Classification

Object Orientated Analysis and Design

Benjamin Kenwright

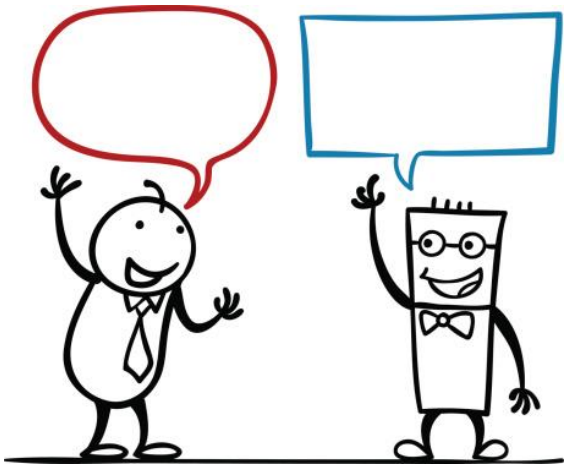
Submissions

- Every group member
  - Submitted Group GitHub Repository Address
- Demonstrated this week
  - Ability to use GitHub
  - Get comfortable with Version Control
  - Add/Remove Files/Documents
    - `Team' Exercise

Discussion Activity

- Explain in your own words why we use version control?

  (5 Minutes)



Outline

- Review
- What do we mean by Classification?
- Levels of Abstraction
- Identifying Mechanisms
- Summary/Discussion
- Conclusion

- Coursework

Revision Question

- An object is an entity that has:

a) state, action, dependencies

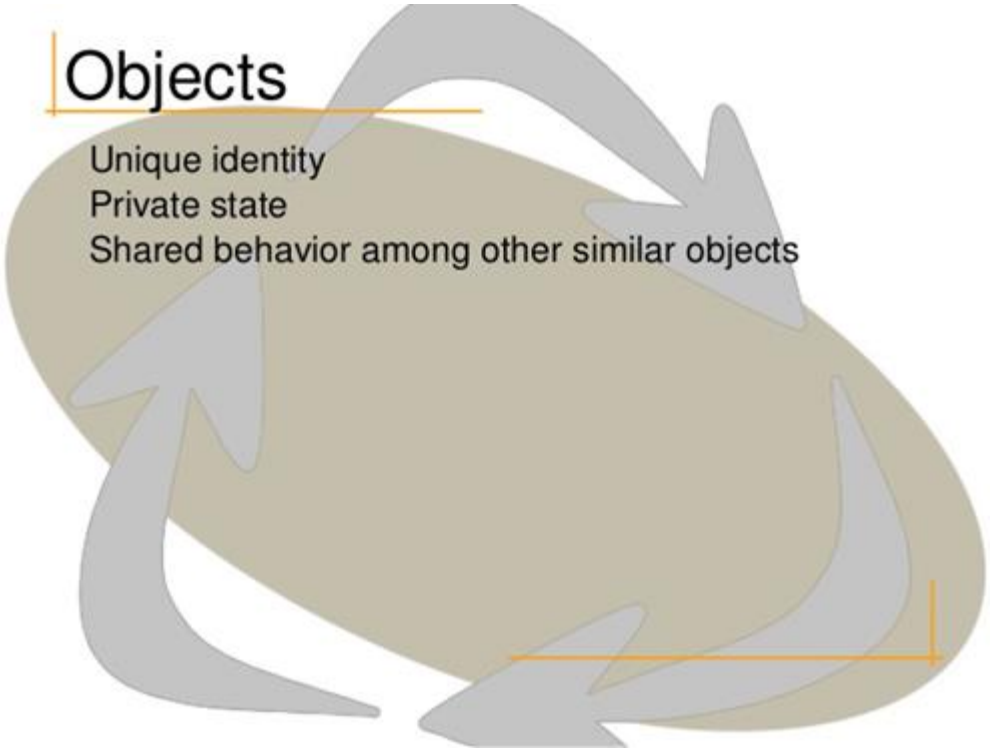
b) behaviour, action, state

c) class, state, memory

d) state, identity, behavior

Answer

- d) identity, state, behavior



Revision Question

- What are the five meaningful metrics to measure the quality of abstraction?

a) Contracts, Cohesion, Completeness, Primitivenes, Sufficiency

b) Cohesion, Coupling, Sufficiency, Completeness, Primitivenes

c) Coupling , Cohesion, Safety, Completeness, Primitivenes

Answer

- b) Cohesion, Coupling, Sufficiency, Completeness, Primitivenes

Revision Question

- What is an advantage of polymorphism?

a) The same program logic can be used with objects of several related types.

b) Variables can be re-used in order to save memory.

c) Constructing new objects from old objects of a similar type saves time.

d) Polymorphism is a dangerous aspect of inheritance and should be avoided.

Answer

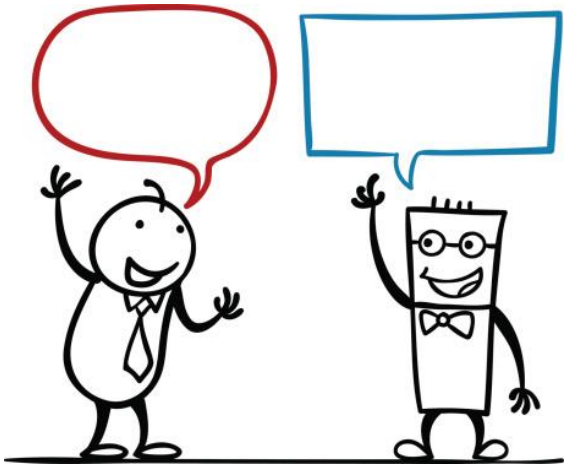
- a) The same program logic can be used with objects of several related types.

Classification

- Classification is the means whereby we **order knowledge**
  - recognize key abstractions and mechanisms with similarities (sameness)
- Seeks to **group** things that have a commonality (e.g., structure or exhibit a common behavior)

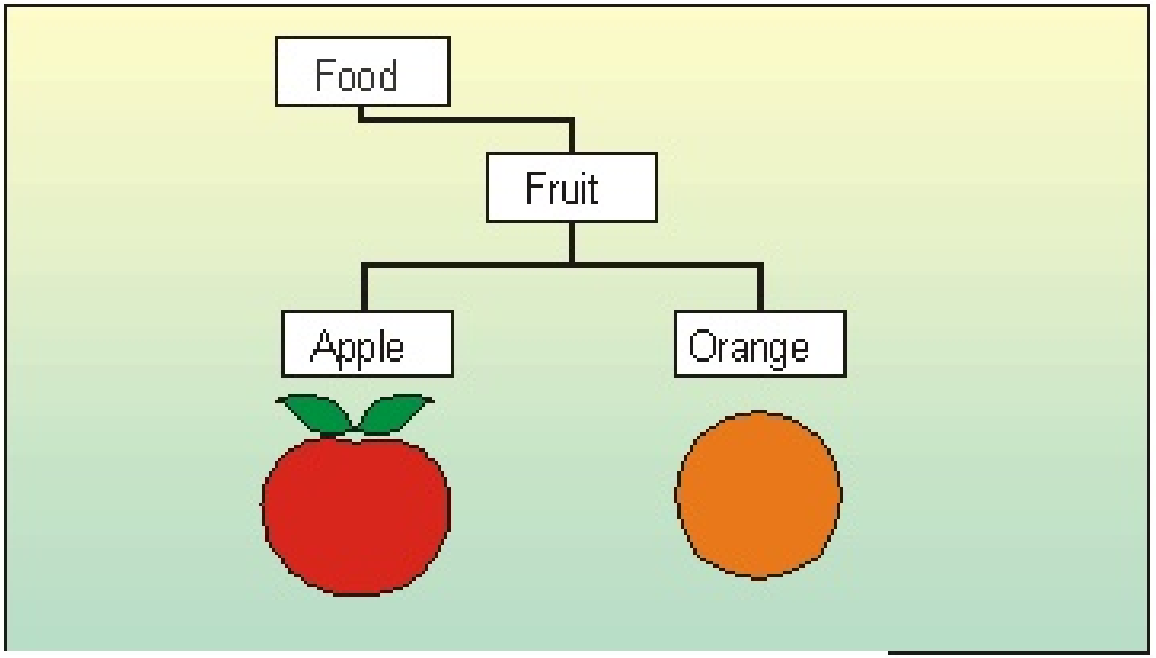
Activity

- Classify the objects in the image below into groups (5 Minutes)



Why Classify Objects?

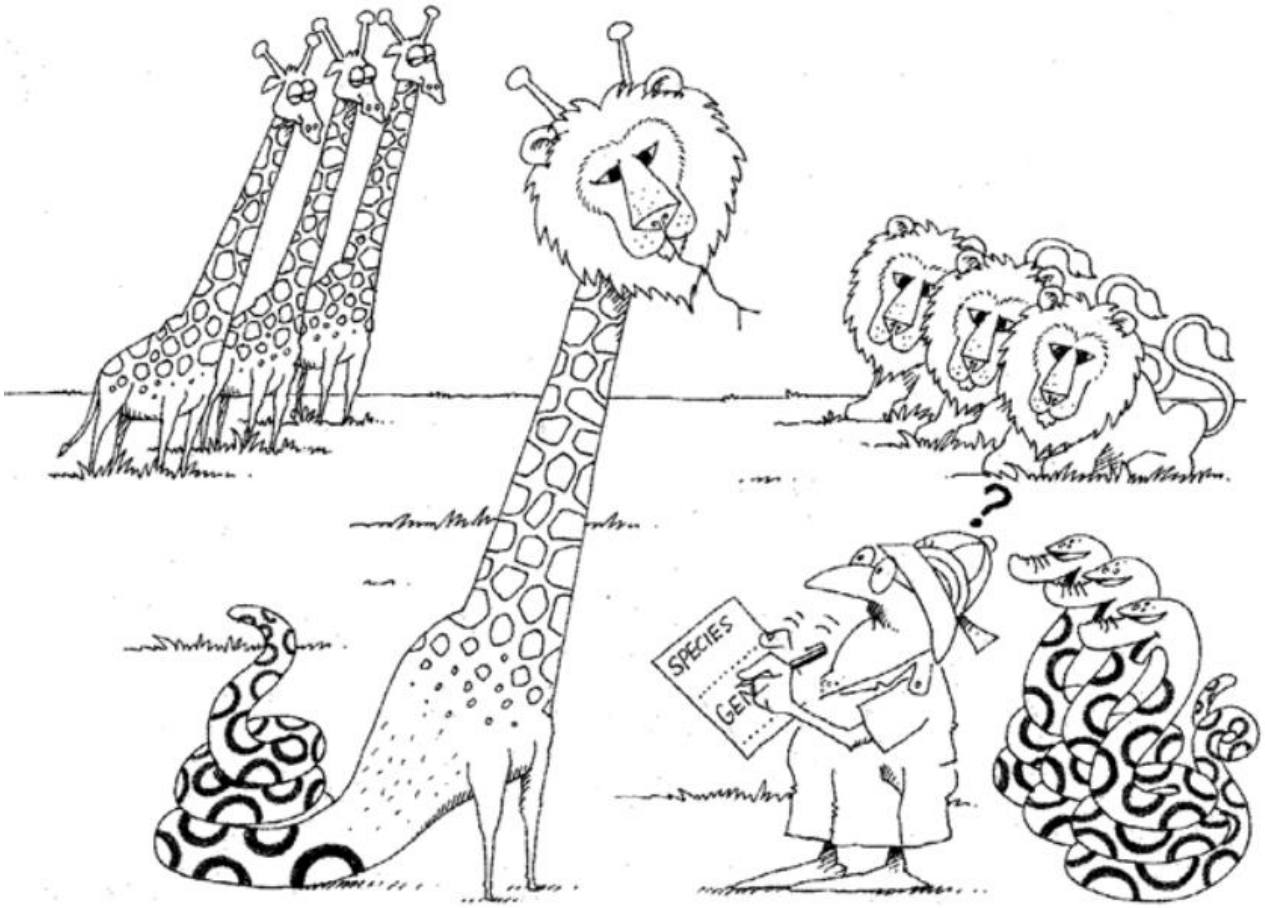
Classification and generalization are important functions for any language describing the physical world



Why Classify Objects?

- Helps us to identify generalization, specialization, and aggregation hierarchies among classes
- Recognizing common patterns of interaction among objects, we come to invent the mechanisms that serve as the soul of the implementation
- Guides decision making about modularization
  - We may choose to place certain classes and objects together in the same module or in different modules, depending on the sameness we find among these declarations
- Coupling and cohesion also indicate a type of sameness
- Plays a role in allocating processes to processors.
  - We place certain processes together in the same processor or different processors, depending on packaging, performance, or reliability concerns

Why is Classification Difficult?



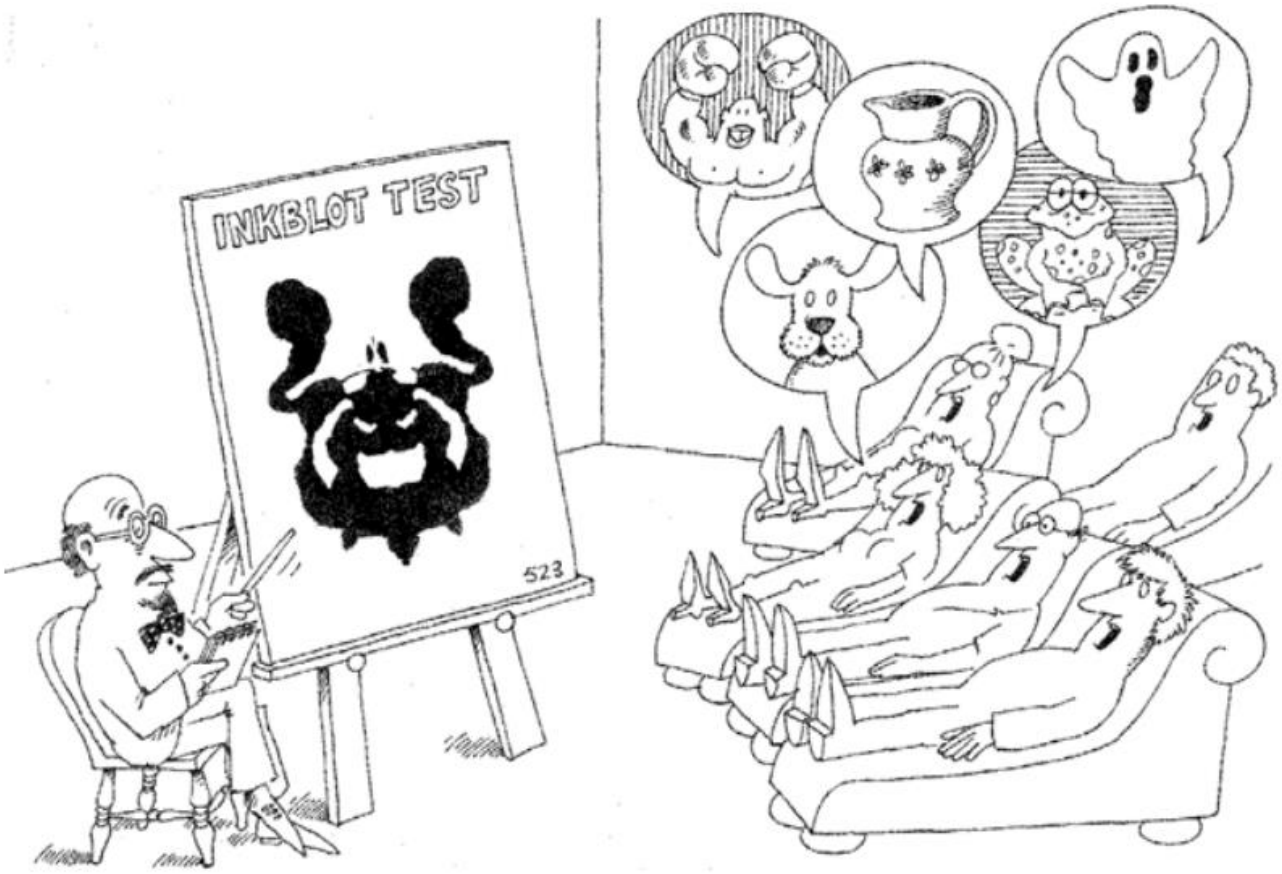
Why is Classification Difficult?

- Boundaries are not always absolute
  - e.g., fuzzy or overlap multiple classifications
  - Example - look at your leg. Where does your knee begin, and where does it end?
  - Example - recognizing human speech, how do we know that certain sounds connect to form a word and are not instead a part of any surrounding words?

Classification Dependencies

- Classification depends on what you want classification to do
  - Highly dependent on the reason for the classification
- For example, in Biology, if you want it to reflect precisely the genetic relatedness among species, that will give you one answer. But if you want it instead to say something about levels of adaptation, then you will get another

Perspective



Important Factors

- Different observers will classify the same object in different ways
- Classification is relative to the perspective of the observer doing the classification
- Intelligent classification requires a tremendous amount of creative insight
  - sometimes the answer is evident, sometimes it is a matter of taste, and at other times, the selection of suitable components is a crucial point in the analysis

Incremental and Iterative Nature of Classification

- In practice, it is common to assert a certain class structure early in a design and then revise this structure over time
  - Experience, Additional knowledge discoveries, Identify previously unrecognized commonality

Question

- Classification is relative to the perspective of the observer doing the classification

a) True
b) False

Answer

- a) True

Identifying Classes and Objects

- Historically, there have been three general approaches to classification:
  - 1. Classical categorization
  - 2. Conceptual clustering
  - 3. Prototype theory

Object-Oriented Analysis

- Boundaries between analysis and design are fuzzy
- Focus on discovering classes and objects that form the vocabulary of the problem domain
- Identify abstractions and mechanisms in different models helps provide a design solution

Classification Approaches

- Approaches for analysis that are relevant to object-oriented systems
  - Classical Approaches
  - Behavior Analysis
  - Domain Analysis
  - Use Case Analysis
  - CRC Cards
    - Class/Responsibilities/Collaborators
  - Informal English Description
  - Structured Analysis

Key Abstractions and Mechanisms

- Identifying Key Abstractions
  - appropriate choice of objects depends, of course, on the purposes to which the application will be put and the granularity of information to be manipulated

Question

- What are the three approaches to classification:

a) classical categorization, pattern grouping, prototype theory

b) classical categorization, conceptual clustering, prototype theory

c) classical categorization, conceptual clustering, system theory

Answer

- b) classical categorization, conceptual clustering, prototype theory

Question

- Classification is straightforward and simple for all object orientated problems?

a) True
b) False

Answer

- b) False

Question

- Which of the following supports the concept of hierarchical classification?

a) Polymorphism

b) Encapsulation

c) Abstraction

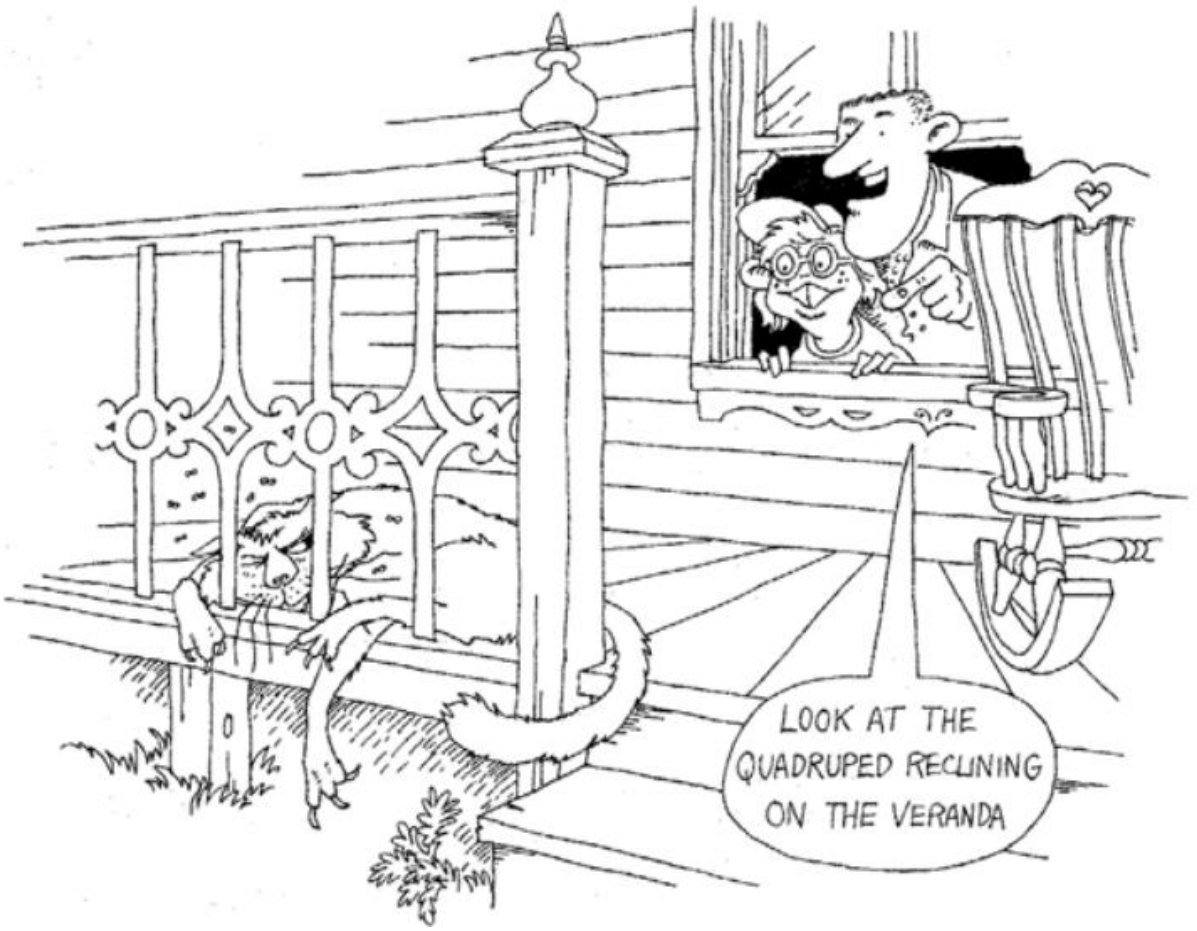
d) Inheritance

Answer

- d) Inheritance

Use of Hierarchical classification avoids defining the properties of object explicitly at each level which have acquired their properties from higher levels.

Level of Abstraction



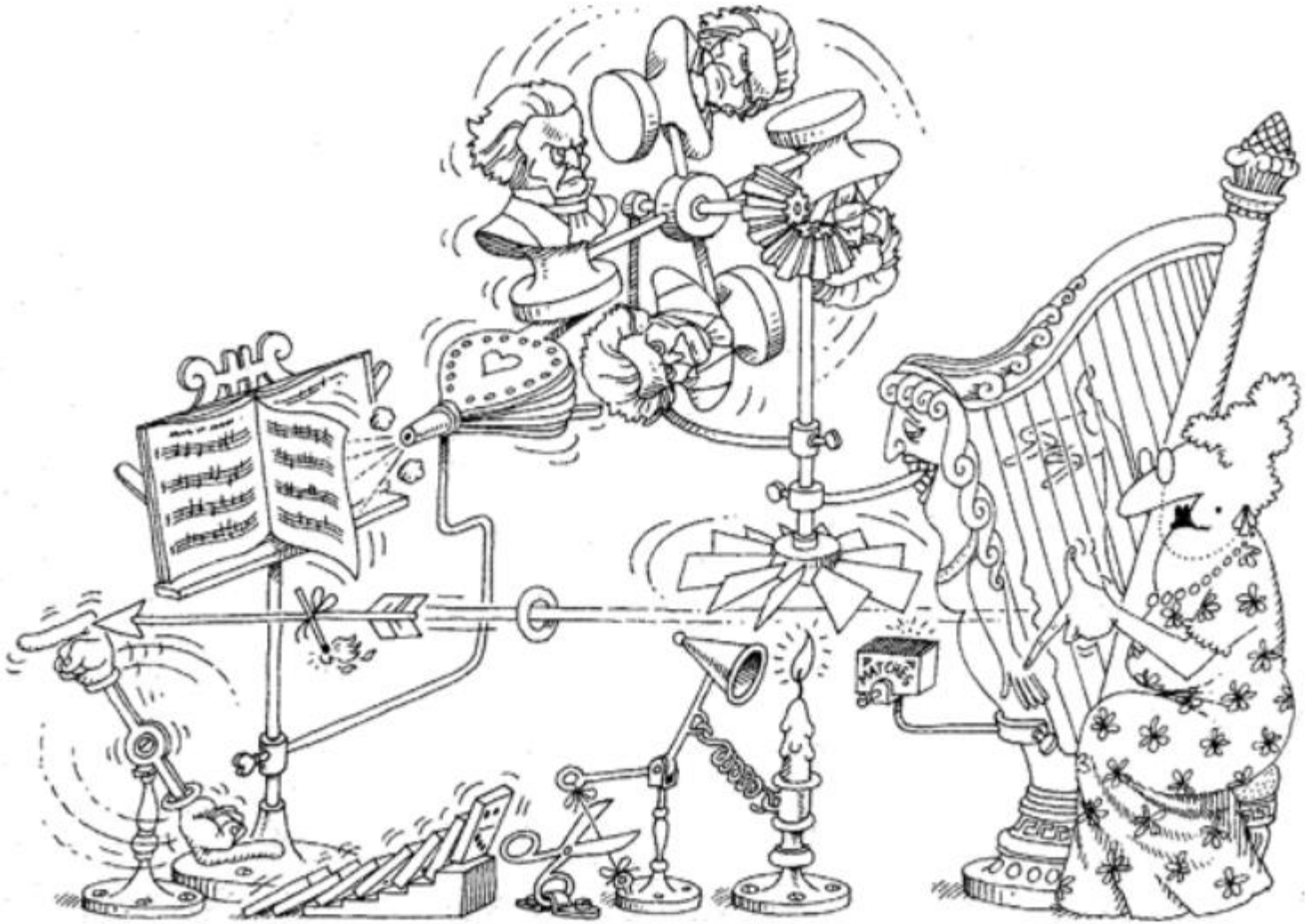
Level of Abstraction

- Classes and objects should be at the right level of abstraction: neither too high nor too low

Naming Key Abstractions

- Objects should be named with proper noun phrases, such as the Sensor or just simply shape
- Classes should be named with common noun phrases, such as Sensor or Shape
- The names chosen should reflect the names used and **recognized by the domain experts**, whenever possible
- Modifier operations should be named with active verb phrases, such as draw or moveLeft
- Selector operations should imply a query or be named with verbs of the form “to be,” such as extentOf or isOpen
- The use of underscores and styles of capitalization are largely matters of personal taste. No matter which cosmetic style you use, at least have your programs be self-consistent

Identifying Mechanisms



Identifying Mechanisms

- Mechanisms are the means whereby objects collaborate to provide some higher-level behavior
- Mechanisms are actually one in a spectrum of patterns we find in well-structured software systems

Example Mechanisms

- A mechanical linkage connects the accelerator directly to the fuel injectors
- An electronic mechanism connects a pressure sensor below the accelerator to a computer that controls the fuel injectors (a drive-by-wire mechanism)
- No linkage exists. The gas tank is placed on the roof of the car, and gravity causes fuel to flow to the engine. Its rate of flow is regulated by a clip around the fuel line; pushing on the accelerator pedal eases tension on the clip, causing the fuel to flow faster (a low-cost mechanism)

Summary

- Clear idea of Classification in Object Orientated Analysis and Design
- Classification and the challenging problem of clustering
- Classification is an incremental and iterative process
- Three approaches to classification include classical categorization (classification by properties), conceptual clustering (classification by concepts), and prototype theory (classification by association with a prototype)

This Week

- Review Slides and Previous Chapters
- Read Chapter 5
- Online Quizzes
- Researching/Planning Project
- Version Control
  - Regularly submitting project updates
  - Each member of the group (evidence)
  - Working as a `Team'

Questions/Discussion

Version Control

- GitHub
  - Register and setup your GitHub username
  - Create repository and every member of the group should have access

Coursework

- Groups
  - Team Orientated Software Engineering Problem
- Version Control
- Research/Investigating Requirements/Solutions/Options

Submissions

1. System analysis: study, understand, and define requirements for the system (model of the system's functional requirements)
2. Defining the boundaries of the problem
3. Use-case model
4. Deployment view
5. Sequence diagram and operation
6. Design to code (UML design diagram)

Important

- Evidence based
  - ▷Citations/facts/statistics
  - ▷Repository history
    - `Group' Project

the problem:

if you think your only tool is a hammer,
every problem starts looking like a nail.

Critical Thinking/Logic

- Ask how you could design a system `incorrectly'
- What is the `wrong' way? (why)
  - Don't just look for a solution
  - Identify alternatives (explore/research)
- What are all the possibilities?
  - Compare/explain

Requirements

- Determine what you must build for your customer by defining the boundary of the problem
  - Determining what you must build (limits)
- First step is to gather documentation of the problem
  - Aim and associated high-level requirements and constraints
  - Vision statement, functional requirements, non-functional requirements, constraints, ...

Final Note

- Remember concepts, such as:
  - flexibility, encapsulation, usability, cost, problem requirements, ..
- Review slides/reference material and ask how they can/cannot help solve the problem
- `Object Orientated' Analysis and Design