Textbook: Object-Oriented Analysis & Design

Chapter 2: Part-B
The Object Model

H.S. Sarjoughian

CSE 460: Software Analysis and Design

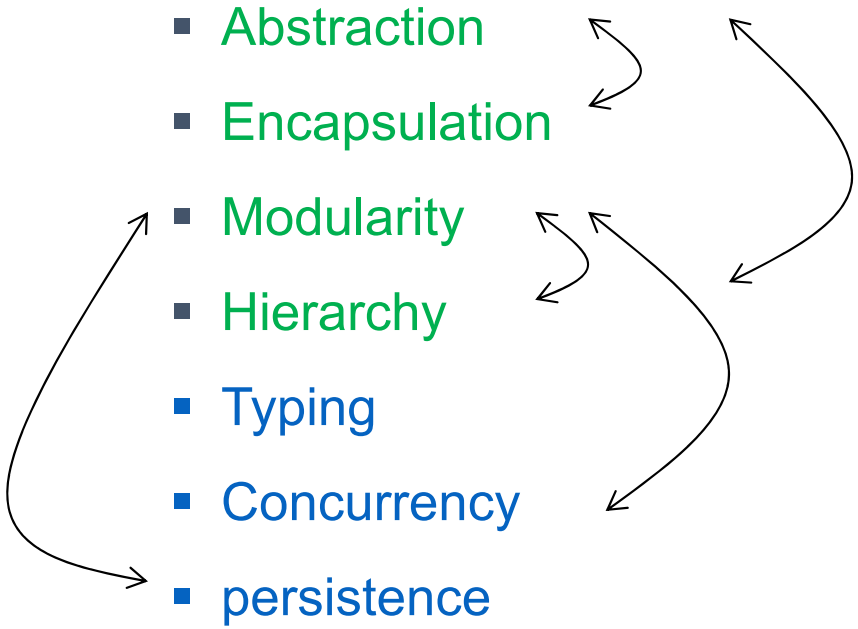
School of Computing, Informatics and Decision Systems Engineering
Fulton Schools of Engineering

Arizona State University, Tempe, AZ, USA

Object Model and Key Relationships

The Object Model is the collection of principles that form the foundation of object-oriented analysis and design and tools.

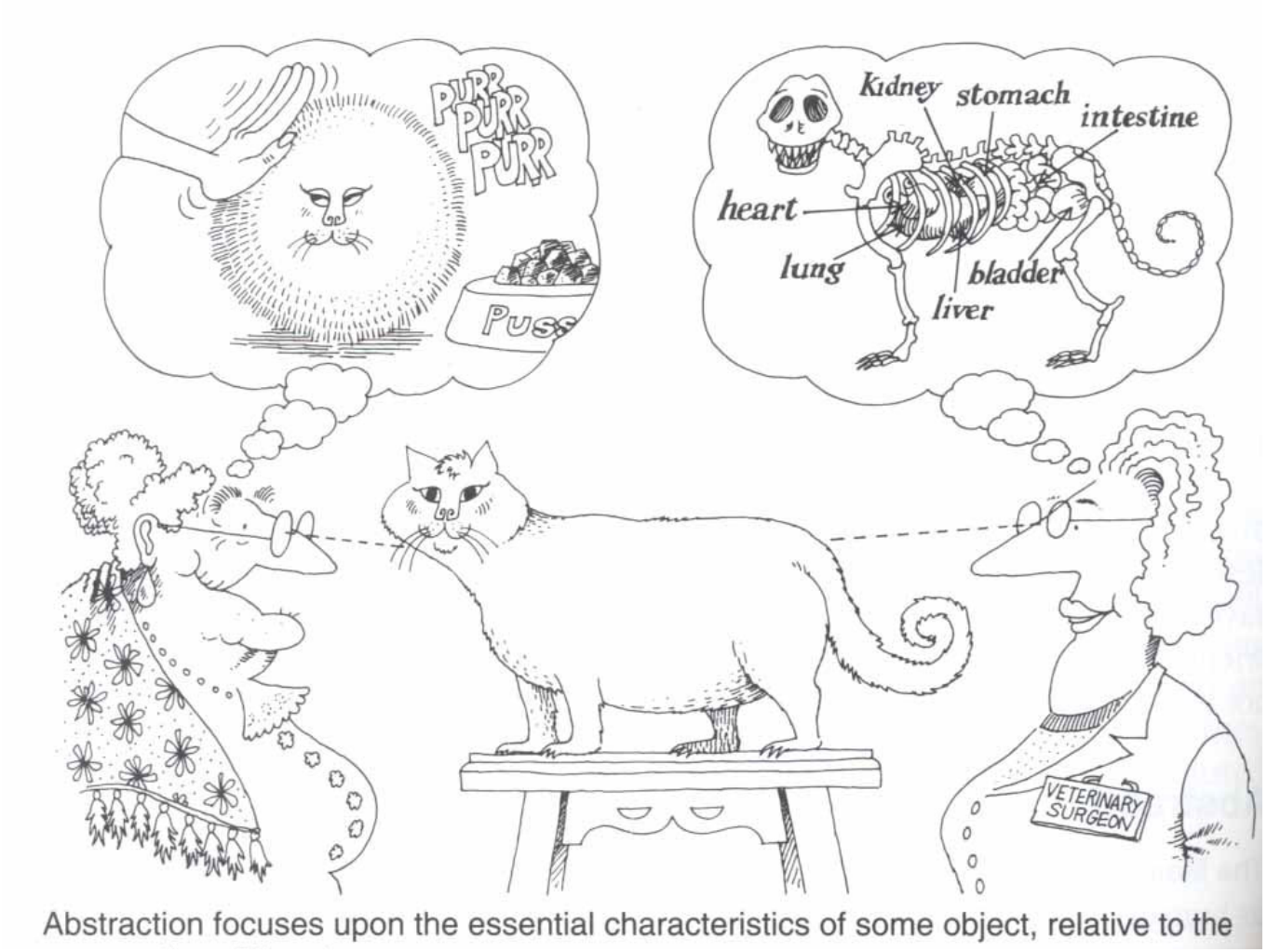
Object model provides a paradigm for software engineering emphasizing the principles of:



key relationships underpinning the "goodness" of software analysis, design, …

Many software engineering tools are founded on the principles of the Object Model.

Abstraction Caricature



G. Booch, OOAD

abstraction focuses upon the *essential characteristics* of **some object**, relative to the *perspective of the viewer*

Abstraction Definition

- An abstraction denotes the ***essential characteristics*** of an object that distinguish it from all other kinds of ***objects***.
- An abstraction provides crisply defined ***conceptual boundaries***, relative to the perspective of the user.
  - Abstraction arises from a recognition of similarities between certain *objects*, *situations*, or *processes* in the real-world and the decision to concentrate upon these while ignoring their differences.
  - An abstraction is a simplified description, or specification, of a system that emphasizes some of its details or properties while suppressing others.

Abstraction Definition (cont.)

Any abstraction has static as well as dynamic properties.

Ex: Data file object (e.g., bank account statement):

- A data object has name and content – these are its static properties
- The values of any static properties can change (dynamic); the values are subject to change depending on the state of the object – e.g., bank account balance may increase.

Abstraction Definition (cont.)

Each abstraction satisfies some ***invariance*** – conditions which it should preserve. For example, operations of an abstraction may be defined in terms of ***pre-conditions*** and ***post-conditions***. These pre- and post-conditions provide the **contractual basis** for the abstraction.

Ex: Consider Client/Server *Objects* of a website:

Invariance of the server object: client object will use a known data transmission rate (e.g., 10^3 gbps) to interact with the server object. A server object incapable of supporting the specified bandwidth fails to satisfy its contract!

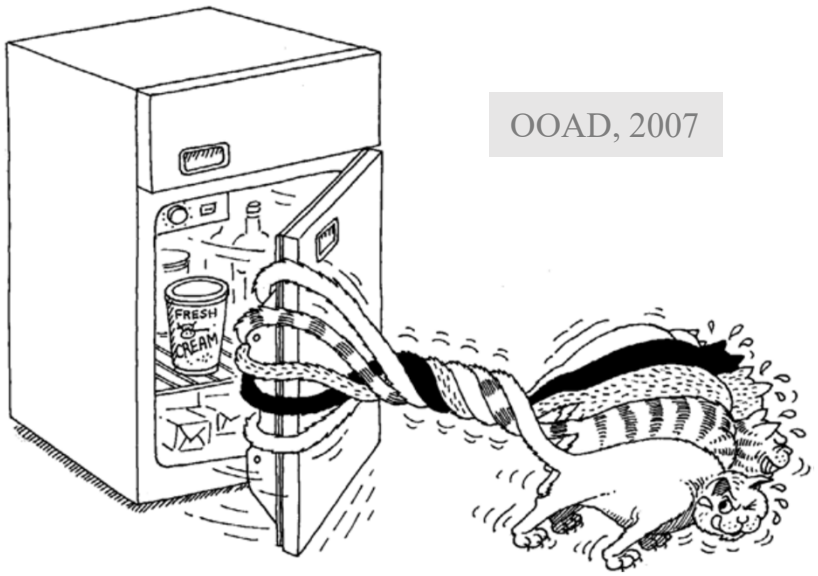
Hydroponics Farm

- A hydroponics farm refers to a type of farming where plants are grown in nutrient solutions – sand or other kinds of soil are not used.
- An automated system (gardener) is necessary to monitor and control temperature, humidity, light, nutrient concentrations, etc.
- An automatic gardener obtains information and controls growth of the plants under varying conditions and requirements

Kinds of Abstractions

- Entity: an object that represents a useful model of a problem-domain or solution-domain entity.
- Action: An object that provides a generalized set of operations, all of which perform the same kind of function.
- Virtual Machine: An object that groups together operations that are all used by some superior level of control or operations that all use some lower-level set of operations.
- Coincidental: An object that packages a set of operations that have no relation to one another.

data and function abstractions are fundamental to software engineering



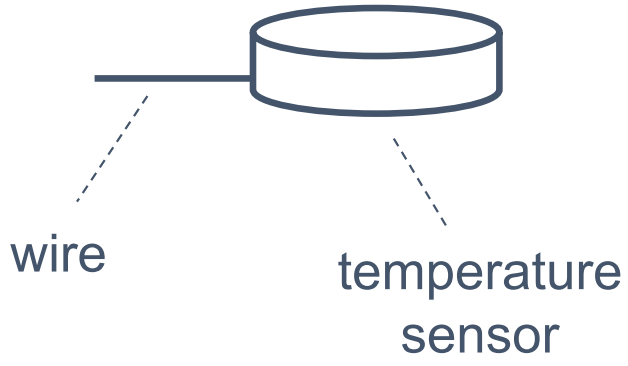
OOAD, 2007

Abstraction Example

Hydroponics Farm

- Sensors: temperature, light, humidity, …

- Temperature sensor
  - Operations:
    - measure and report temperature
    - calibrate



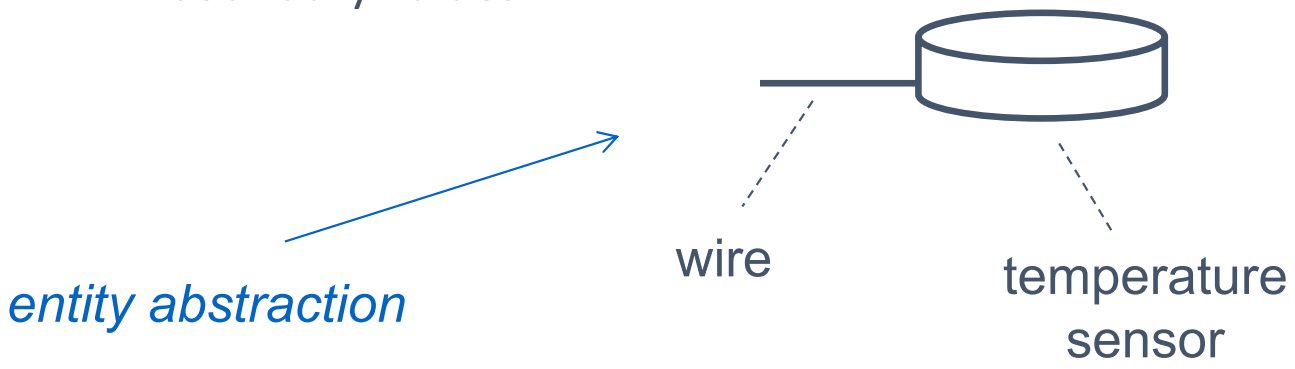
Abstraction Example (cont.)

Abstraction 1 (*passive*):

Temperature sensor

action abstraction

- Operations:
  - measure and report temperature *upon request*
- Attributes:
  - Name – string
  - Location – location can be a name, a number, …
  - Range – a specified set of values with minimum and maximum boundary values



entity abstraction

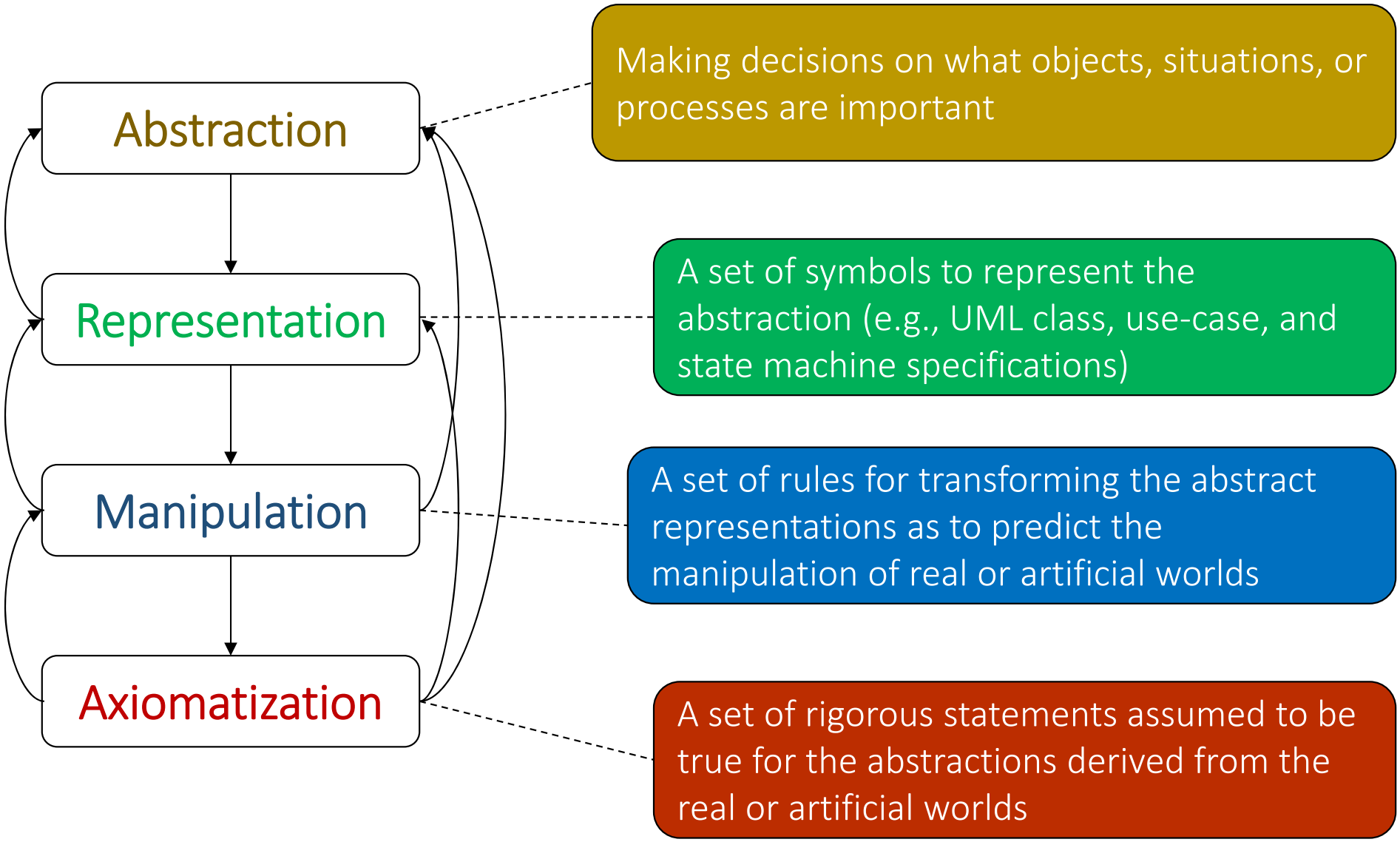
Abstraction Example (cont.)

Abstraction 2 (*active*):

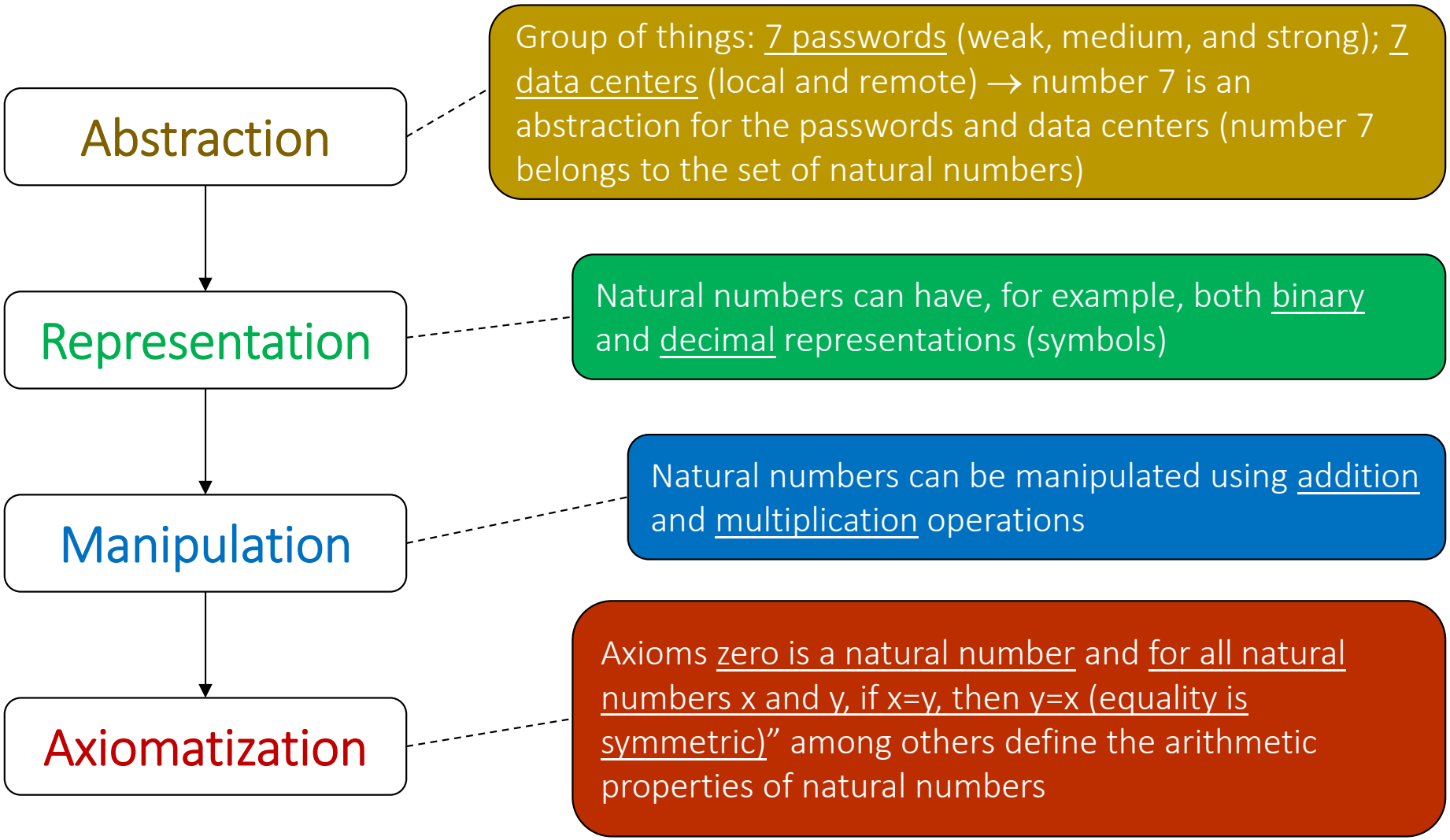
Temperature sensor

- Operations:
  - measure and notify an object *when temperature exceeds a certain set point value*
- Attributes:
  - Name – string
  - Location – location can be a name or a number
  - Range – a specified set of values with minimum and maximum boundary values

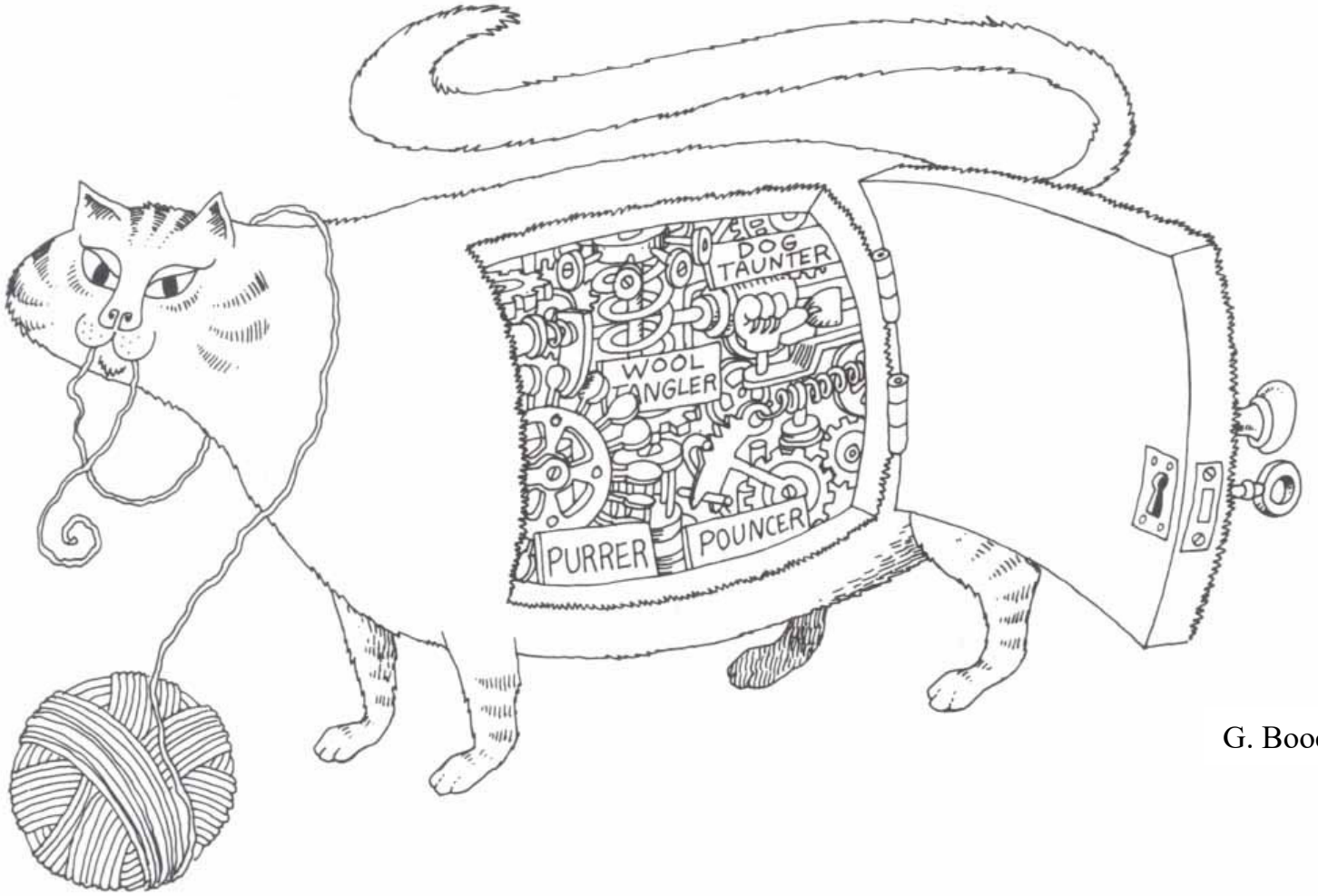
A Process for Abstraction: Concept



A Process for Abstraction: An example



Encapsulation Caricature



G. Booch, OOAD

encapsulation **hides the details** of the *implementation of the object*

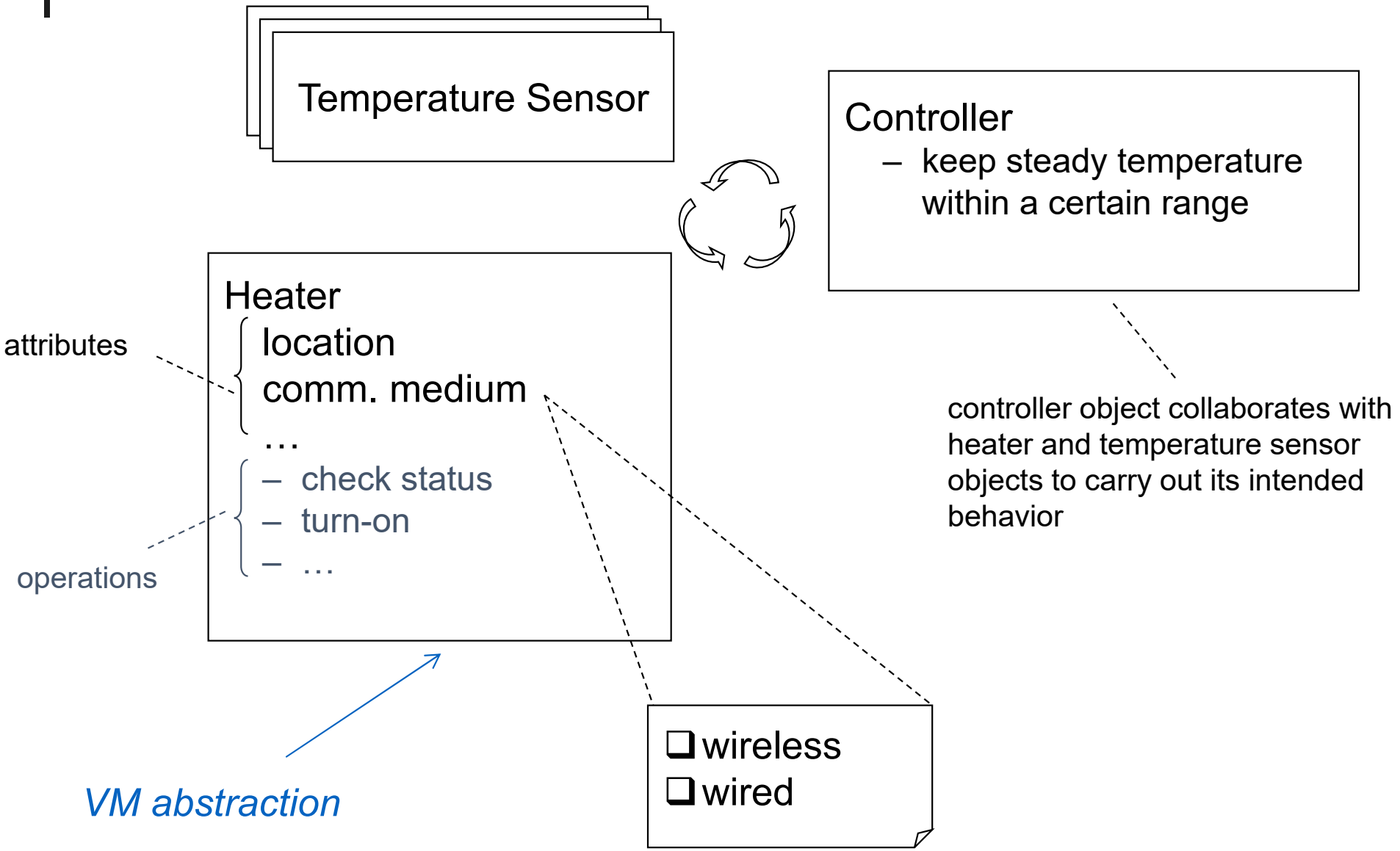
Encapsulation Definition

- Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior
- Encapsulation serves to separate the *contractual interface* of an abstraction and *its implementation*

Encapsulation and abstraction are complementary concepts

- Abstraction focuses on observable behavior of an object
- Encapsulation focuses upon the implementation that gives rise to some desired observable behavior

Encapsulation Examples (1)



Encapsulation Examples (2)

- An *abstraction* of an object should precede the decisions about its *implementation*. [OOAD (page 50)]
  - Operations (refers to functions or methods)
  - Objects
- Once an implementation is selected, it should be treated as a secret of the abstraction and hidden from most clients. [OOAD (page 50)]

Encapsulation Examples (3)

- Example: Weather Channel App for the city of Tempe, Arizona
  - **operation**
    - A sensor measures temperature every 1 minute (or less frequently)
    - A sensor measurement can be the average of multiple sensor readings
  - **object**
    - A heater can be turned on/off and be queried about its status (running or not)
    - A heater can be maintain the temperature of an enclosure to remain within some lower and upper limits

Modularity Caricature

G. Booch, OOAD

modularity packages abstractions into discrete units

Modularity Definition

Modularity is the property of a ***system (software)*** that has been decomposed into ***a set of cohesive and loosely*** coupled modules

Modularity, encapsulation, and abstraction are complementary concepts

- An object provides a crisp boundary around a single abstraction
- Modularity and encapsulation provide barriers around this abstraction

choices of modules are based on domain knowledge, patterns, …

What is a Module?

- A unit of code that serves as a building block for the physical structure of a system – it consists of interface and implementation

- A program that contains declarations, expressed in the vocabulary of a particular programming language, that form the physical realization of some or all of the classes and objects in the logical design of the system

*modules are necessary to support **structural changes***

Modularity Example

Hydroponics example

- Module for plans of how to grow various plants
- Module for graphical user of the system
- Module for monitoring sensor readouts
- Module for control of greenhouse environment
- …

choices of modules are based on requirements, domain knowledge, …

Hierarchy Caricature

abstractions form a hierarchy

Abstractions form a hierarchy.

G. Booch, OOAD

Hierarchy Definition

Hierarchy is a ranking or ordering of abstractions

- *Part-Of* hierarchy – aggregation
- *Is-A* hierarchy (inheritance) – generalization & specialization

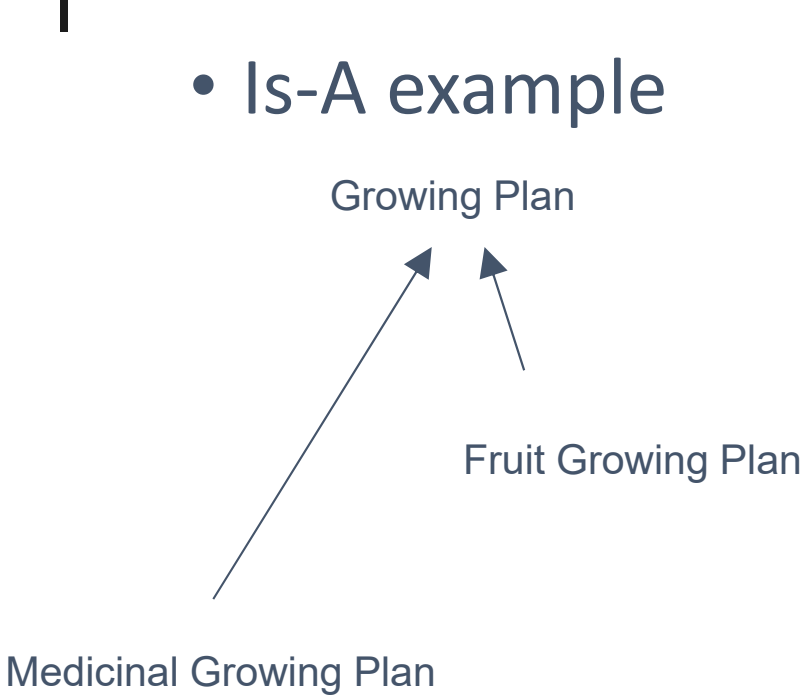
- Single inheritance: one class shares the structure and behavior of one class

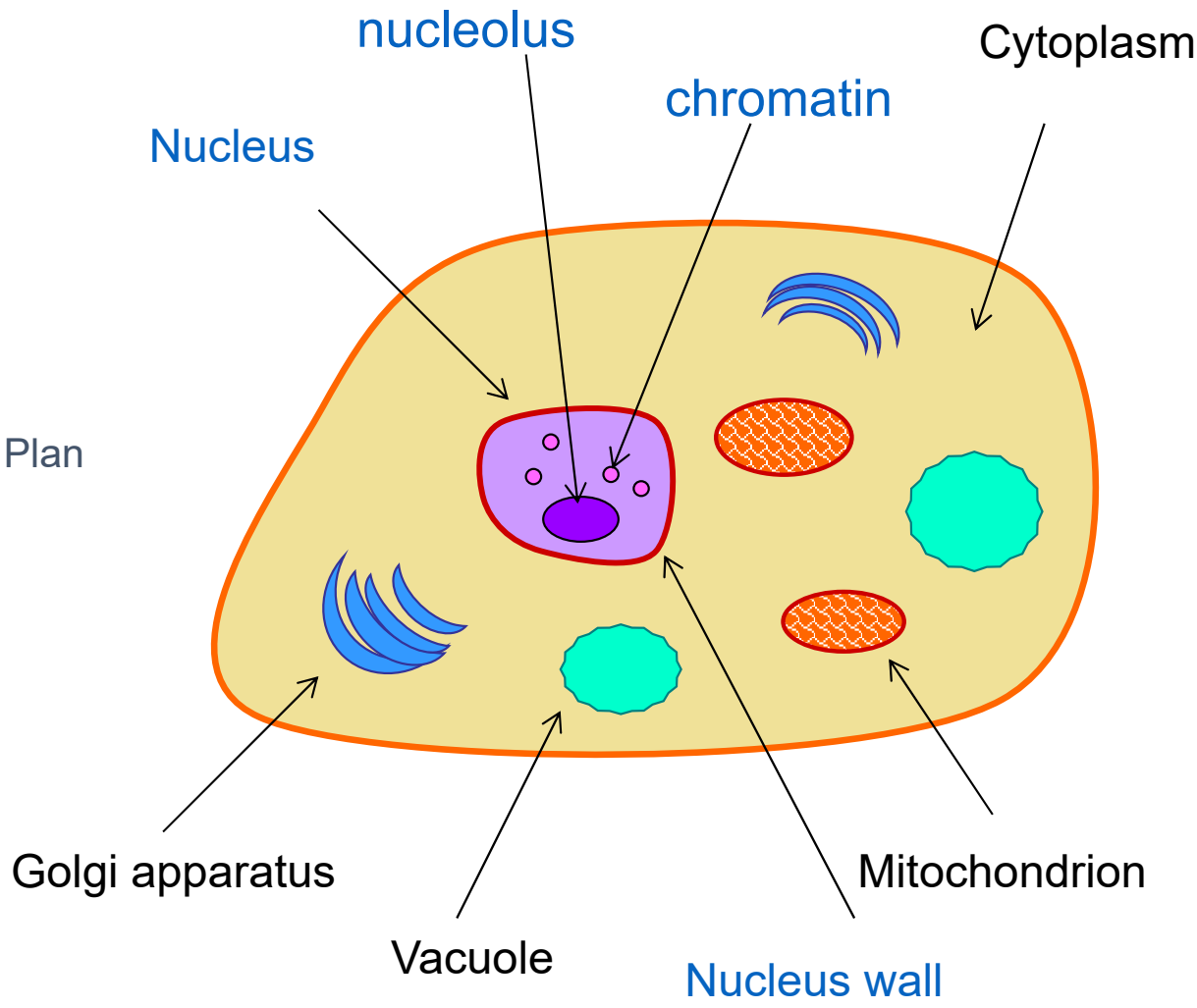
- Multiple inheritance: one class shares the structure and behavior of multiple classes

the principles of abstraction, encapsulation, modularity, and hierarchy work together.

Hierarchy Example

- Is-A example





plant cell structure

Exercise

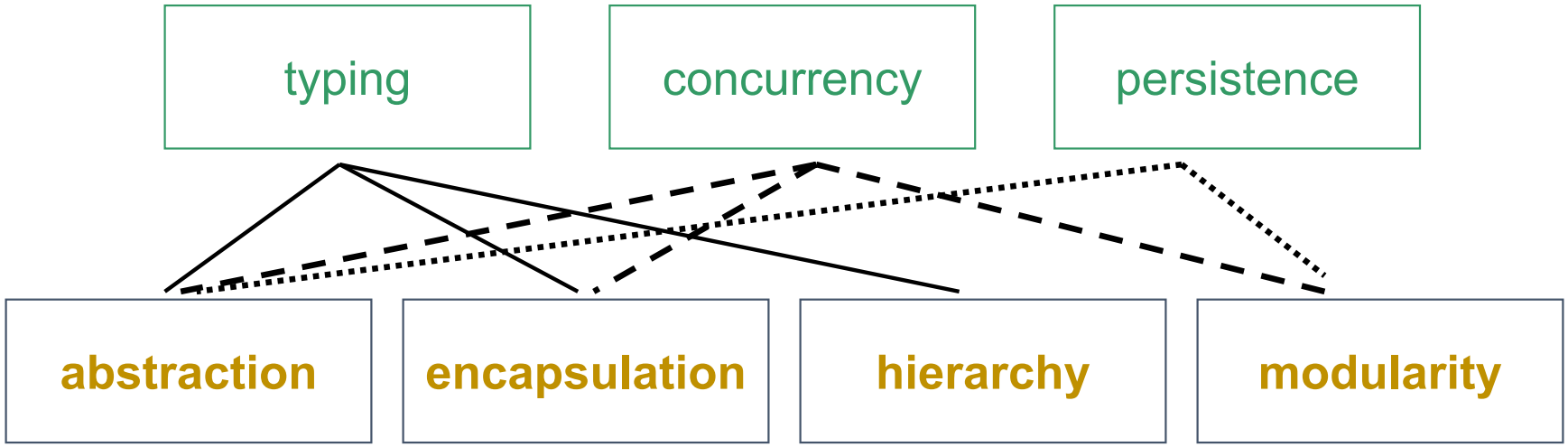
Characterize the different hierarchical relationships implied for:

- A soccer team consists of 18 players 11 of which can play at any time during a game. One scheme is to have three defensive players, four midfielder players, three offensive players, and one goalkeeper. Only one goalkeeper is allowed. A team must have a captain who is a member of a team and is a player on the field (i.e., not a player sitting on the reserve bench!).

The Object Model

The Object Model is based on seven principles which collectively provide the foundation for object-oriented software development

higher-level elements



lower-level elements

——— directly related to the Object Model

– – – – somewhat directly related to the Object Model

..... indirectly related to the Object Model