IS 525 – Chapter 1: Part 2

The Concept of Object Orientation

Dr. Nesrine Zemirli

Assistant Professor.
IS Department – CCIS / King Saud University
E-mail: nzemirli@ksu.edu.sa
Web: http://fac.ksu.edu.sa/nzemirli/home
Office: B 9 - Room 810

Chapter Topics

- Real-world objects.
- Object *identity*.
- Object's attributes and operations.
- Classes and classification.
- Encapsulation and information hiding
- Object interface.
- Aggregate and composite objects.
- inheritance and polymorphism.
- Object-oriented technology.
- Object-oriented modeling and the Unified Modeling Language (UML).

Introducing Objects

- *Definition*
  - ***Object Orientation** is about viewing and modelling the world/system as a set of interacting and interrelated objects.*

- To understand object-oriented technology, methodology and modeling, we must first understand what objects are.

OO Principles

- Abstraction
- Encapsulation
- Modularity
- Hierarchy

Principle 1: Abstraction

- A process allowing to focus on most important aspects while ignoring less important details.
- **Abstraction** allows us to manage complexity by concentrating on essential aspects making an entity different from others.

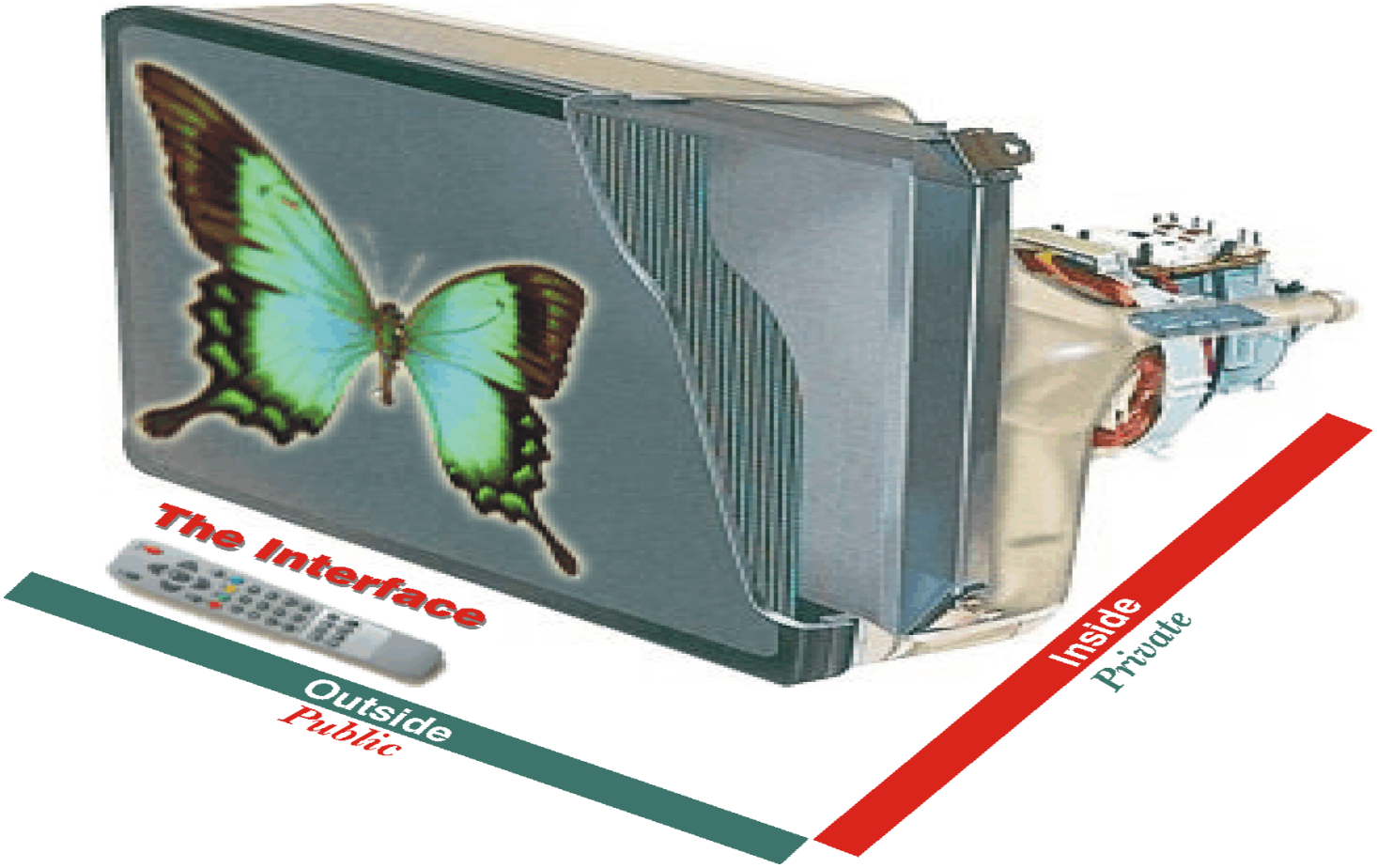
- **Abstraction** is identifying those characteristics of an entity that distinguish it from other **kinds** of entities.

An example of an order processing abstraction

Principle 2: Encapsulation

- **Encapsulation** separates implementation from users/clients.
- **Encapsulation** is the packaging of data and processes within one single unit

Encapsulation
And Information Hiding
Object as "Black Box"

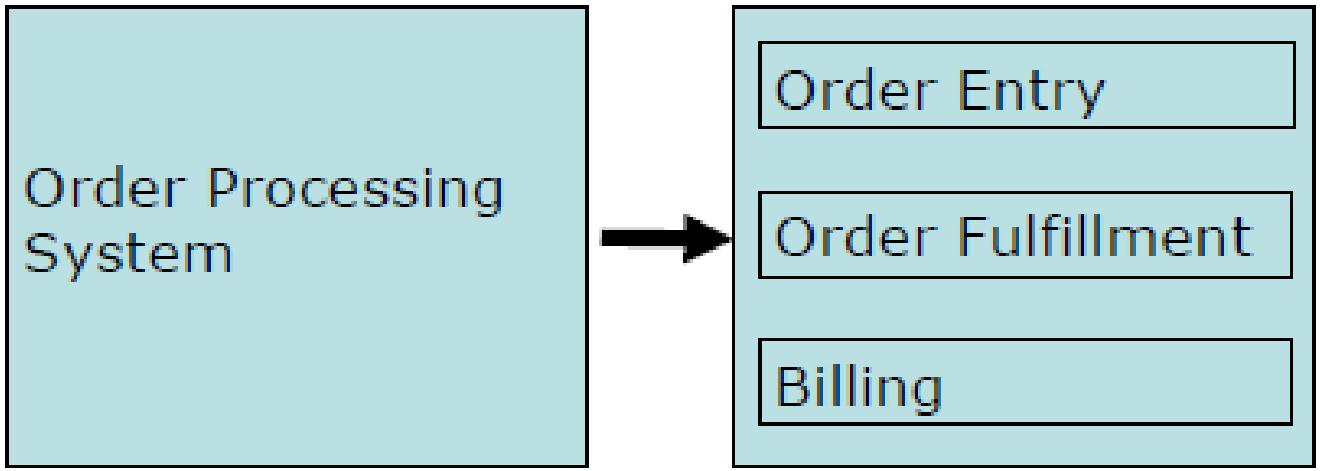


Information Hiding

- Information hiding conceals and protects what goes on inside an object from the outside world.
- When you use an ATM, encapsulation and information hiding ensure that:
  - ❶ you are not burdened with the complexity of how the machine works,
  - ❷ cannot perform operations that you are not allowed to, and
  - ❸ cannot change the way the machine operates.

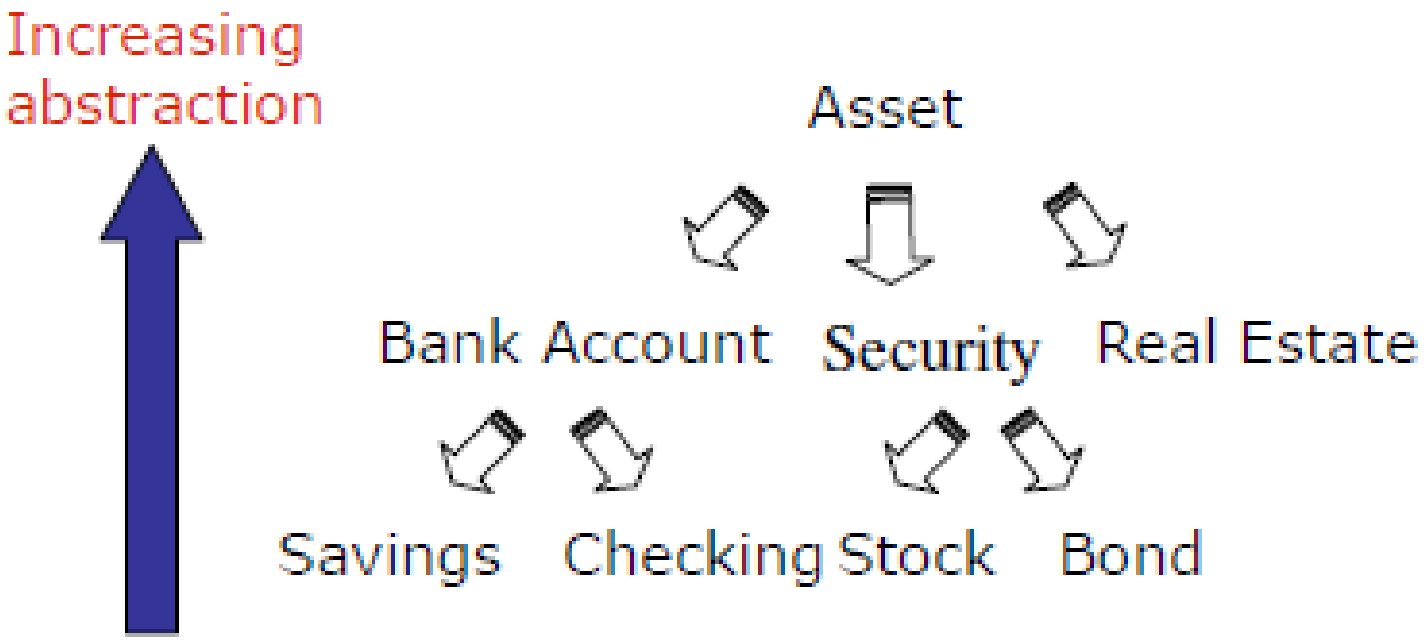
Principle 3: Modularity

- **Modularity** breaks up complex systems into small, self-contained pieces that can be managed independently.



Principle 4: Hierarchy

- Ordering of abstractions into a tree-like structure.



OO Concepts

1. Object
2. Class
3. Attribute
4. Operation
5. Interface
6. Implementation
7. Association
8. Aggregation
9. Composition
10. Generalization
11. Super-Class
12. Sub-Class
13. Abstract Class
14. Concrete Class
15. Polymorphism

Concept 1: Objects

What is an object?

- something that is perceived as an entity and referred to by name;
- something perceptible by one or more of the senses;
- something intelligible or perceptible by the mind.

- any abstraction that models a single thing
- a representation of a specific entity in the real world
- may be tangible (physical entity) or intangible

Objects in Our Business World

Files

Competitors

Employees

Assets

Customers

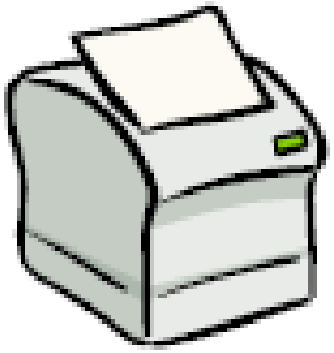
Systems

An Object has

Two aspects:

- **Information:**
  - has a unique identity
  - has a description of its structure
  - has a state representing its current condition
- **Behaviour:**
  - what can an object do?
  - what can be done to it?

Aspects Object example



1. **information:**
    1. serial number
    2. model
    3. speed
    4. memory
    5. status
2. **behaviour:**
    1. print file
    2. stop printing
    3. empty the queue

Object Identity

- The identity of an object is what distinguishes it from all other objects.
  - **Unique:** The object's identity remains solid and inviolable, regardless of errors or deliberate attempts by one entity to fake the identity of another entity.
  - **Unchanging:** an object may change superficially or profoundly, but our perception of its unique identity does not change.

Object Attributes

- Attributes are features, properties, qualities or characteristics that are associated with an object.

- Attributes are usually **paired** with **values** that **qualify** or **quantify** the attribute.

Object: John Doe

Attributes

- **Name**: John Doe
- **Born**: 1972
- **Sex**: Male
- **Marital Status**: Single
- **Citizenship**: USA
- **Degree**: MBA
- **Current Position**: VP of Purchasing
- **Years with Company**: 5
- **Starting Salary**: $70,000
- **Current Salary**: …

Attributes & Operations As Perceived By His Employer

Operations

- Order
- Approve Order
- Get Bids
- Evaluate Bids
- Report to President
- Report to Board
- Notify Inventory
- Hire
- Fire
- Evaluate Employees
- Approve Vacations
- …

Object Operations

- An operation is what an object does or is capable of doing.
  - If an object is the *subject* of a sentence with an *active* voice, then the **verb** expresses an operation:
    - Dog barks
    - Ball bounces
    - Sun shines

Concept 2: Class

What is a class?

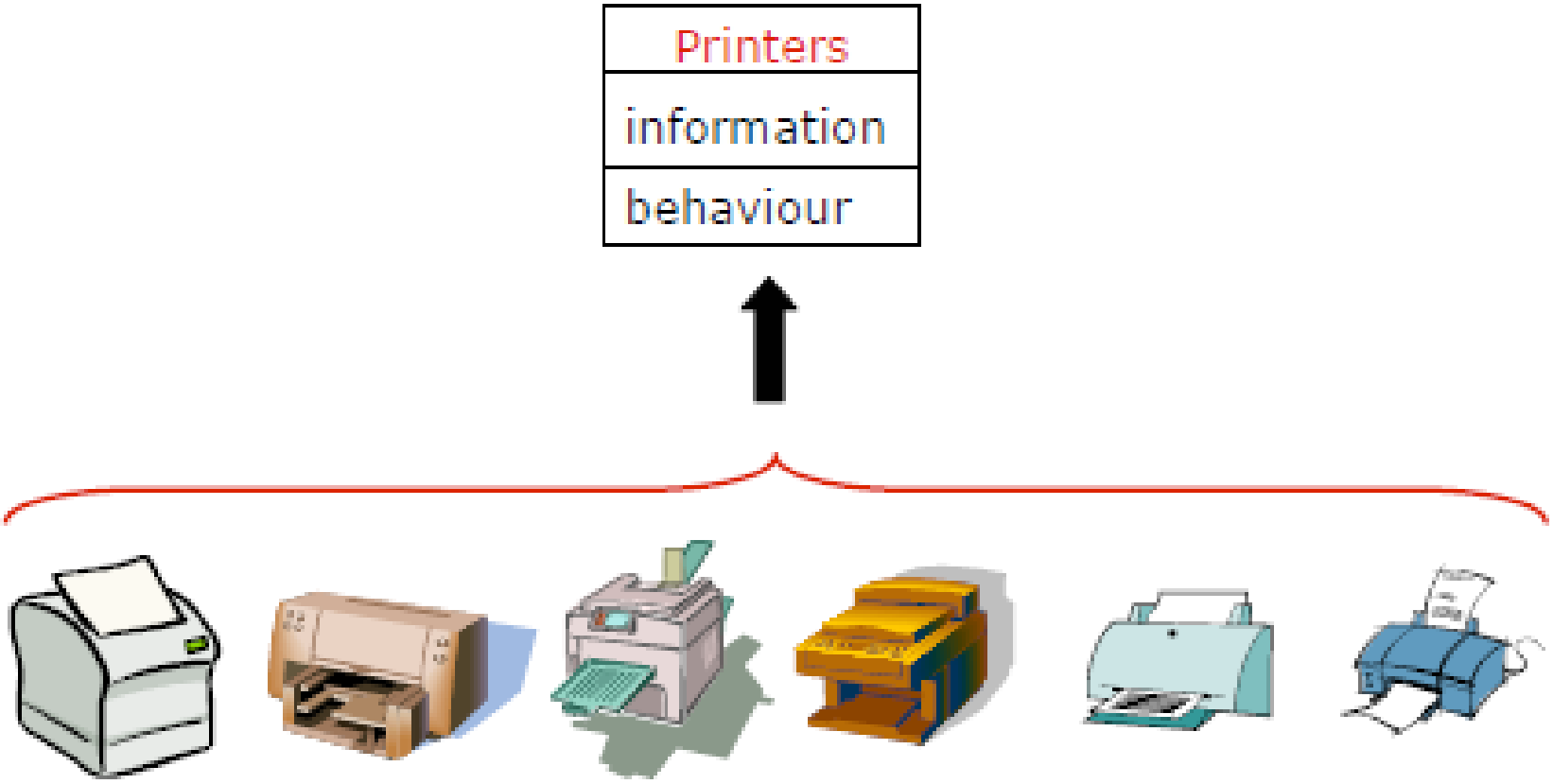
- any uniquely identified abstraction of a set of logically related instances that share similar characteristics
- a definition or template that describes how to build an accurate representation of a specific type of objects

Class is a set of objects that share the same attributes and operations.

  - Examples: agency, citizen, car, etc.

- Objects are created using class definitions as templates.

Class Example



Instance

- An instance is the concrete manifestation of a class.
  - For example, **John Doe** is an "instance" of the class **Human**.

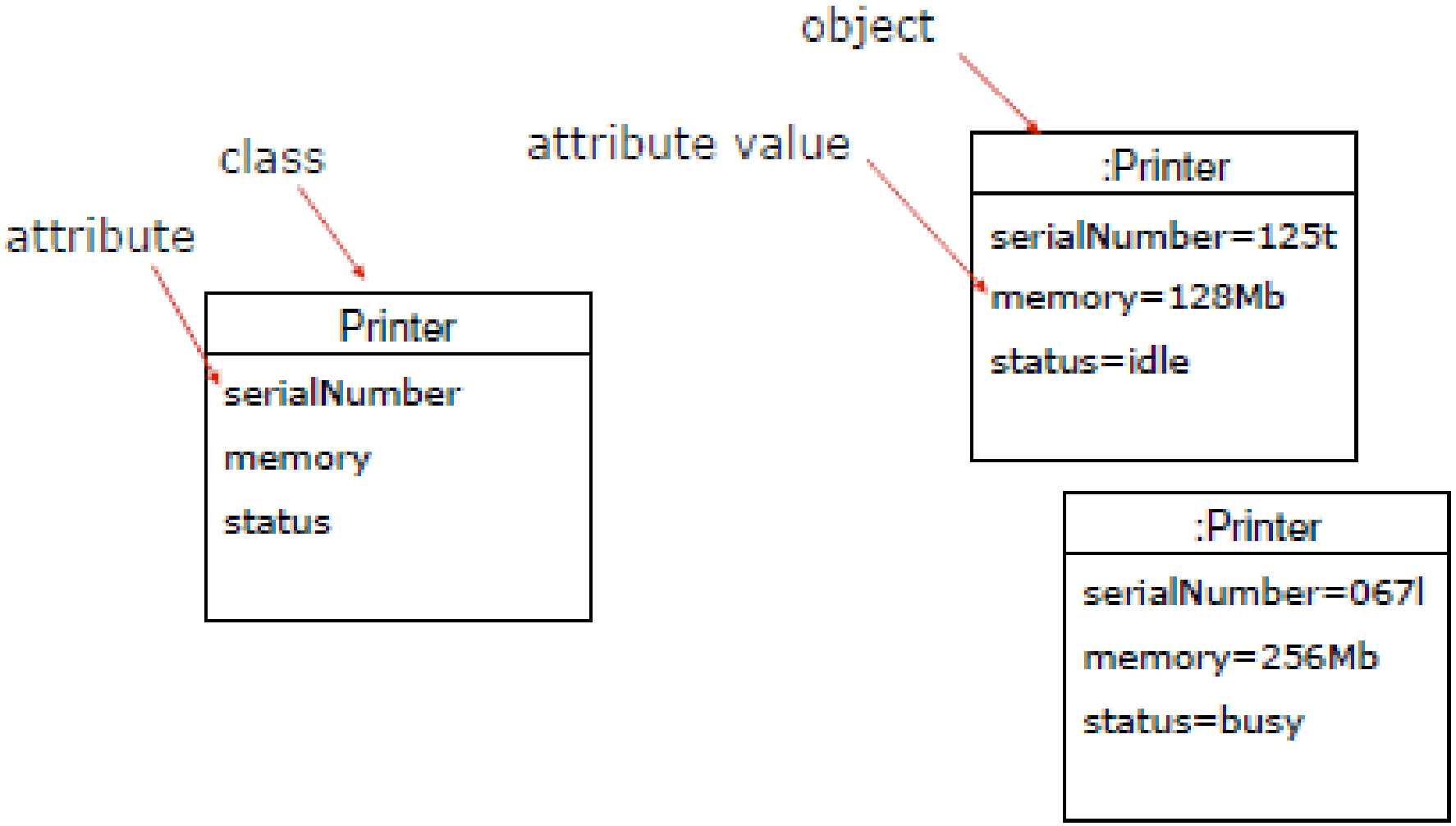
Two Types of Class

- Business Classes
  - "Business" classes are those that have a counterpart in the real world. The discovery of this type of classes and their relationships is the main task of **analysis**.
- Utility Classes
  - Utility classes are those that lack a *direct* counterpart in the real world and are used to create objects that manage the responsibilities of the information system.
- The discovery and the definition of utility classes and their relationships is the task of **design**.

Concept 3: Attribute

- **Attribute** is a named property of a class describing a range of values that instances of the class may hold for that property.
- An **attribute** has a type and defines the type of its instances.
- Only the object is able to change the values of its own attributes.
- The set of attribute values defines the state of the object.

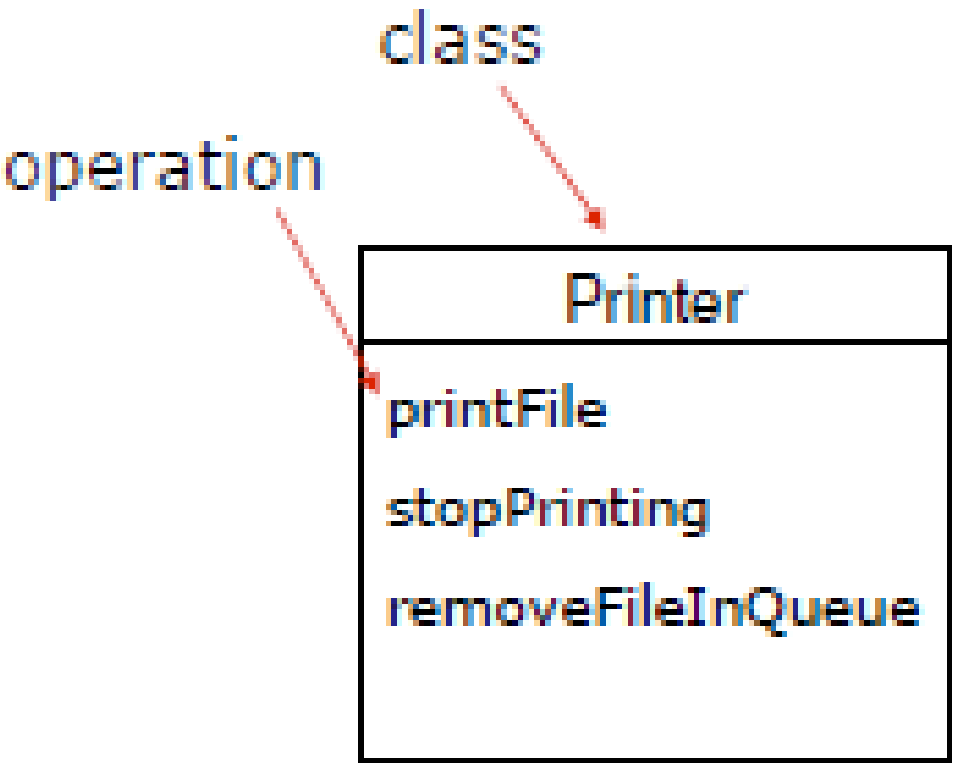
Attribute Examples



Concept 4: Operation

- **Operation** is the implementation of a service that can be requested from any object of a given class.
- An **operation** could be:
  - a question - does not change the values of the attributes
  - a command – may change the values of the attributes

Operation Example



Applying Abstraction

- Abstraction in Object Orientation:
  - use of objects and classes to represent reality
  - software manages abstractions based on the changes
- occurring to the real-world objects

Concept 5: Interface

- An object's interface consists of operations that are available to the public.
- minimum informationrequired to use an object
- allows users to access the object's knowledge
- must be exposed
- provides no direct access to object internals

Concept 6: Implementation

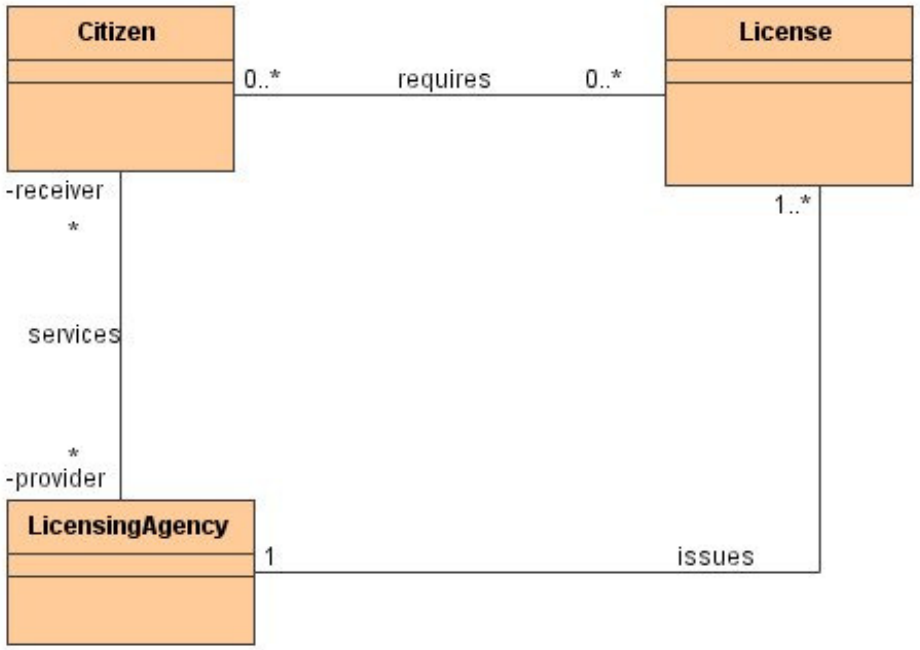
- information required to make an object work properly
- a combination of the behaviour and the resources required to satisfy the goal of the behaviour
- ensures the integrity of the information upon which the behaviour depends

Relations and Links

- Relationships:
  - between classes (relations)
  - between objects (links)
- Three kinds of relations between classes:
  1. **association**
  2. **aggregation**
  3. **composition**

Concept 7: Association

- the simplest form of relation between classes
- peer-to-peer relations
- one object is aware of the existence of another object
- implemented in objects as references



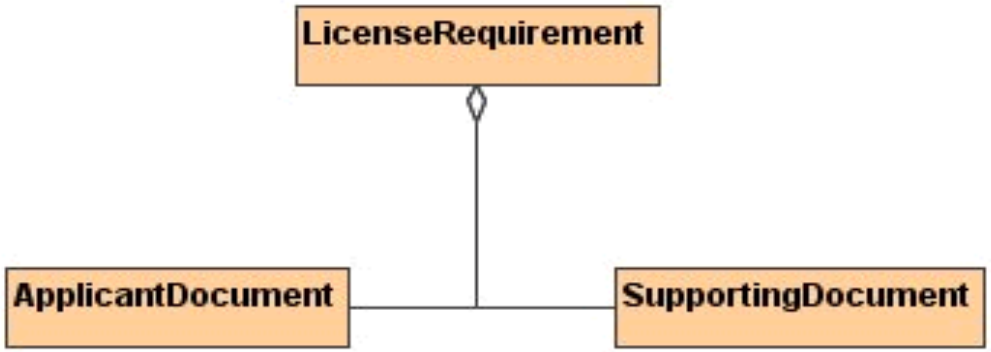
Association Examples

Associations between classes **A** and **B**:

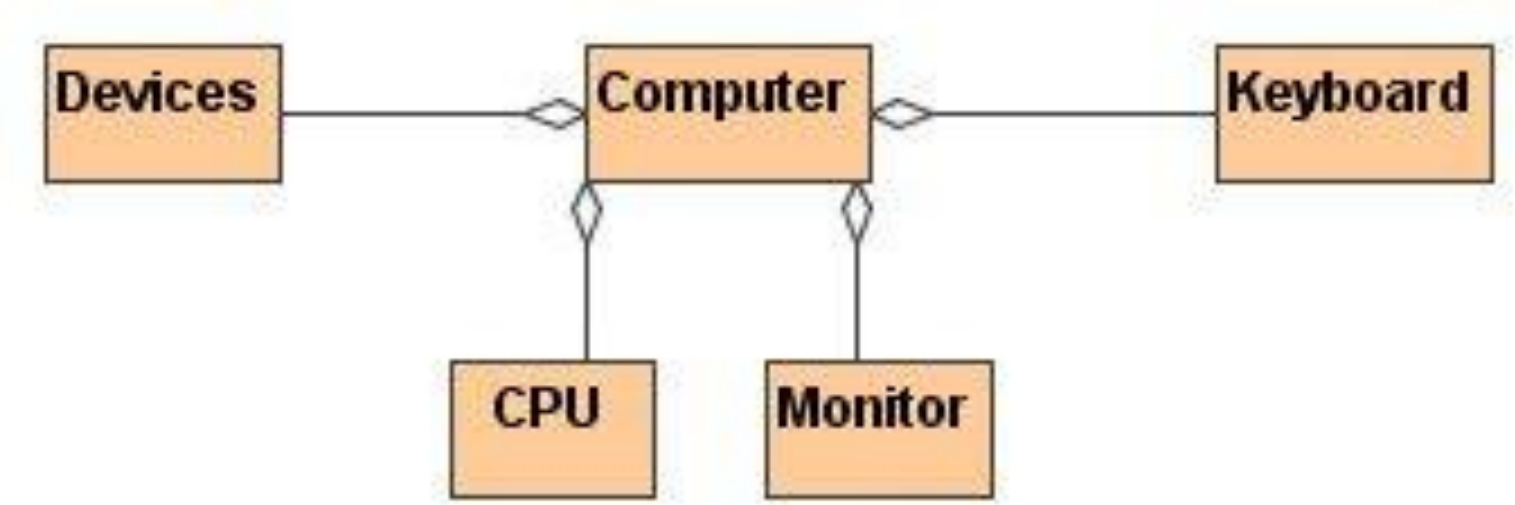
- **A** is a physical or logical part of **B**
- **A** is a kind of **B**
- **A** is contained in **B**
- **A** is a description of **B**
- **A** is a member of **B**
- **A** is an organization subunit of **B**
- **A** uses or manages **B**
- **A** communicates with **B**
- **A** follows **B**
- **A** is owned by **B**

Concept 8: Aggregation

- a restrictive form of "part-of" association
- objects are assembled to create a more complex object
- assembly may be physical or logical
- defines a single point of control for participating objects
- the aggregate object coordinates its parts



Aggregation Example

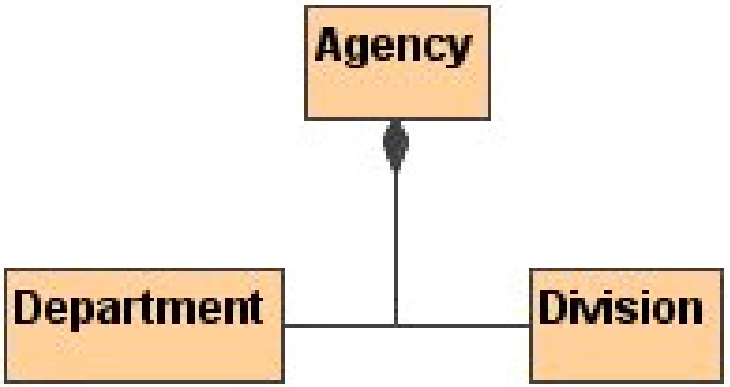


A CPU is part of a computer.

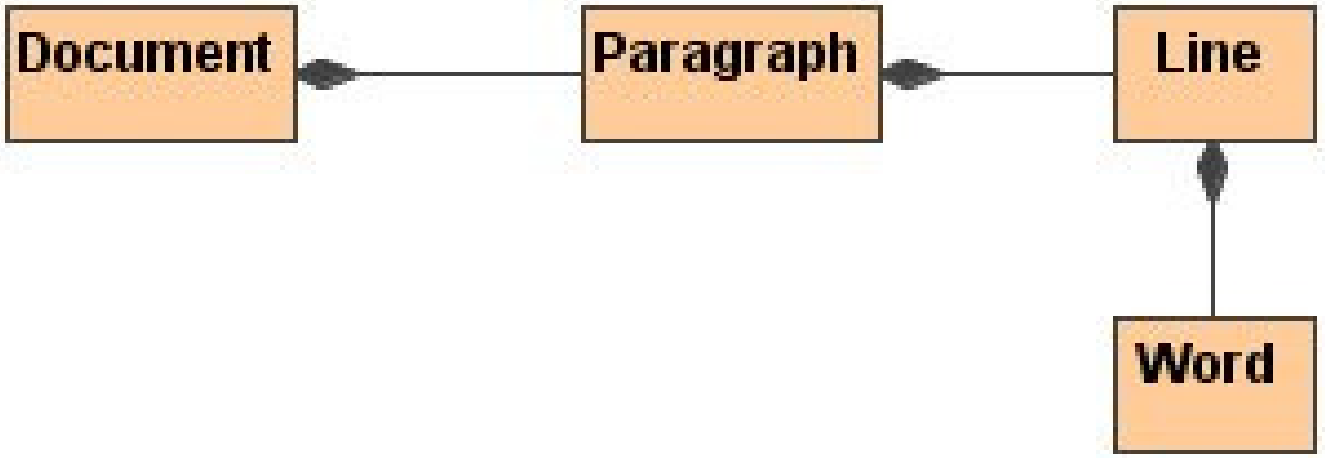
CPU, devices, monitor and keyboard are assembled to create a computer.

Concept 9: Composition

- a stricter form of aggregation
- lifespan of individual objects depend on the on lifespan of the aggregate object
- parts cannot exist on their own
- there is a **create-delete dependency** of the parts to the whole



Composition Example



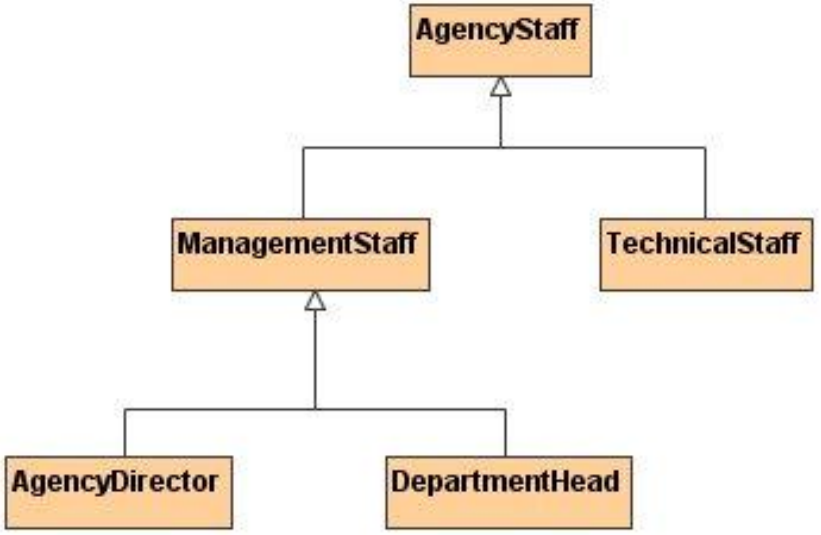
A word cannot exist if it is not part of a line.

If a paragraph is removed, all lines of the paragraph are removed, and all words belonging to that lines are removed.

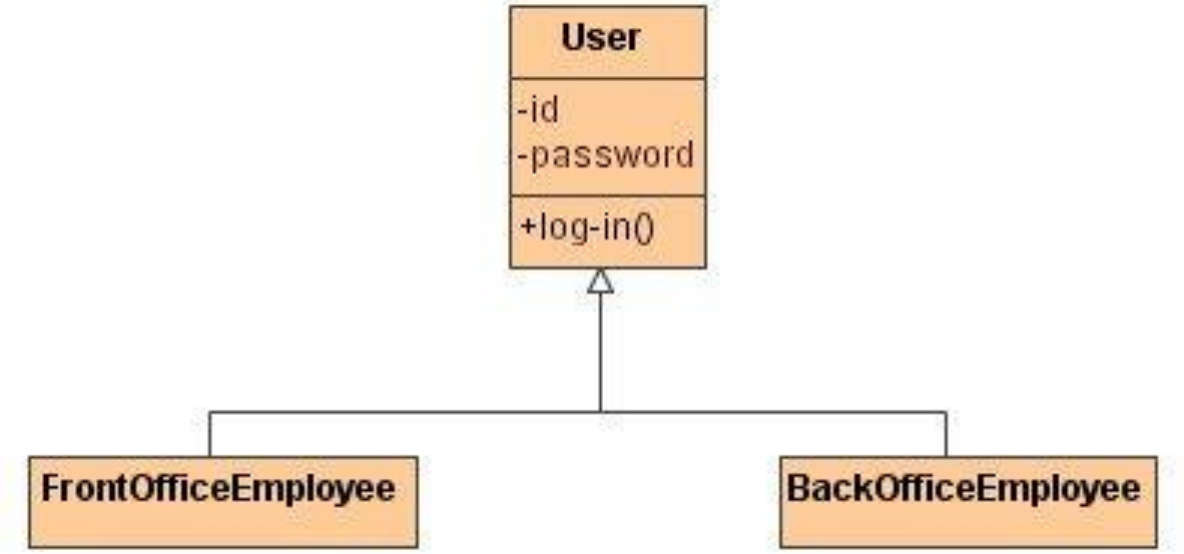
Concept 10: Generalization

- a process of organizing the features of different kinds of objects that share the same purpose
- equivalent to “kind-of” or “type-of” relationship
- generalization enables inheritance
- specialization is the opposite of generalization

not an association



Generalization Example



Common features are defined in User.
FrontOfficeEmployee and BackOfficeEmployee inherit them.

Concept 11: Super-Class

- **Super-Class** is a class that contains the features common to two or more classes.
- A super-class is similar to a superset, e.g. agency-staff.

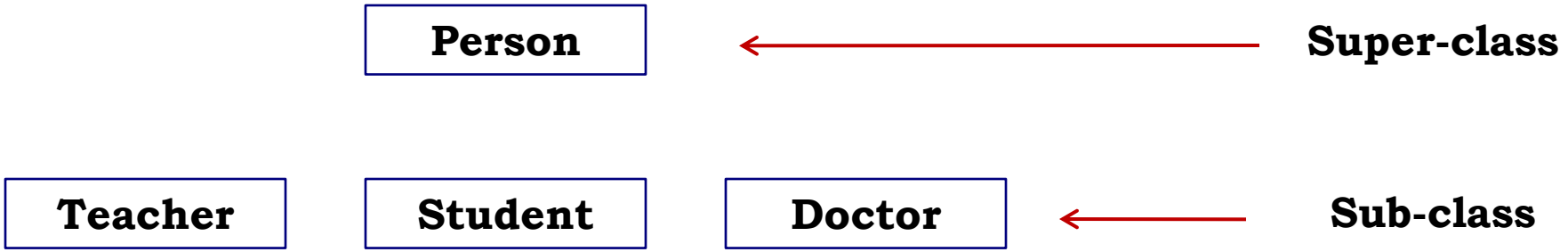
Concept 12: Sub-Class

- **Sub-Class** is a class that contains at least the features of its super-class(es).
- A class may be a sub-class and a super-class at the same time, e.g. management-staff.

Superclass and Subclass

- A superclass results from *generalizing* a set of classes.
- A subclass results from *specializing* a superclass.
- The relationship among superclasses and subclasses is called class hierarchy.

Super/Sub-Class Example



Abstract and Concrete Classes

- Classes that can be instantiated into actual (real *or* virtual) objects are called **concrete** classes.

- Classes that cannot be instantiated into actual (real *or* virtual) objects are **abstract** classes.

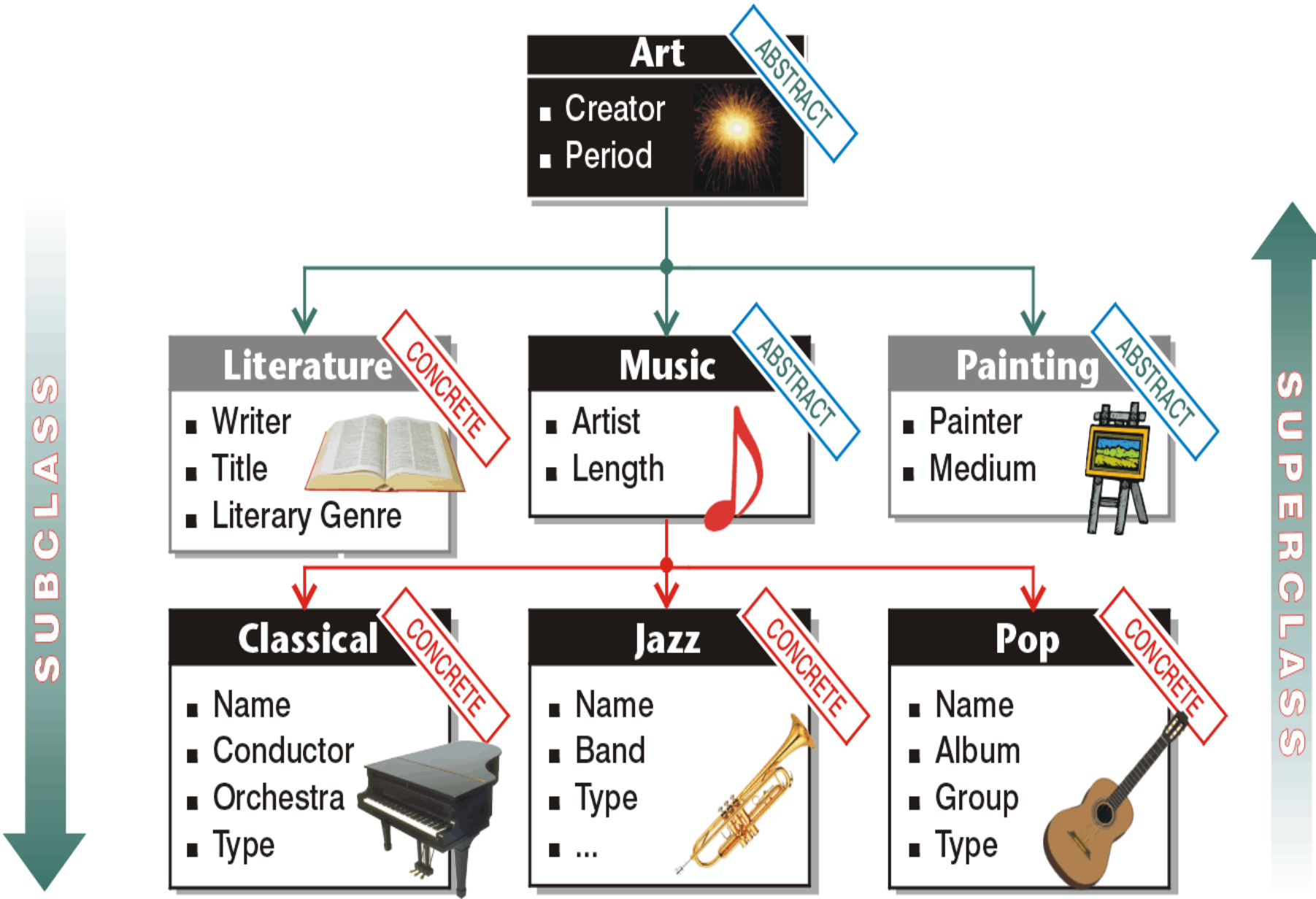
Concept 13: Abstract Class

- a class that lacks a complete implementation – provides operations without implementing some methods
- cannot be used to create objects; cannot be instantiated
- a concrete sub-class must provide methods for unimplemented operations

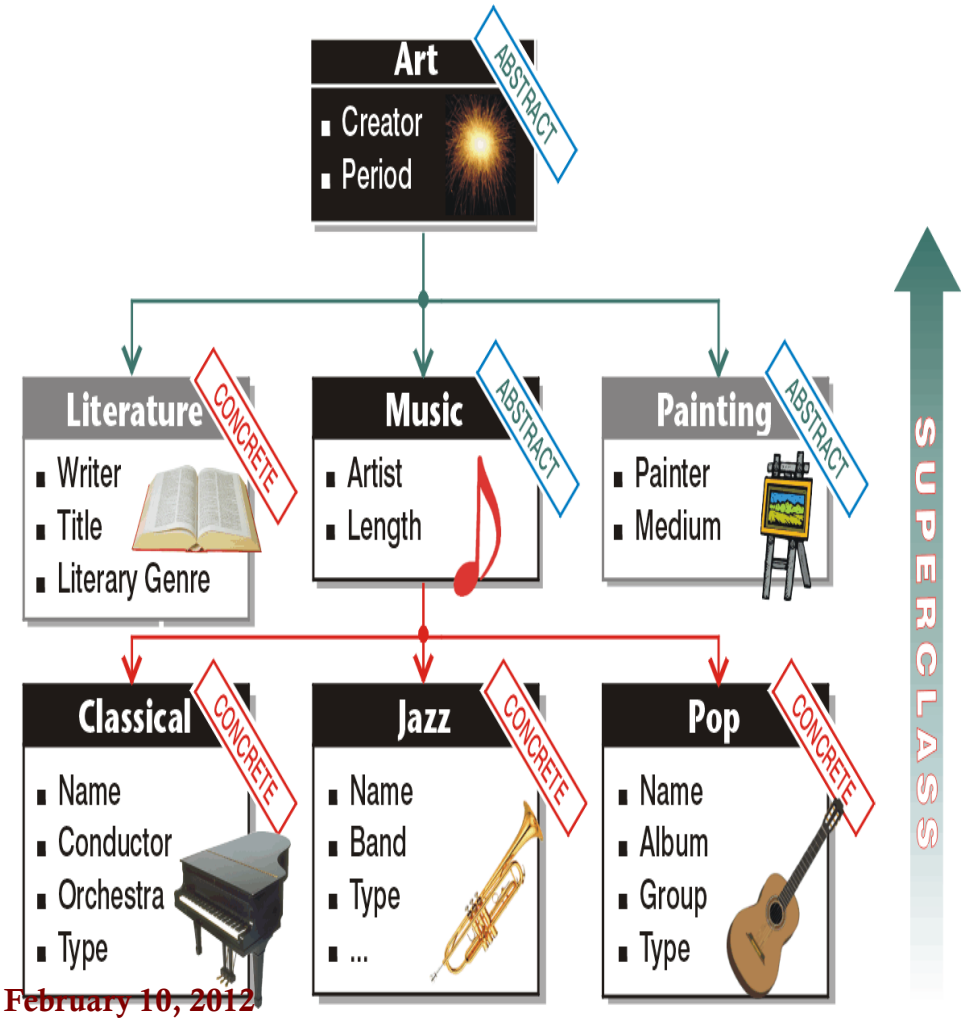
Concept 14: Concrete Class

- has methods for all operations
- can be instantiated
- methods may be:
  - defined in the class or
  - inherited from a super-class

Abstract & Concrete Classes

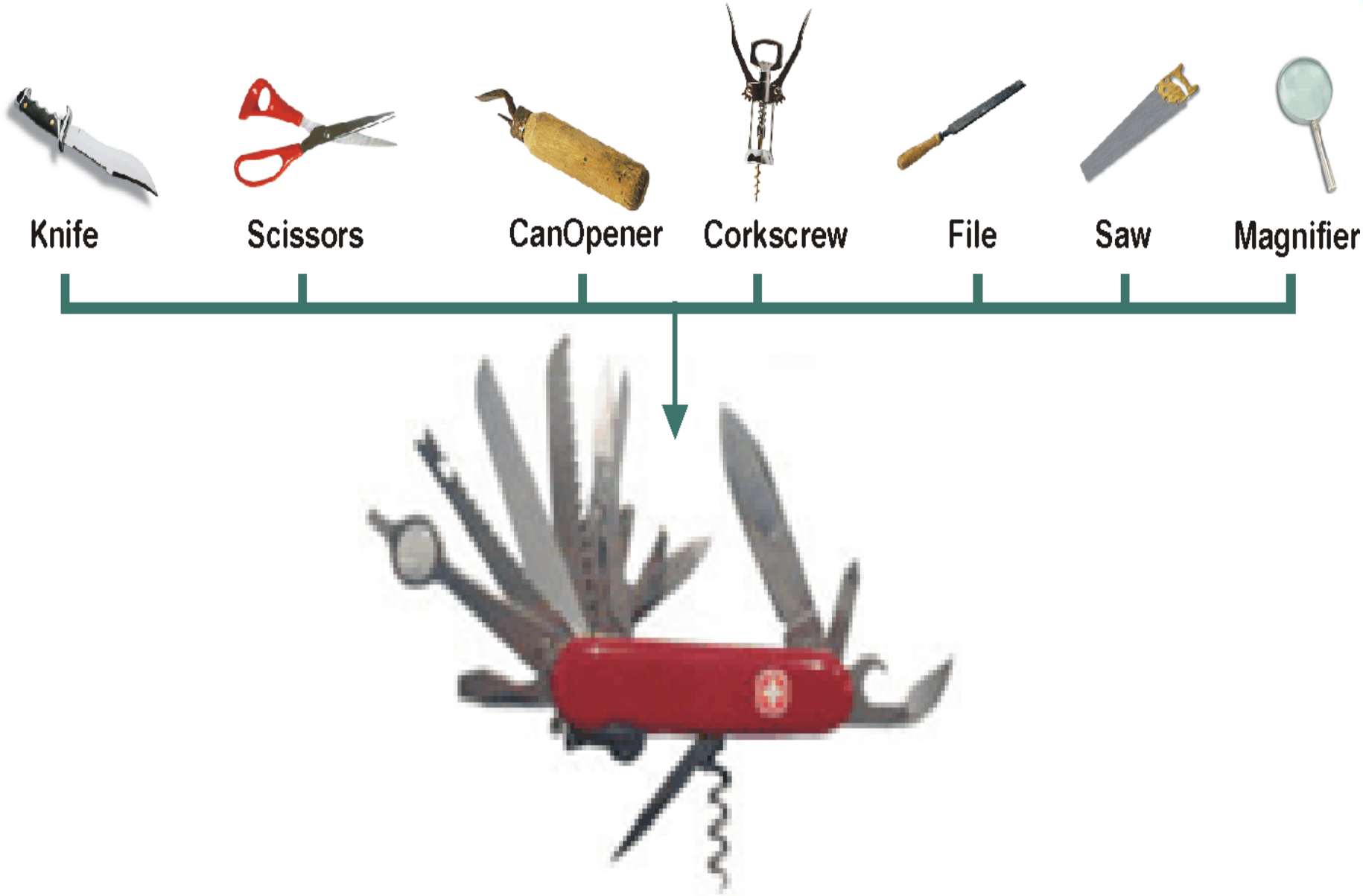


Inheritance



- Inheritance is the mechanism by which a subclass incorporates the behavior of a superclass.

Multiple Inheritance



Concept 16: Polymorphism

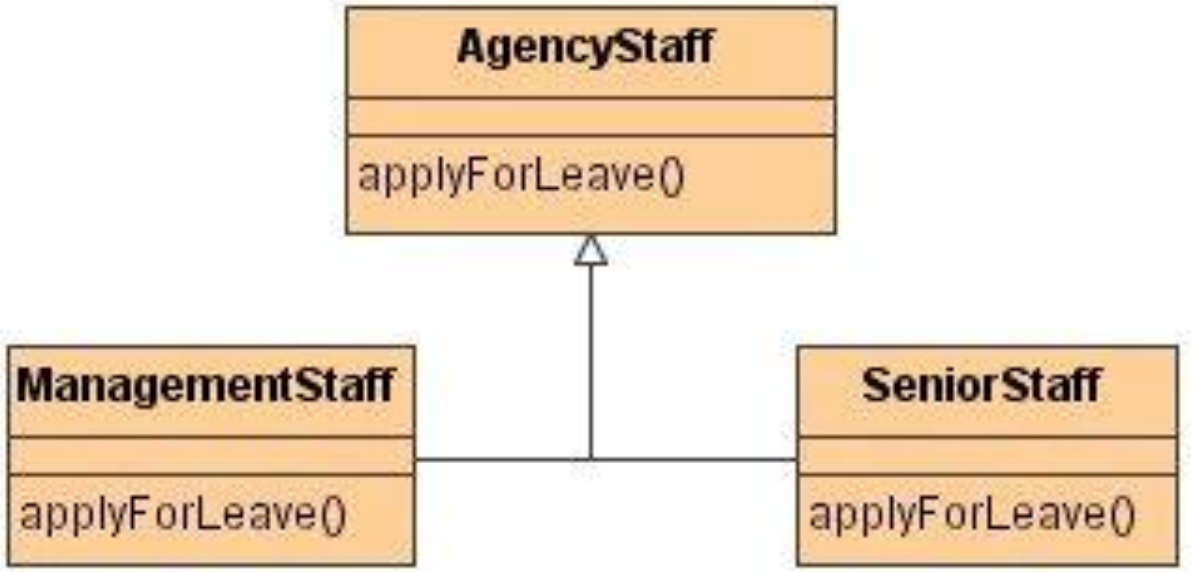
- **Polymorphism** is the ability of objects belonging to different classes to perform the same operation differently.
- facilitated by encapsulation and generalization:
  - **encapsulation** –

    separation of interface from implementation
  - **generalization** –

    organizing information such that the shared features reside in one class and unique features in another
- Operations could be defined and implemented in the superclass, but re-implemented methods are in unique sub-classes.

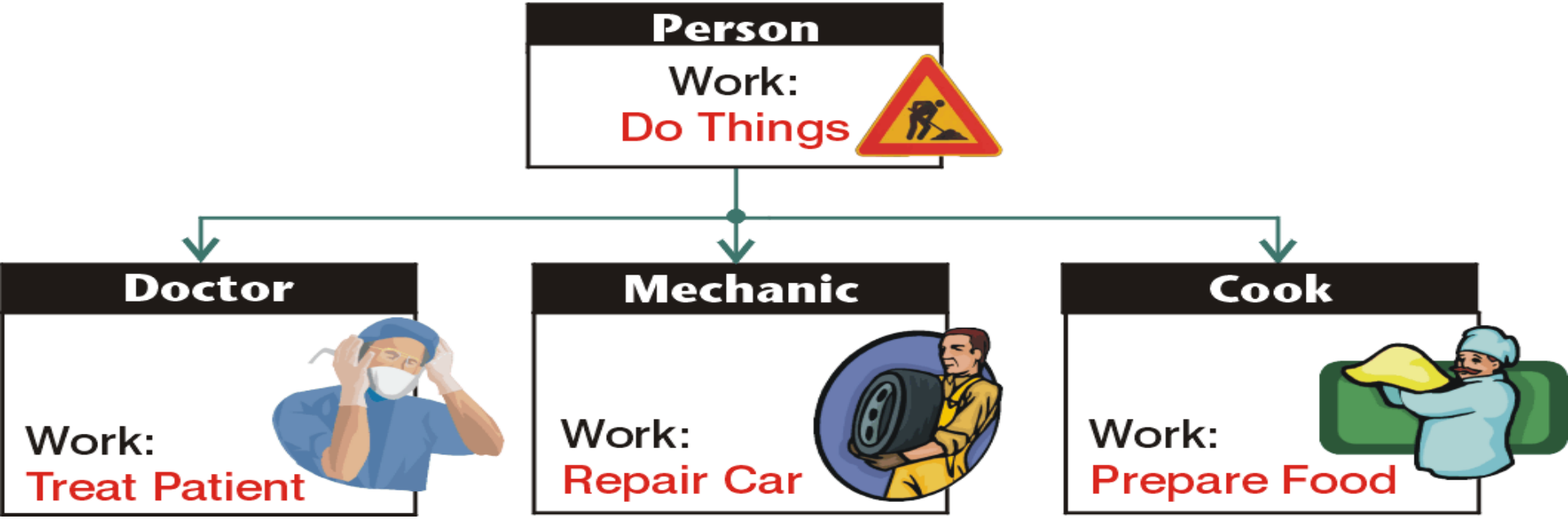
Polymorphism Example

- Many ways of doing the same thing!
- Example: management-staff and agency-staff can apply for leave, but possibly in different ways.



Polymorphism

Same Operation, Performed Differently



Object-Oriented Technology

- Object-oriented technology is a response to the ever-increasing demand for complex information systems. It has become possible by the immense leaps achieved by the information technology.

Object-Oriented Languages

- Simula
- Smalltalk
- C++
- PowerBuilder
- visual Basic
- Java
- .Net

Object-Oriented Modeling

- Object-oriented analysis and design is using an object-oriented approach to building conceptual and logical models of the system.

The Unified Modeling Language (UML)

- UML is a modeling *language* for object-oriented system analysis, design and deployment.
- UML is *not* a product, nor is it a process or a methodology.

The Unified Modeling Language

- The Unified Modeling Language is a language, that provides the "**primitives**" (or the basic elements) for building object-oriented **conceptual** (analysis) and **concrete** (design) models.

UML Supports Multiple Views of Same System

- **Owner's View**
  - what the owner (or business) wants, or the ***conceptual* view** of the system.
- **Architect's View**
  - how the architect conceives the solution, or the ***logical* view** of the system.
- **Builder's View**
  - the blueprints for building the product, or the ***physical* view** of the system.

UML Embodies Four Properties

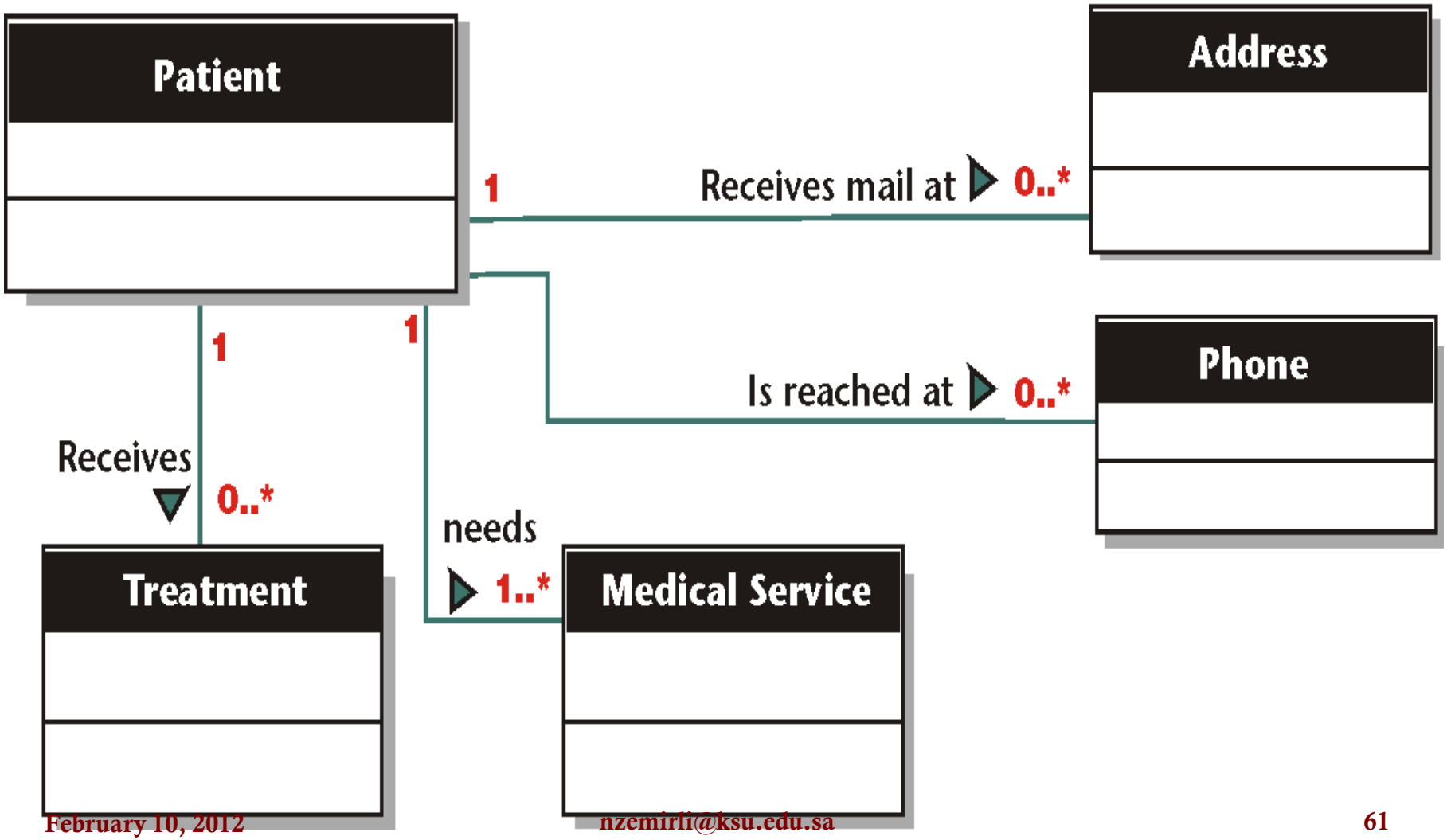
1. Visualization
2. Specification
3. Construction
4. Documentation

Visualization: UML Diagrams

- UML provides a set of graphical elements that are combined to form diagrams. Each diagram is a visual presentation or *view* of the system and satisfies one or *more* broad but overlapping types of modeling:
  - **Behavioral**
    - modeling represents the interaction of the system with the outside world.
  - **Structural**
    - modeling represents the components of the system and their interrelationships.
  - **Dynamic**
    - modeling represents how the components of the system interact with the outside world and with each other to satisfy the behavioral requirements of the system.

A UML Diagram

Modeling Object-Oriented Concepts



Specification, Construction, and Documentation

- Specification
  - UML provides precise and complete models for the three major activities of system development: analysis, design, and implementation.
- Construction
  - UML models are compatible with object-oriented languages.
- Documentation
  - UML modeling tracks major development activities throughout the system lifecycle.

Next: Methodology

- In the next chapter, we shall discuss methodology. We shall also argue that an iterative and object-oriented approach, combined with modeling, offers the best hope for software development projects.