# SYS466:
# Analysis and Design using OO Models

Lecture 2: Domain Model

# What is a domain model?

- A domain model is a visual representation—a picture--of the concepts or physical things that are involved in the domain we are modeling.
- The domain might be a business process, an engineering process...something that we are hoping to automate with a software system.

# Concepts in a domain model

- We identify these things—concepts—using a variety of techniques.
- We might simply identify the concepts from the information we are given in some form of requirements; we might look at common business patterns; or we might draw them from our experience, and things we “know”.
- Examples of concepts in a school environment might be Student, Course, and Assessment.

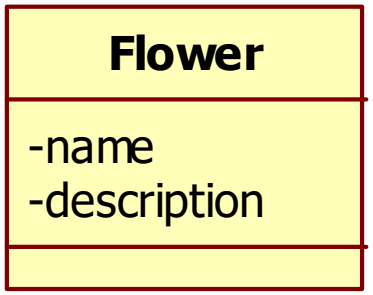
# Concepts as Classes

- Concepts we identify are drawn as classes in a UML domain class diagram.
- At this point these classes are just “ideas”; they are not software classes—some may turn into software classes and some may not.
- Some may become classes that hold data (attributes), others may become controllers that manage the data classes and interactions with other domains.
- As we identify classes we look for potential properties—we call these attributes. Attributes could become the data that a class carries and is responsible for.

# UML Domain Class

- Represents some concept—physical or electronic or recorded—that is used, and probably needs to be remembered, in a domain such as a business.
- The class defines the “template” for the concept; specific instances are objects instantiated from the class template. For example the objects Easter Lily, Tea Rose, and Tulip might be objects of the class Flower
- The class defines the “template” for the concept; specific instances are objects instantiated from the class template. For example the objects Easter Lily, Tea Rose, and Tulip might be objects of the class Flower

# UML Domain Class *continued*




Objects:

- name “Easter Lily”, description “a white, spring flowering lily“
- name “Tea Rose”, description “a garden rose with flowers having a subtle tea scent”
- name “Tulip”, description “a spring flowering plant that grows from a bulb and has cup-shaped flowers. It is from the lily family”

# Objects and Classes: Review

# Objects and Classes

- Objects interact with each other in order to carry out the logic in scenarios in use cases
- Classes define the behaviour of the objects and the attributes (properties) of the objects

# Objects and Classes

- An object has state, behaviour, identity
- A class is a “template” that defines behaviour and structure of similar objects
- Object = instance of a class

# Objects

- Include procedures and data; they perform operations and save local state (the “attributes” and their values ).
- An object can:
  - Change state (the value of its attributes)
  - Behave
  - Be manipulated
  - Relate to other objects

# State

- All of the properties of an object plus the current values for each property.
- The value of a property might be simple or it might be another object
- Example:
  - Customer has a first name—a simple attribute with a value e.g. Barb.
  - Customer has an address—a complex attribute which is actually an object of the Address class—it has number, street, suite number, city, province, etc.

# Behaviour

- Defines how an object acts—how it changes state, how it passes messages (e.g. Returns a value of a property), how it reacts.
- Object behaviour is defined by operations:
  - Java—methods
  - C++--member functions
  - SmallTalk—passing messages
- Behaviour can affect the state of an object

# Operations define Behaviour

- Common operations
  - Select (get)
  - Modify (set)
  - Iterate (get all)
  - Construct (instantiate)
  - Destruct (destroy)
- Protocol: collectively, all of the operations of an object.

# Identity

- The property that distinguishes an object from all others
- Not the same as equality
- Not typically visible to external users; dependent on language:
  - E.g. each object has its own region of memory. The object identifier is essentially a pointer (C++)
- Assigned at object creation (instantiation)

# Drawing a Domain Model

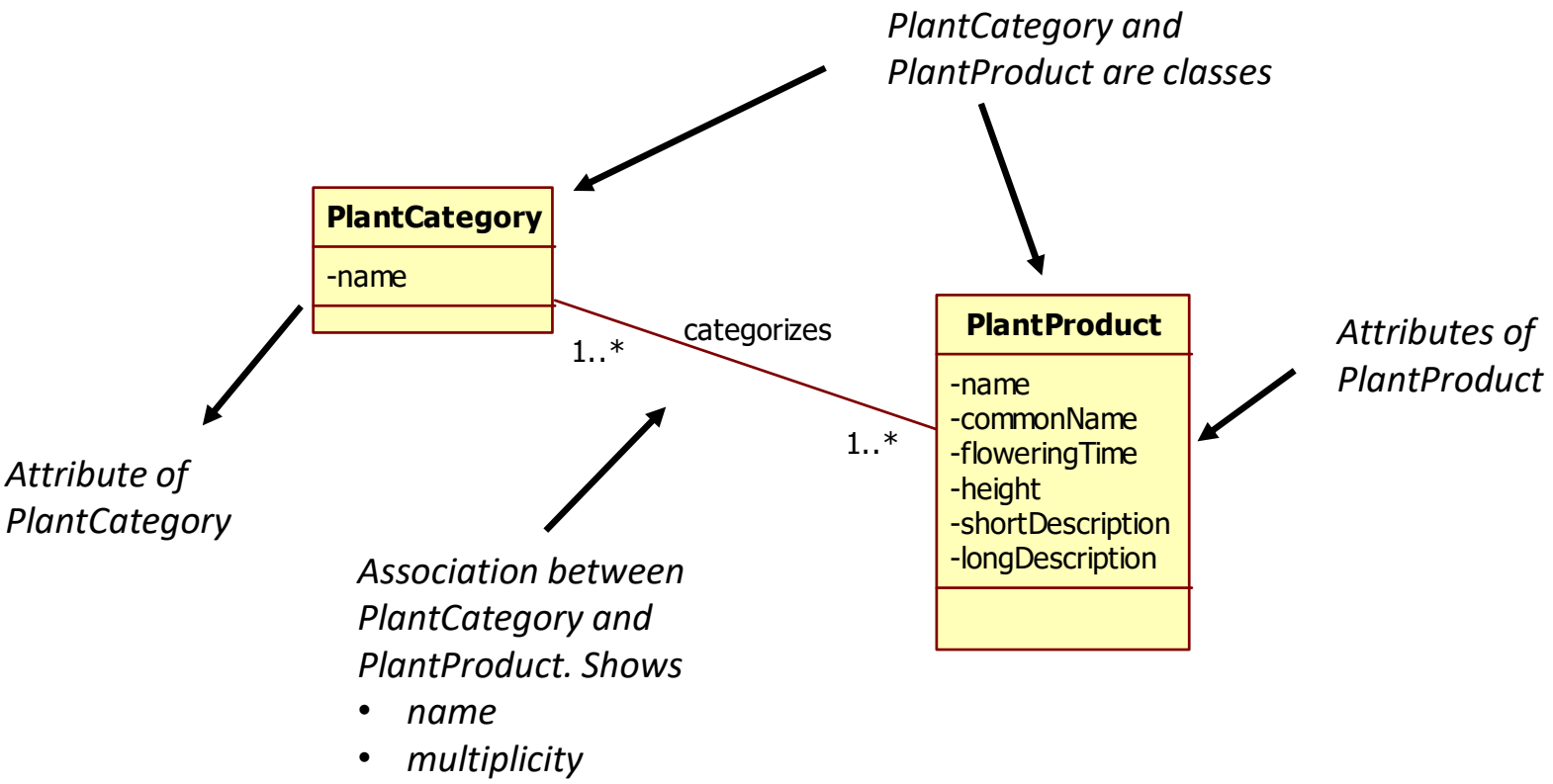
# Components of a Domain Model

- Conceptual classes
- Associations between these classes including name and multiplicity
  - Associations should be an indication of a “meaningful and interesting” connection
  - Assocation name describes the nature of the relationship between the classes
  - Multiplicity indicates the number of objects of one class that can be related to an object of another class at any point in time.
- Attributes of these classes

# What does not belong in a domain model

- Things that do not belong in a domain model:
  - Software artifacts such as a page, window, database *(unless we are modeling a software domain)*
  - Methods/functions
- A domain model is NOT a data model—domain models can include classes that are not persisted.

# Domain Classes -- Example



# How do we find classes?

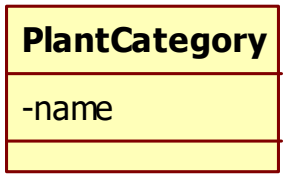
- We look for things that represent business concepts or significant concepts in the system being modeled:
  - Identify nouns or noun phrases
  - Speak with business domain experts
  - Look for known patterns
  - Find descriptions
- We often start with written scenarios

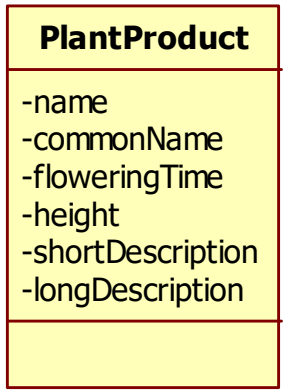
# Examples of Finding Classes

# Example 1

Scenario: Display Plant Product

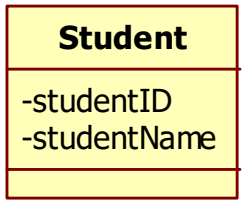
| Actor (Manager) | System |
|---|---|
| Requests to view plant categories | Displays a list of available categories—shows category name for each. |
| Selects a category | Displays list of plant products in that category. For each shows name and short description |
| Selects a plant product | Displays name, common name, flowering time, height, and short and long descriptions for the product. |



# Example 2:

| Actor | System |
| --- | --- |
| Requests student 123 | Retrieves and displays name of student 123 |
| Requests to see course sessions that student 123 is enrolled in | Retrieves and displays course sessions (course session id's) that student 123 is enrolled in |





*Note: we can assume that "123" is the value of some kind of student identifier*

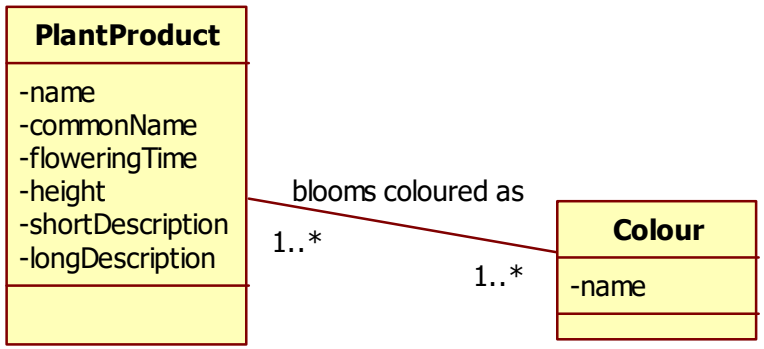
# Differentiating between classes and attributes

# Class or Attribute?

- Primitive data type?
  - Probably an attribute
  - e.g. Patient age is an integer; probably an attribute
- Complex, with perhaps its own attributes?
  - Could be a class
  - e.g. Address has number, street name, unit name, and so on
- Do we want to reuse it?
  - Could be a class
  - e.g. Address could be used across a system or systems

# Class or Attribute? *(continued)*

- Multiple occurrences (even though each might be simple)?
  - Probably a class
  - e.g. a PlantProduct can have many colours of flowers. Each colour is just a string but a plant can have many of them.



# In-class Exercises

# Identify the classes and attributes in the following written descriptions

## Exercise 1: Help Desk

A customer calls in to the help desk to report a problem. The help desk operator creates a problem ticket for the problem including date logged and problem description. The system assigns a problem ticket number and the operator gives that number to the customer over the phone.

## Exercise 2: Credit Card Payment

An online bank customer wants to use funds from a chequing account to make a payment toward a credit card balance. The customer enters the amount and requests a transfer of funds from the chequing account to the credit card account. The system deducts the amount entered from the chequing account and applies it to the credit card account. The balances of both the chequing account and credit card account are updated to reflect the new total. The system keeps the credit card transfer information including date and amount.

## Exercise 3: Writers and Articles

| Actor (Writer) | System |
|---|---|
| **Requests to submit article** | Requests userID and password. |
| **Enters userID and password.** | Authenticates the writer successfully.<br>Retrieves and displays a list of articles that the writer has submitted in the past.<br>Requests title and content for the new article. |
| **Enters article title. Copies in article content.** | Sets the article submitted date to the current date, the article content to the entered content, the writer to the writer entering the content and saves the article.<br>Requests entry of any other writers that worked on this article—displays a list of all writers. |
| **Selects a writer and requests to add.** | Adds the writer to the article and saves. |
| **Repeats the above row until done** | Persists article and writer information to the database. |