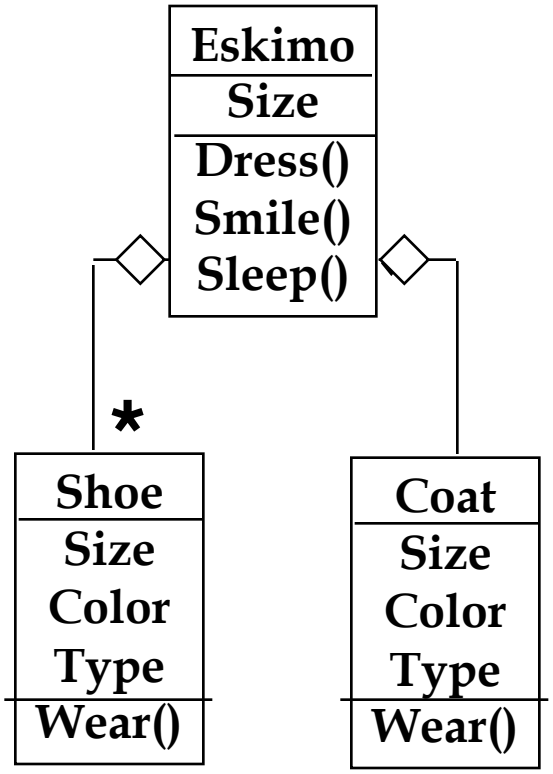
An Eskimo!



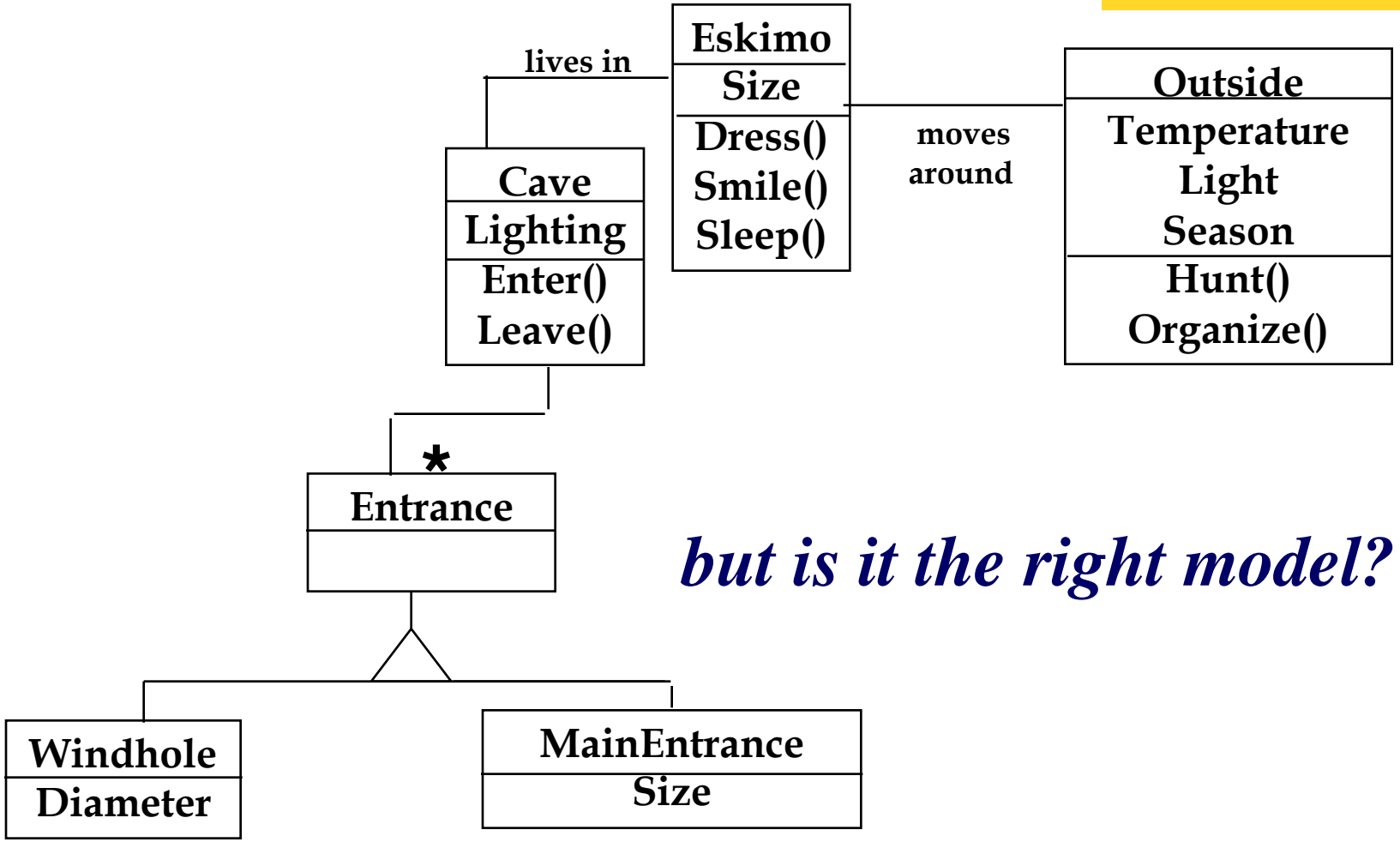
A Face!



Model of an Eskimo

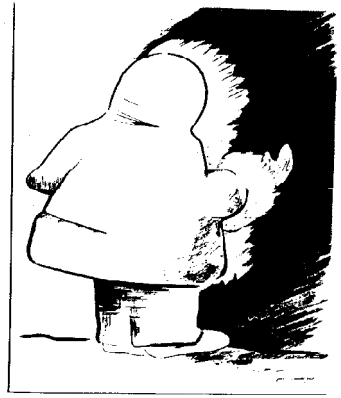


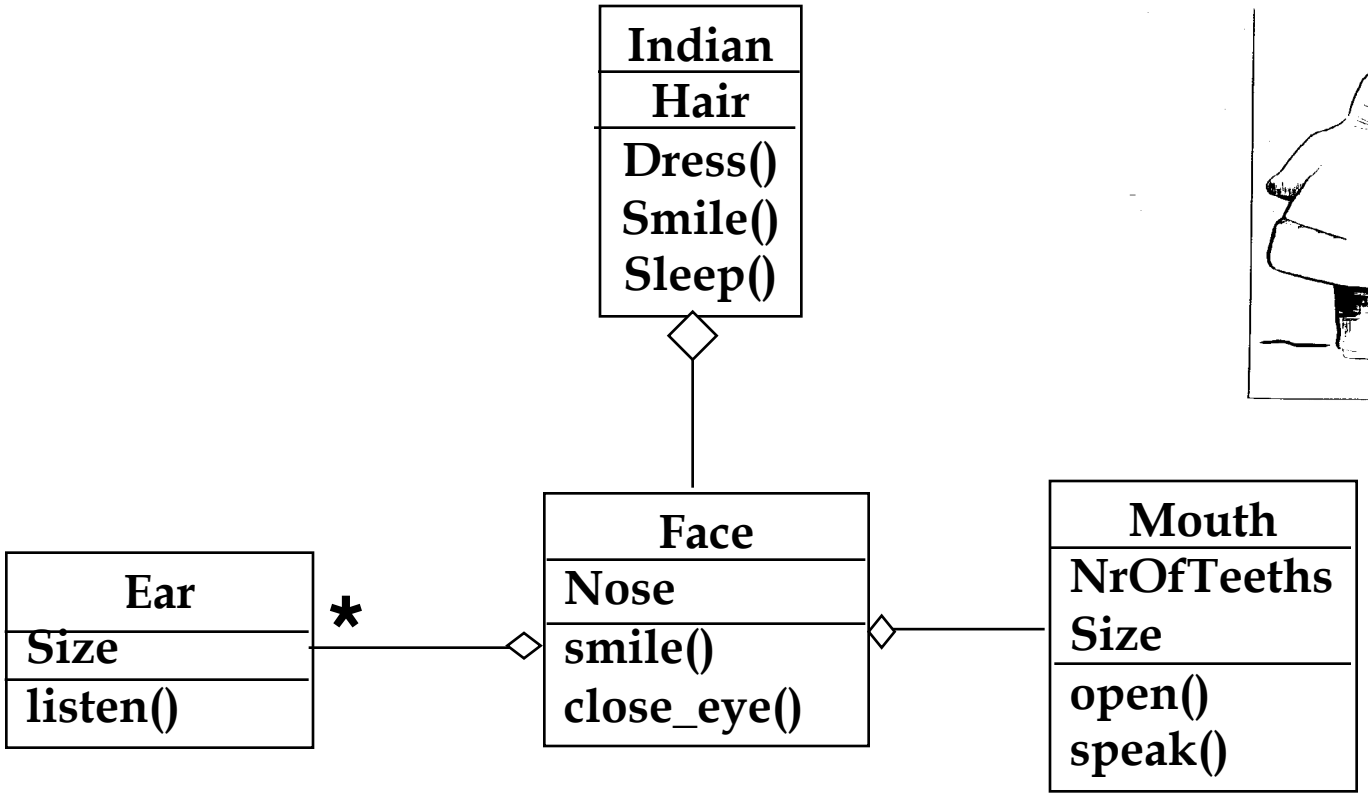
Iterative Modeling then leads to ....



but is it the right model?

Alternative Model: Head of an Indian





Class Identification

- Class identification is crucial to object-oriented modeling
- Basic assumption:
  1. We can find the classes for a new software system - called *Greenfield Engineering*
  2. We can identify the classes in an existing system - called *Reengineering*
  3. We can create a class-based interface to any system - called *Interface Engineering*

Background on Object Technology

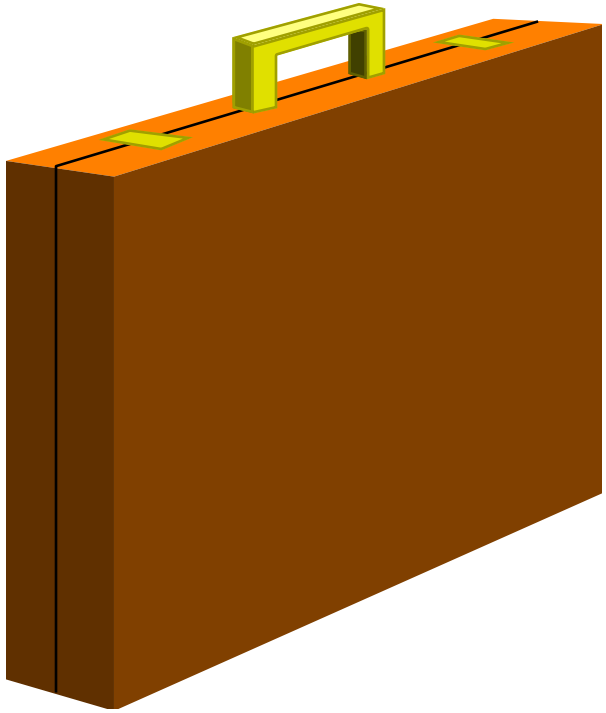
- What is a Methodology ?
  - process for organized production of systems/software
  - using a collection of pre-defined techniques and notational conventions
- What is an Object-Oriented Methodology?
  - A development approach that organizes a system as a collection of objects containing both data and behavior

Major Concepts of Object-Oriented Analysis and Design

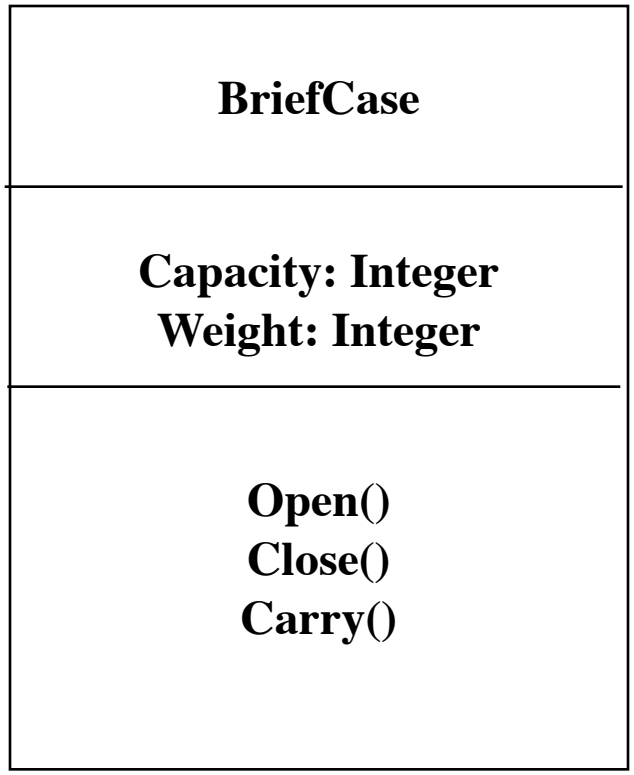
What is an Object?

- An object has structure ~ attributes
- An object must be an entity ~ a thing that can have properties and not be a property itself.
- An object has behavior
- An object has unique identity
- An object is generally stated as a noun
- For example : Thermometer is an object, temperature is not an object it is a property (attribute) of the thermometer

What is this Thing?



Modeling a Briefcase



A new Use for a Briefcase

BriefCase
Capacity: Integer Weight: Integer
Open() Close() Carry() SitOnIt()

Questions

- Why did we model the thing as “Briefcase”?
- Why did we not model it as a chair?
- What do we do if the SitOnIt() operation is the most frequently used operation?
- The briefcase is only used for sitting on it. It is never opened nor closed.
  - Is it a “Chair”or a “Briefcase”?
- How long shall we live with our modeling mistake?

Exercise 1

- Identify the objects likely to be encountered in the following systems/domain:
  - A Convertible Car
  - An Airline
  - A Computer network

Exercise 1 Sample Objects

Convertible	Airline	Computer Network
Engine	Airplane	File
Chasis	Terminal	Protocol
Steering Wheel	Baggage Claim	Server
Brake	Schedule	Workstation
Accelerator	Ticket	Cable
Radio	Reservation	Port
Tire	Pilot	Printer
Auto Light	Flight Attendant	Disk
Windshield	Passenger	Process
Wiper	Flight Plan	Test Equipment
	Stock Holder	Access Priviledge
	Gate	

OO Concepts

- What is a Class ?
  - A group of objects with
    - similar properties (attributes)
    - common behavior (operations)
    - common relationships to other objects (associations)
    - common semantics

- What is the difference between a Class and an Object ?

Exercise 2: Class Interpretation

- What classes would you create for the following lists of objects?
  - 747, Lear jet, twin engine plane, stealth bomber
  - laser printer, dot matrix printer, ink jet printer, fax machine, photocopier
  - Fog-light, headlight, blinker, brake light, back-up light, turn signal

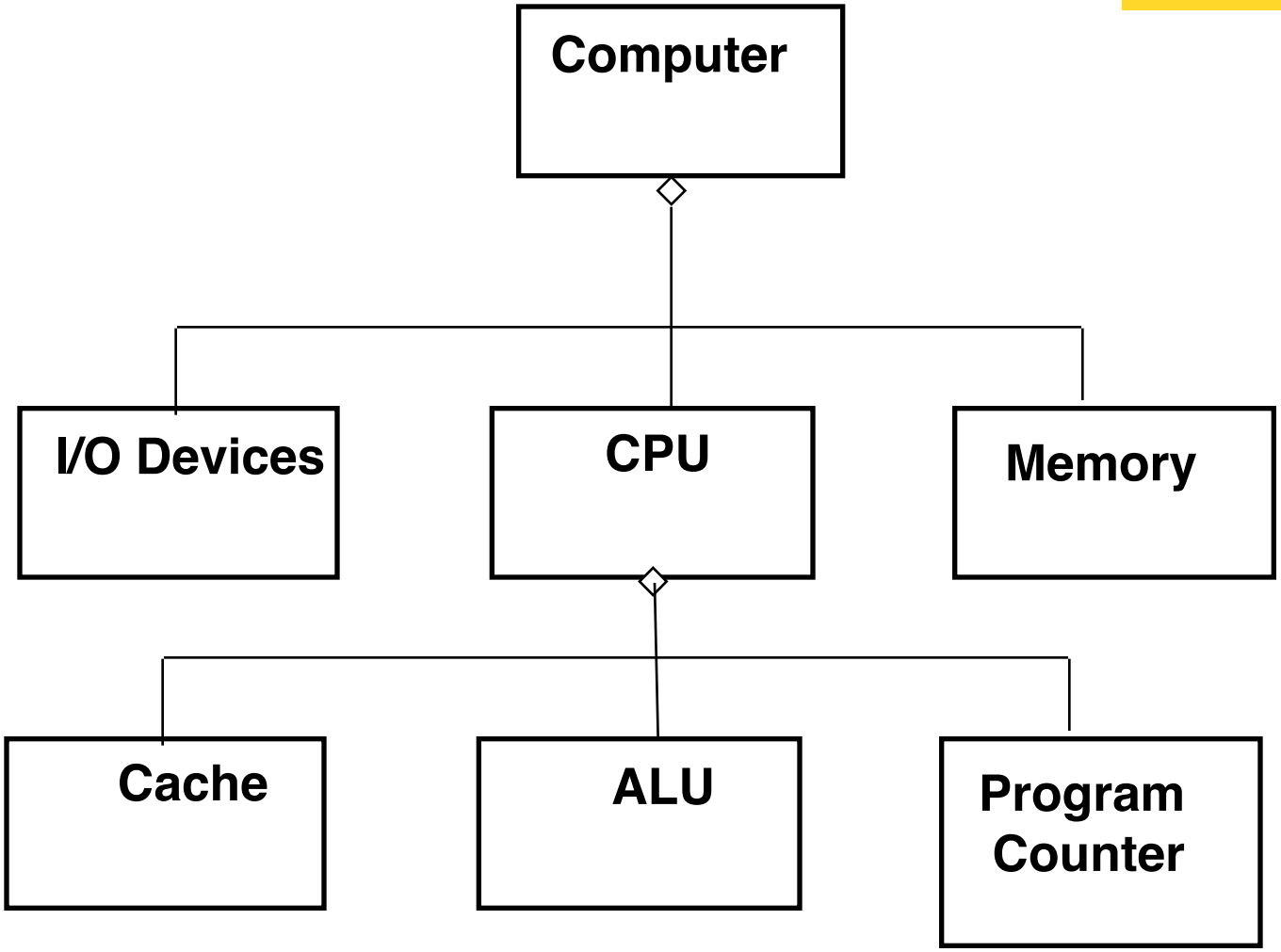
Exercise 2: Suggested Classes

- Airplane
- Printer - Things that Jam
- Light, Auto Light, Motor Vehicle Light

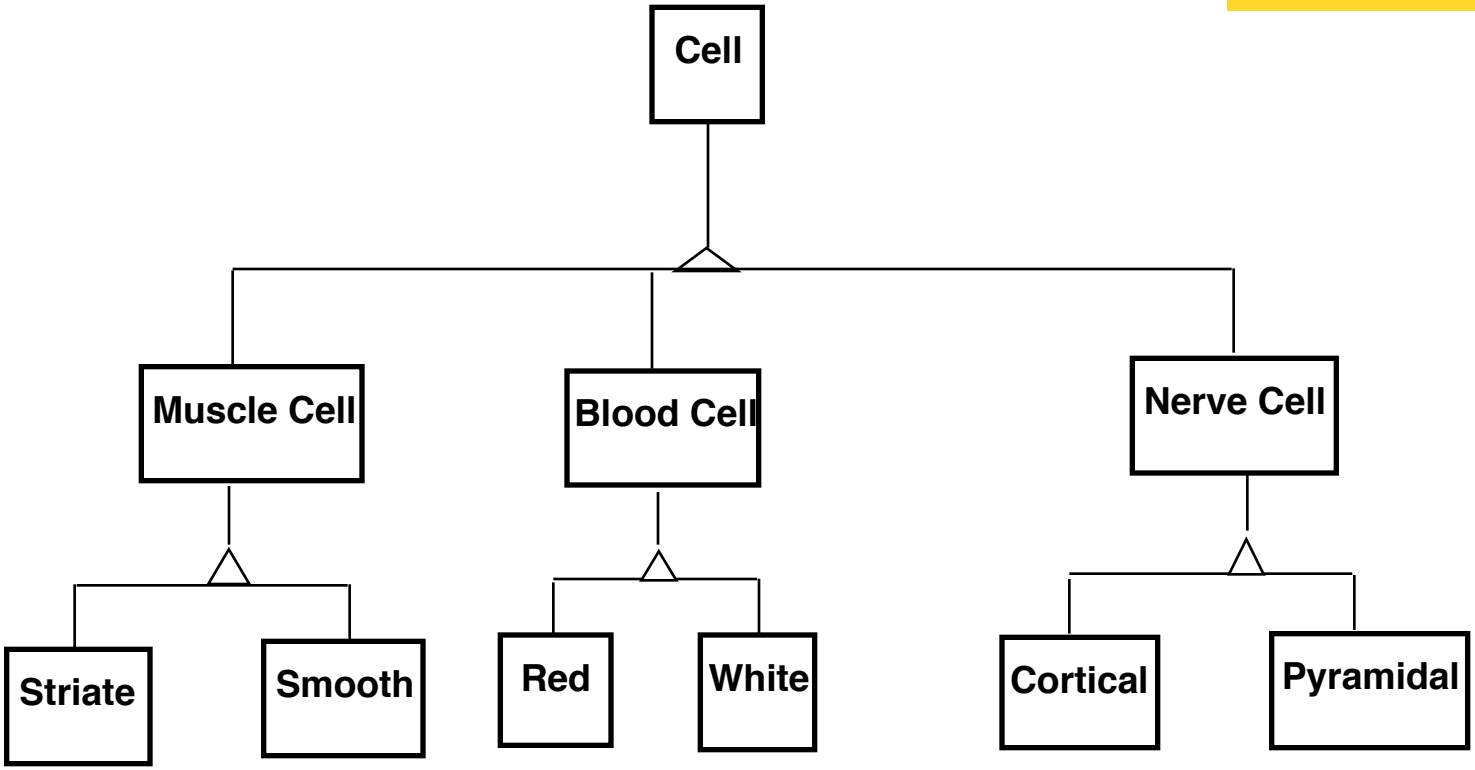
3. Hierarchy

- We got abstractions and decomposition
  - This leads us to chunks (classes, objects) which we view with object model
- Another way to deal with complexity is to provide simple relationships between the chunks
- One of the most important relationships is hierarchy
- 2 important hierarchies
  - "Part of" hierarchy
  - "Is-kind-of" hierarchy

Part-of Hierarchy



Is-Kind-of Hierarchy (Taxonomy)



So where are we right now?

- Three ways to deal with complexity:
  - Abstraction, Decomposition, Hierarchy
- Object-oriented decomposition is a good methodology
  - Depending on the purpose of the system, different objects can be found
- How can we do it right?
  - Many different possibilities
  - Our current approach: Start with a description of the functionality (Use case model), then proceed to the object model
  - This leads us to software lifecycles