03 Classes

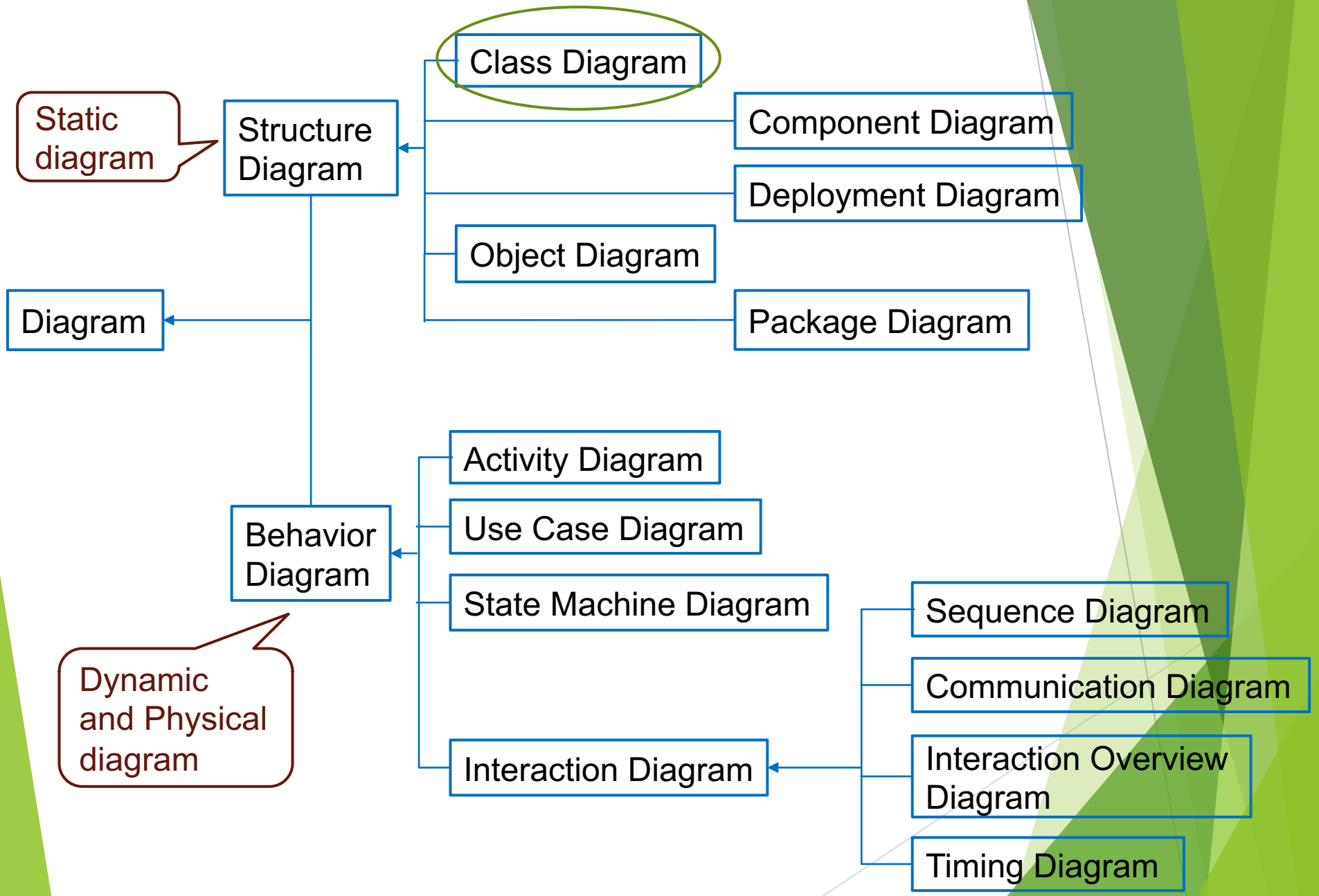
Introduction to OOA OOD and UML

2022 Spring

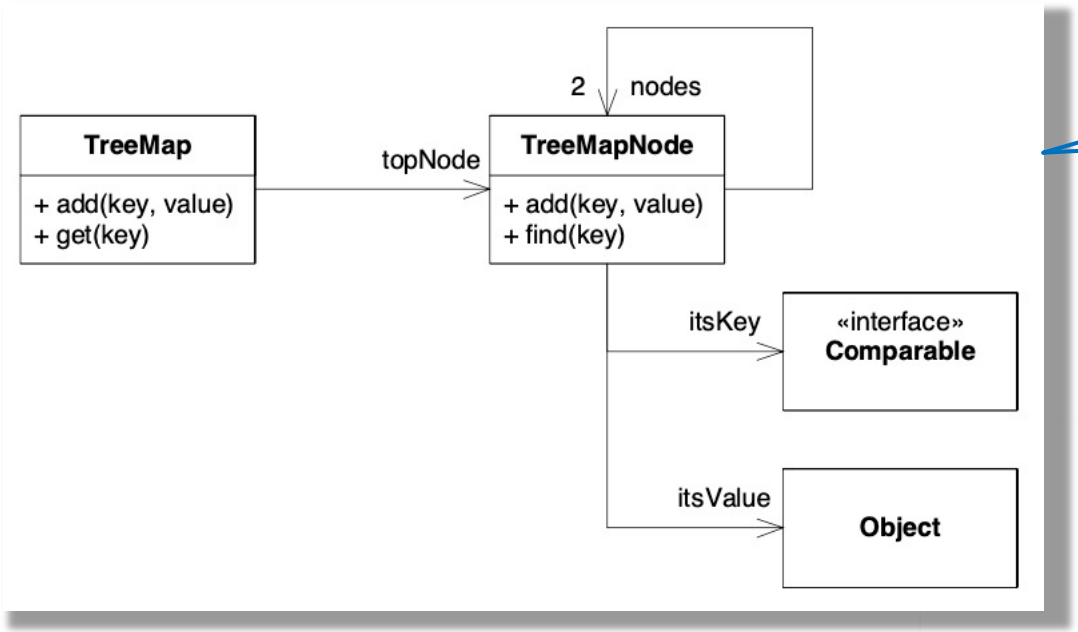
College of Information Science and Engineering

Ritsumeikan University

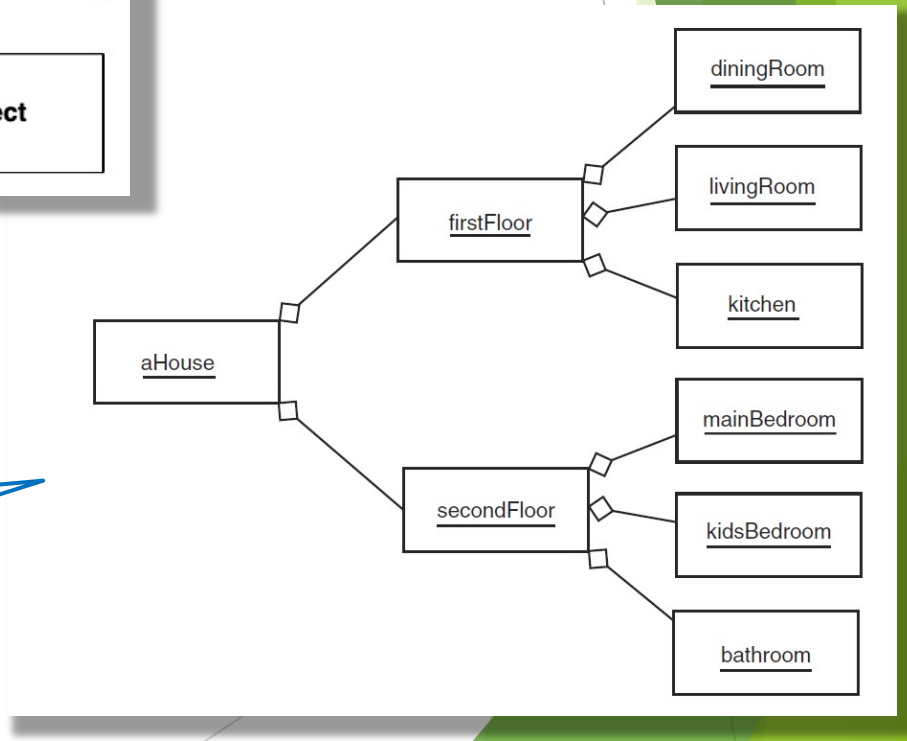
Yu YAN


Class Diagram
Static diagram
Structure Diagram
Component Diagram
Deployment Diagram
Object Diagram
Diagram
Package Diagram
Activity Diagram
Behavior Diagram
Use Case Diagram
State Machine Diagram
Dynamic and Physical diagram
Sequence Diagram
Communication Diagram
Interaction Diagram
Interaction Overview Diagram
Timing Diagram

Object and Class Diagrams



A class diagram



An object diagram

Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

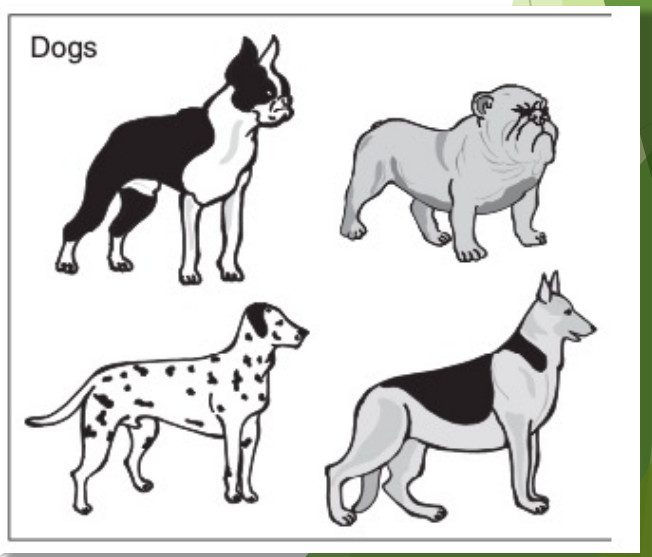
What Is a Class ?

- A class encapsulates characteristics common to a group of objects.
- Biological analogy:

For example, a set specifies what featurs its member objects will have

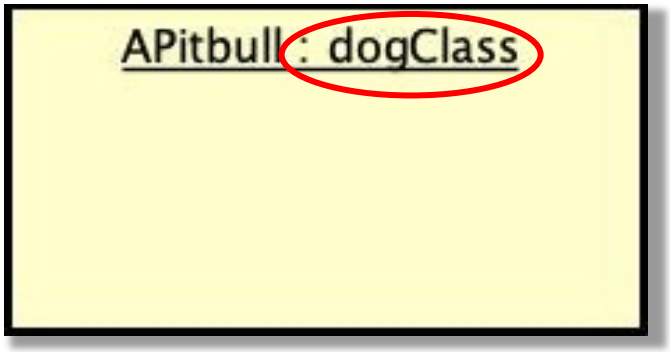
- In programming, a class is consisdered as a template or set of instructions to build a specific type of object.

In programming, every object is built from a class



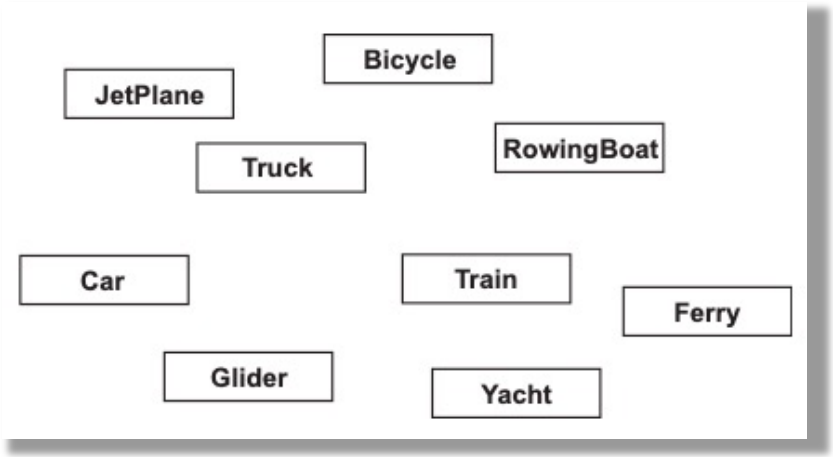
Present a Class in An Object Diagram

- Concrete (specific) object “aPitbull” is derived from (specific) class “dogClass”.



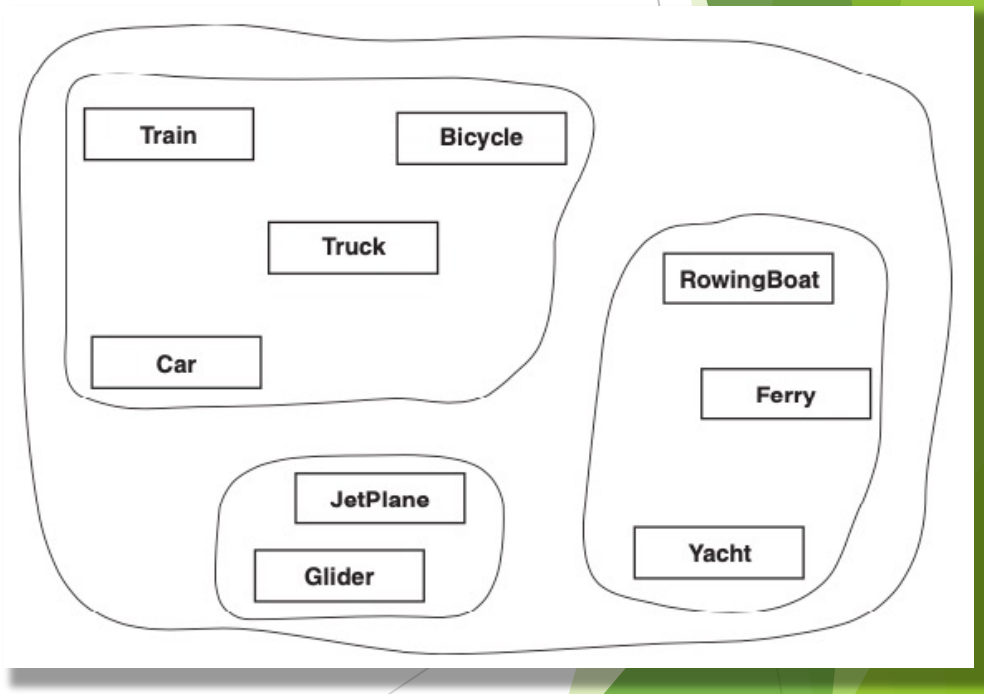
Classification

- Classification is grouping things into classes. For example:



Some classes

Classification



Groups of classes

Outline

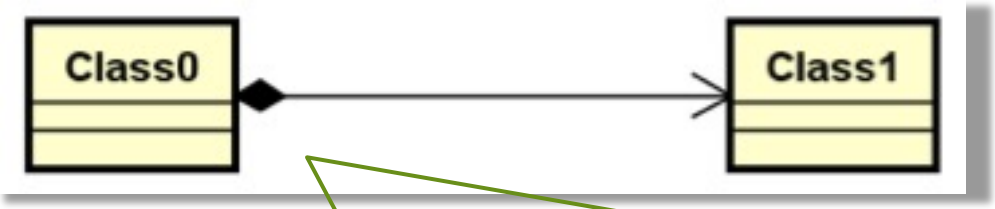
- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

Review – Object Connection Types

- Between objects, we have already known two types of connections:
  - Associations
  - Aggregations
- Links can be navigable/non-navigable

Another Object Connection Type

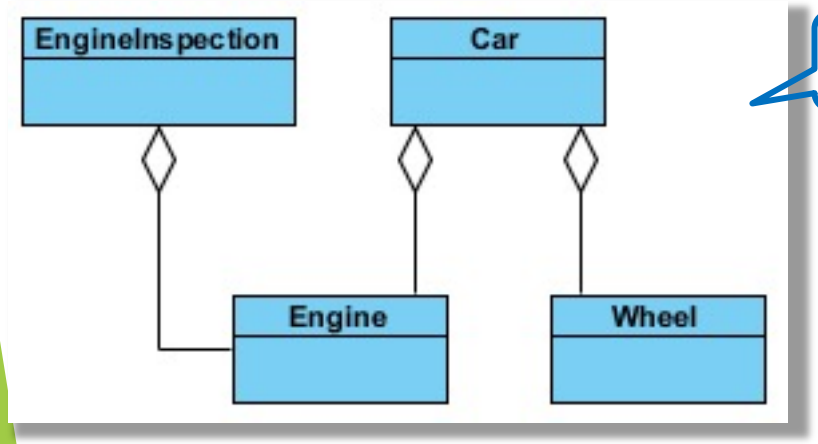
- A composition in UML is a special case of association that describes a relationship between a whole and its **existential** parts.
- In a composition, a part can never be larger than the whole
- Composition is also called "death relationship"



- ✓ For example: A room belongs to one building and not to several buildings at the same time.
- ✓ The whole determines the life cycle of the parts.

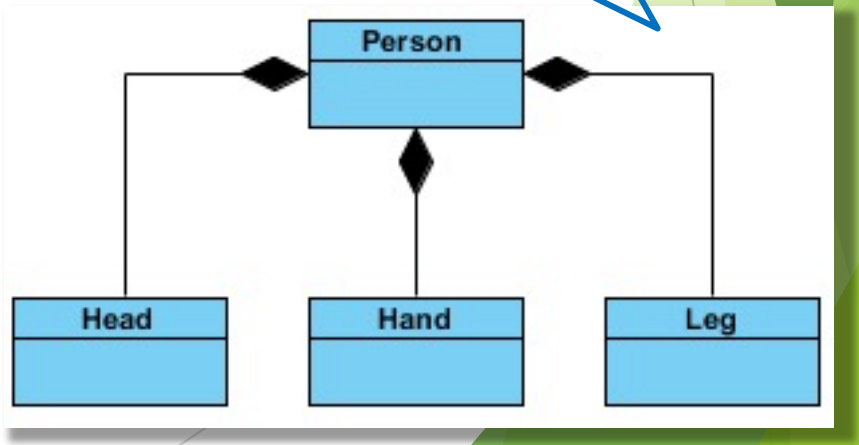
Association VS Aggregation VS Composition

- Aggregation and composition are subsets of association
  - ✓ Aggregation implies a relationship where the child can exist independently of the parent
  - ✓ Composition implies a relationship where the child cannot exist independent of the parent



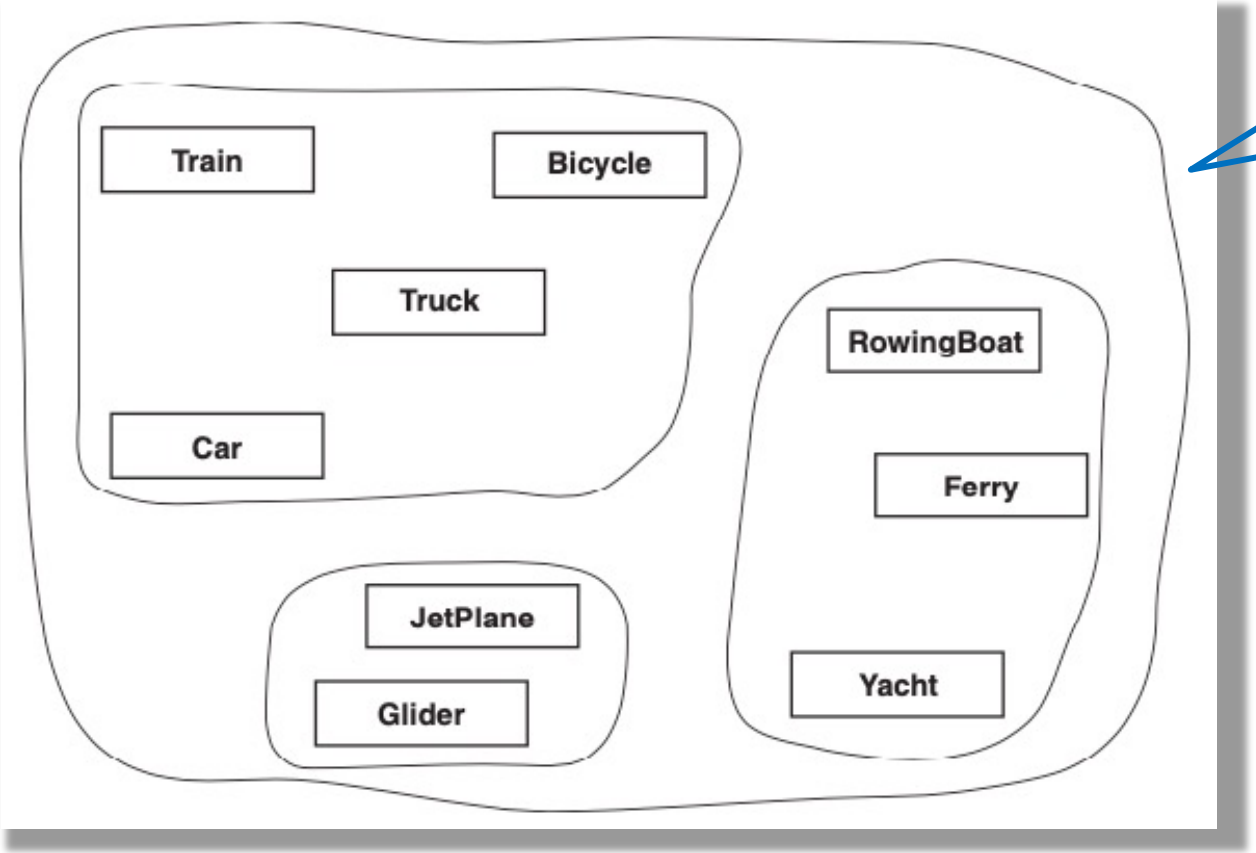
Aggregation

Composition



Class Hierarchy

- We use "hierarchy" to describe relationships among classes

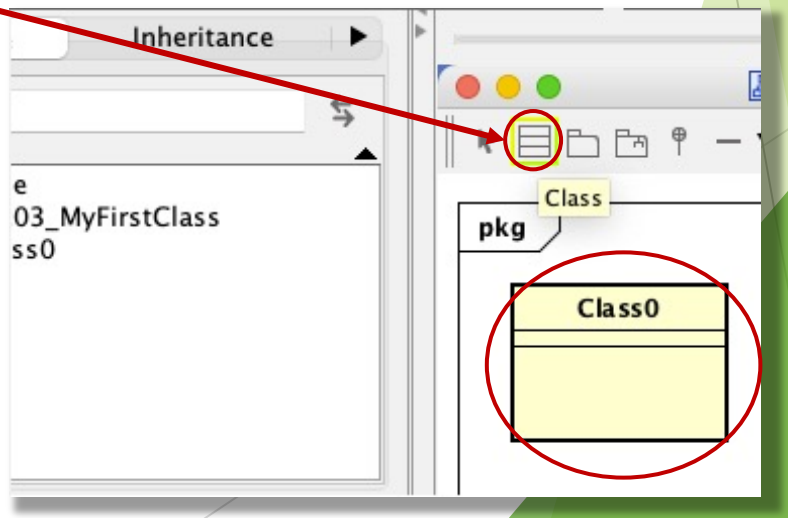


Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

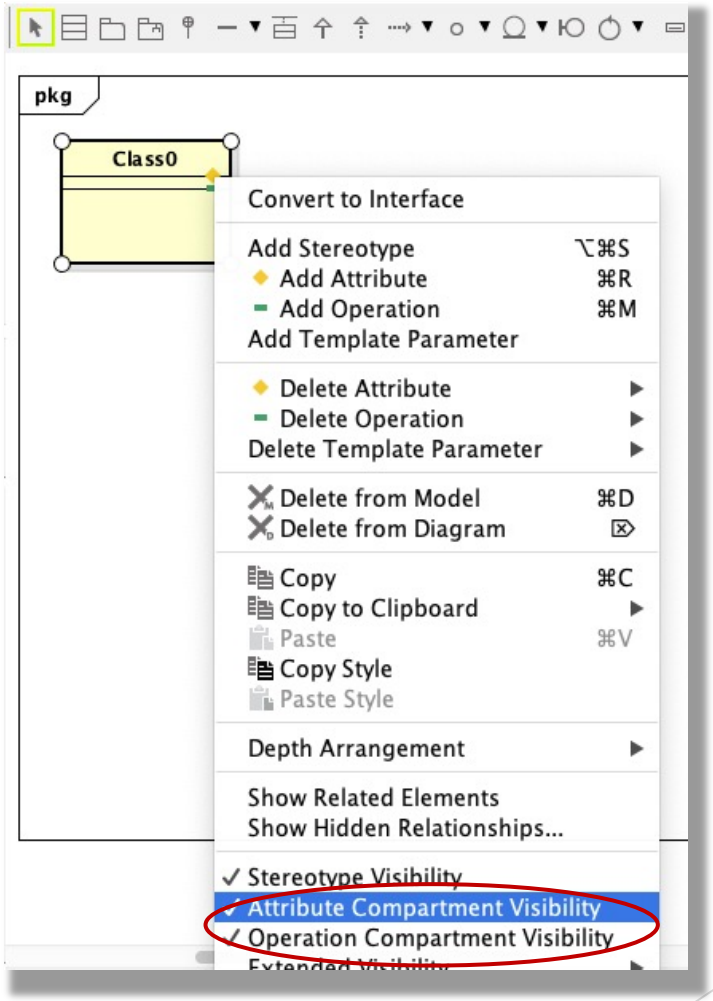
Depicting Classes by Astah (1)

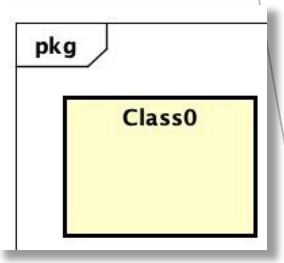
- In UML, classes are drawn as boxes on a class diagram
  - Class names (on class diagrams) are not underlined, while object names (on object diagrams) are
  - Class names start with a capital letter. On a class diagram, they are shown in bold
- In terms of using Astah to depict a class, the following steps can be followed: Main Menu --> Diagram → Class Diagram → Click "Class" button



Depicting Classes by Astah (2)

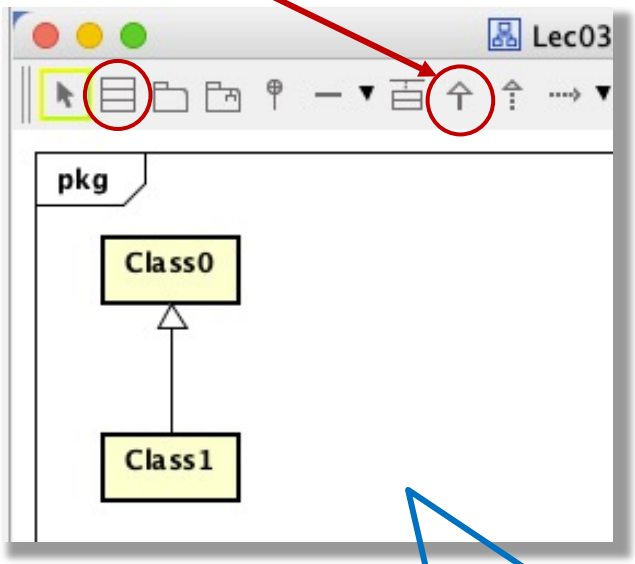
- Right click on “Class 0” → Uncheck “Attribute compartment visibility” and “Operation compartment visibility”.





Depicting Hierarchy by Astah (1)

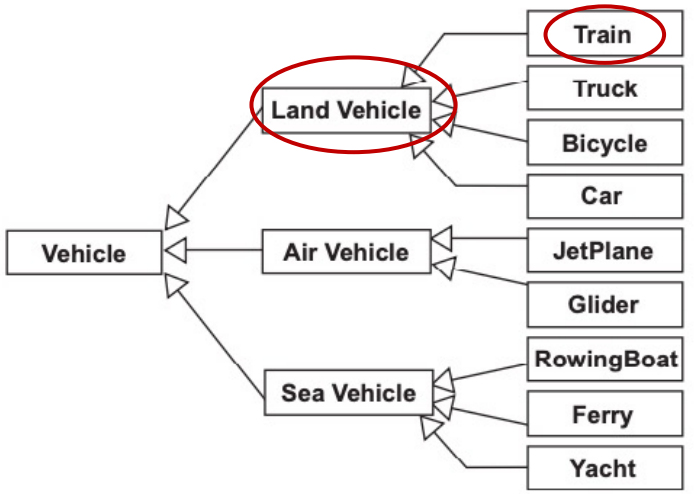
- Arrow with white triangle arrowhead is called generalization
  - Direction of a generalization is from a more detailed concept to a less detailed one



Click generalization button and connect classes

Depicting Hierarchy by Astah (2)

- An example of class hierarchy:
  - Specialization/Generalization: *Train* is more specialized than *LandVehicle*. *LandVehicle* is more generalized than *Train*.
  - Inheritance: *Train* inherits characteristics of *LandVehicle.*
  - Parent/child: *LandVehicle* is the **parent** of *Train*; *Train* is a **child** of *LandVehicle*.
  - Superclass/subclass: *LandVehicle* is the **superclass** of *Train*; *Train* is a **subclass** of *LandVehicle.*
  - Base/derived: *LandVehicle* is the **base** from which *Train* is **derived.**



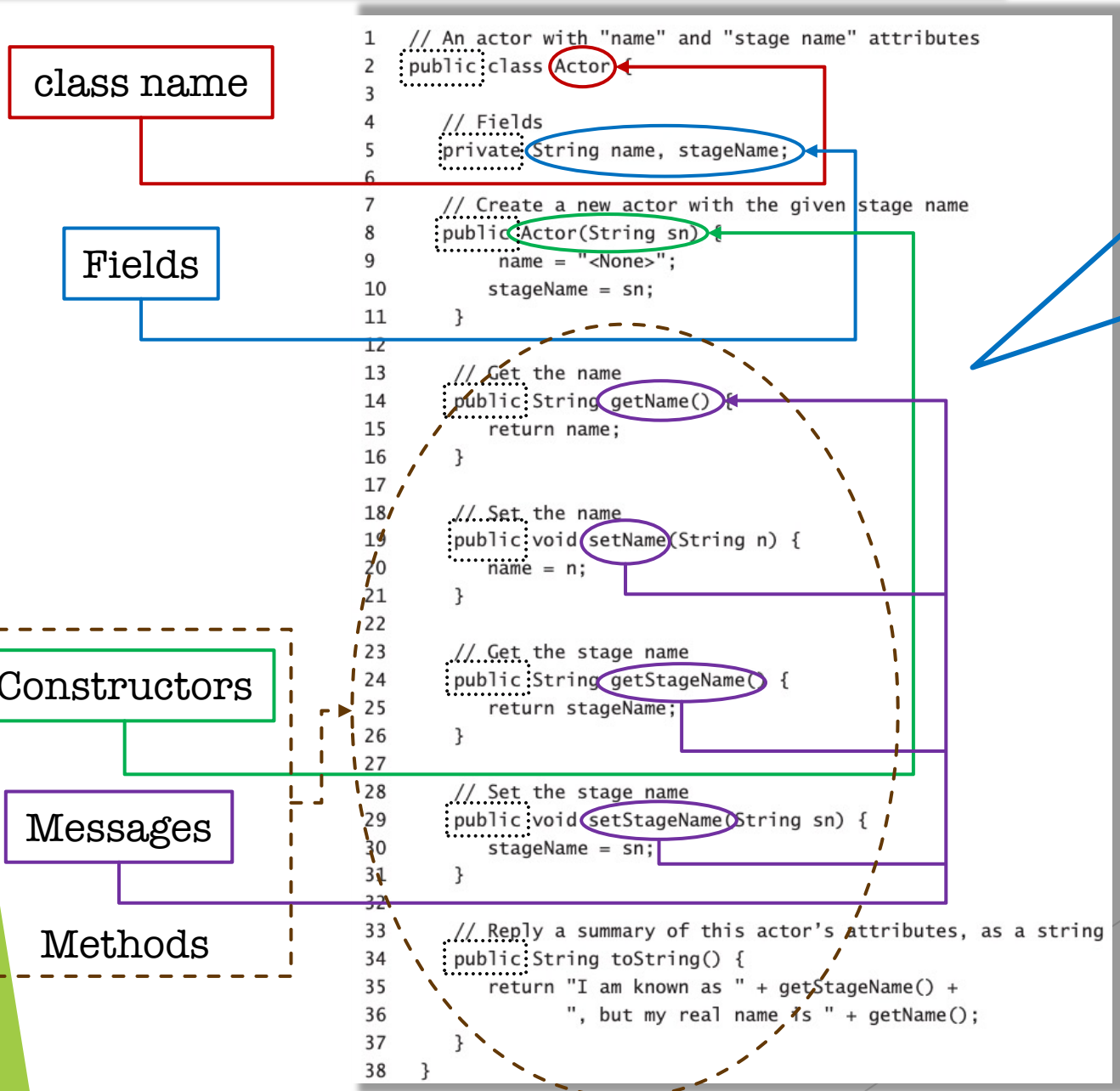
Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

What Does a Class Define? (1)

- Object-oriented (OO) developers use classes to describe the programming elements that particular kinds of object will have.
  - Without classes, we would have to add these elements to every individual object.
- Essential elements of a class in the OO Programming:
  - Class name: for referring to the class elsewhere in our code.
  - Fields: for describing the information stored by this kind of object.
  - Constructors: for controlling initialization of objects.
  - Messages: for providing other objects with a way to use the objects.
  - Methods: for telling the objects how to behave.
  - Comments: for telling programmers how to use or maintain the class (ignored by the compiler).

What Does a Class Define? (2)




```
1   // An actor with "name" and "stage name" attributes
2   public class Actor {
3
4       // Fields
5       private String name, stageName;
6
7       // Create a new actor with the given stage name
8       public Actor(String sn) {
9           name = "<None>";
10          stageName = sn;
11      }
12
13      // Get the name
14      public String getName() {
15          return name;
16      }
17
18      // Set the name
19      public void setName(String n) {
20          name = n;
21      }
22
23      // Get the stage name
24      public String getStageName() {
25          return stageName;
26      }
27
28      // Set the stage name
29      public void setStageName(String sn) {
30          stageName = sn;
31      }
32
33      // Reply a summary of this actor's attributes, as a string
34      public String toString() {
35          return "I am known as " + getStageName() +
36                 ", but my real name is " + getName();
37      }
38  }
```

Each elements can be specified by programmers so that a system can access the elements or not

Usually, an element can be specified as: public(visible everywhere), private(only available to the objects themselves), protect (only available to the "child objects")

Methods

- A method in OOP is a procedure (function) completely defined in a class
  - A method is a named action applied to the object
  - A method may have arguments and may return some specific typs of values
  - It is preferable to get access to fields through methods
  - The method includes the Message and the Constructor
- Message is a request sent from one object to another
  - Methods are associated with messages and an object: they executed when an object receives a message.
- Constructor in a class is a special type of function which means “to create an object”
  - The constructor prepares the new object, often accepting arguments to initialize fields
  - The constructor mostly has the same name as its class

Terminology

- Naming in UML and OOP

UML	OOP
Class name	Class name
Attributes	Fields, member variables, data members, instance variable, property, state
Operations	Methods, member funtions

Members are all components in a class: Fields, Methods, Inner classes and Constructors

Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

Instantiation

- An instance is a specific object built from a specific class
  - An instance is assigned to a reference variable to access all the instance properties and operations
  - The process of making a new instance is called instantiation. The process is typically done via using the "new" keyword in OOP.
  - NEW OBJECT DEFINITION: An object is an instance of a class.

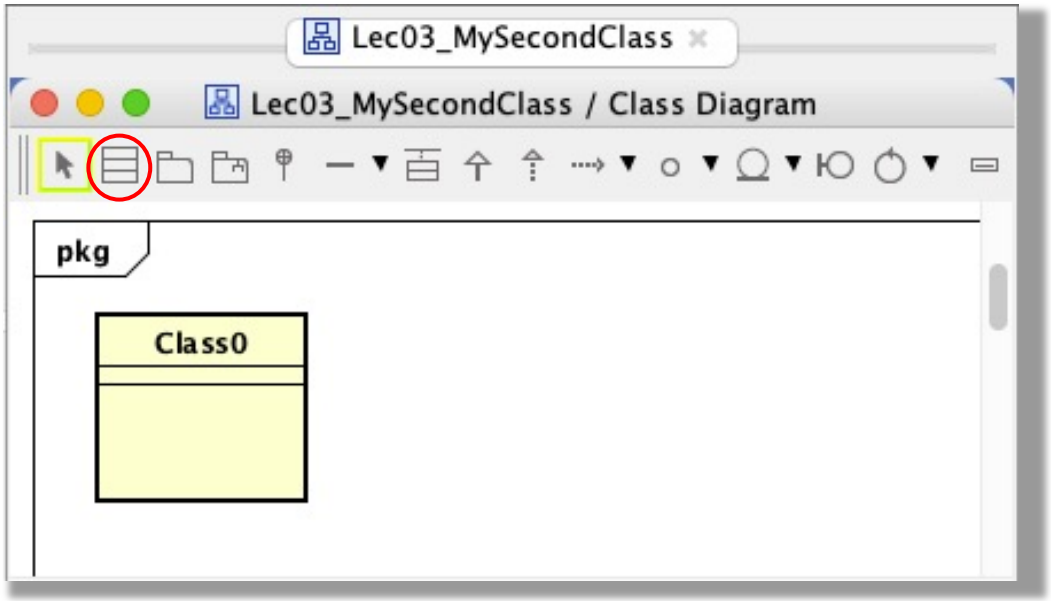
```
aDog = new dogClass ();
anotherDog = new dogClass ();

aDog.weight = 10.0;
anotherDog.weight = 15.0;
```

Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

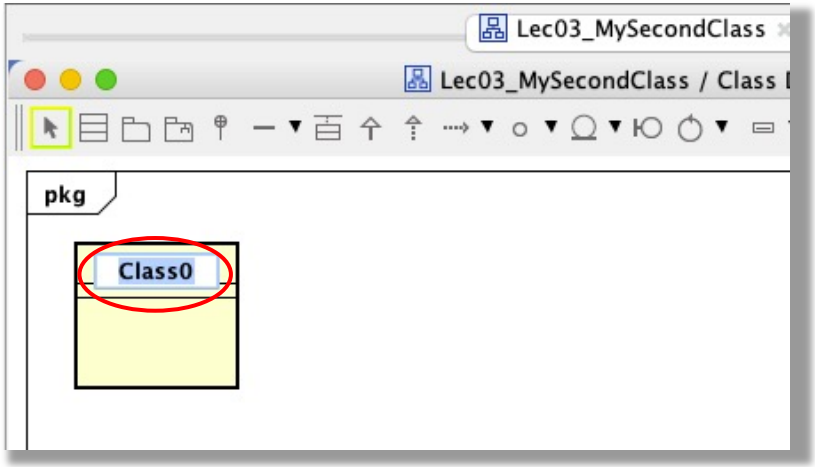
Step 1: Create a Class

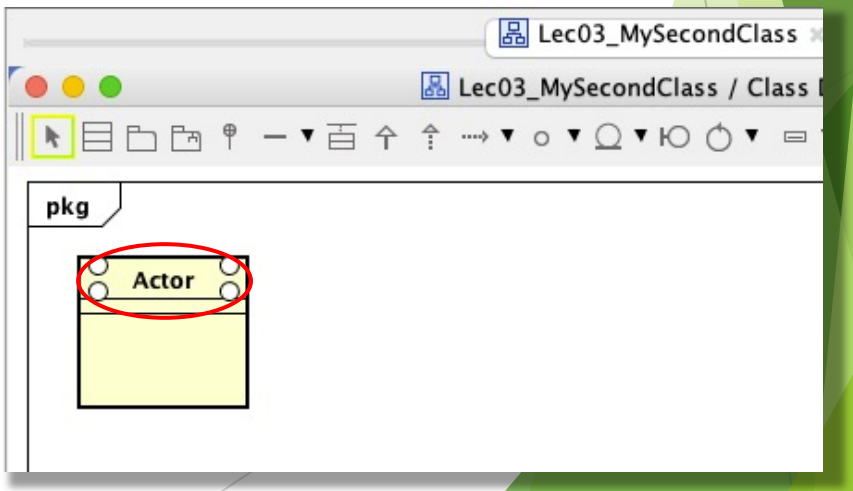
- Left click on “class” button, then click on white diagram field. There are three parts in the class box. (name, attributes and operations).



Step 2: Rename the Class

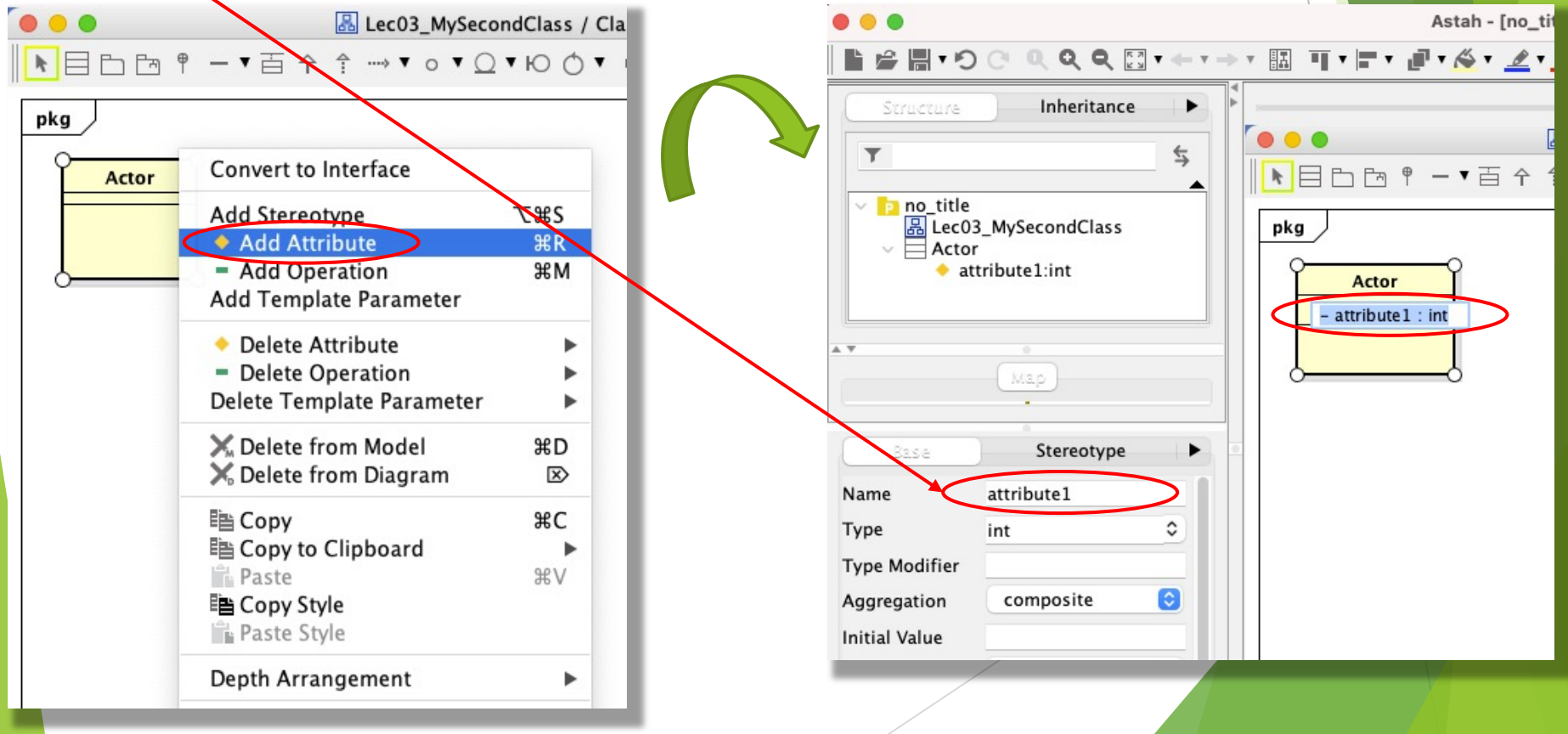
- Double click on "Class0" string (highlighted by blue color), then type in the highlighted area with a new name "Actor", finally press enter to finish renaming.





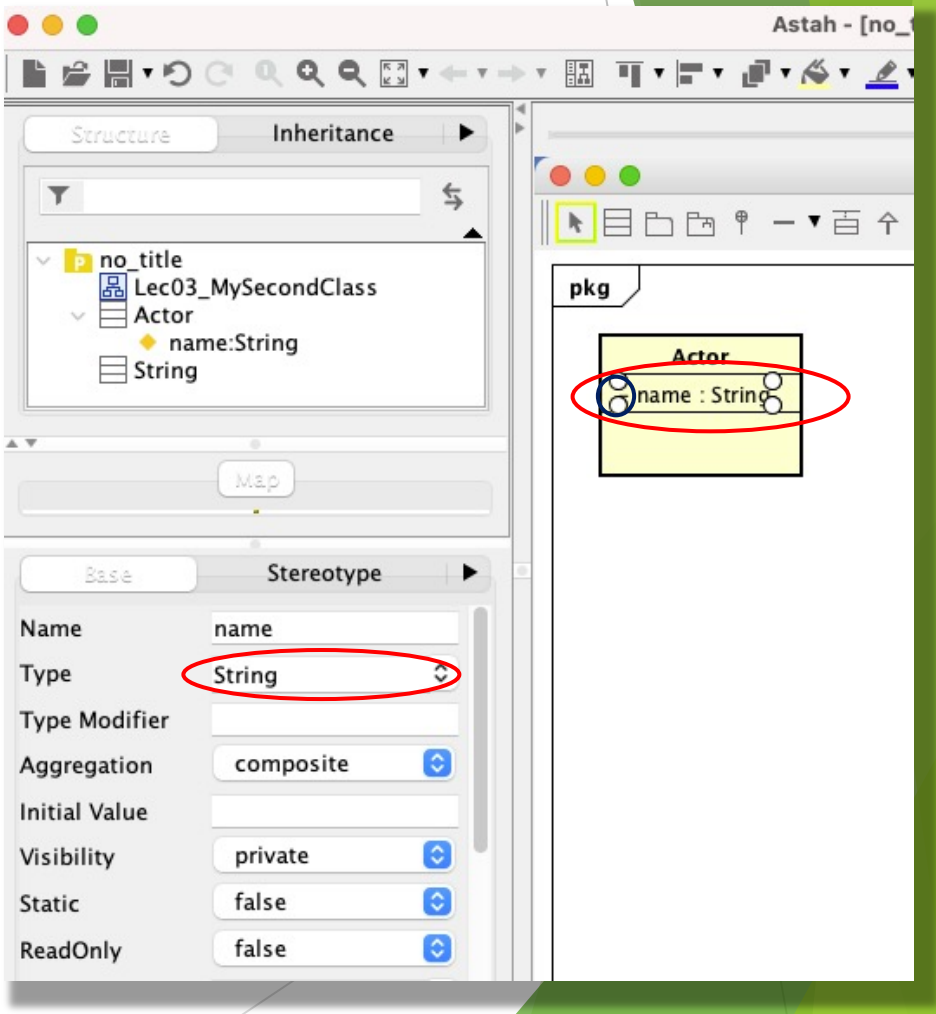
Step 3: Add Attributes

- Right click on “Actor” class box (the popup menu will appear), then select “Add Attribute”.
- Renaming the attribute will be done in the blue field, or in “Name” field.



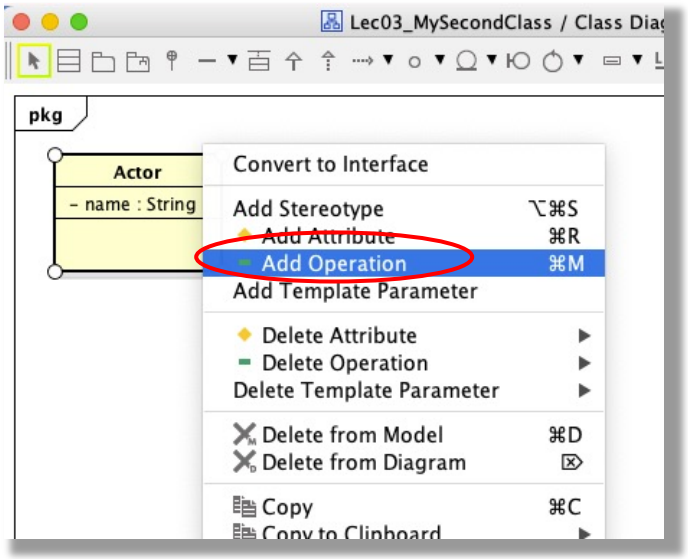
Step 4: Change Attribute Type

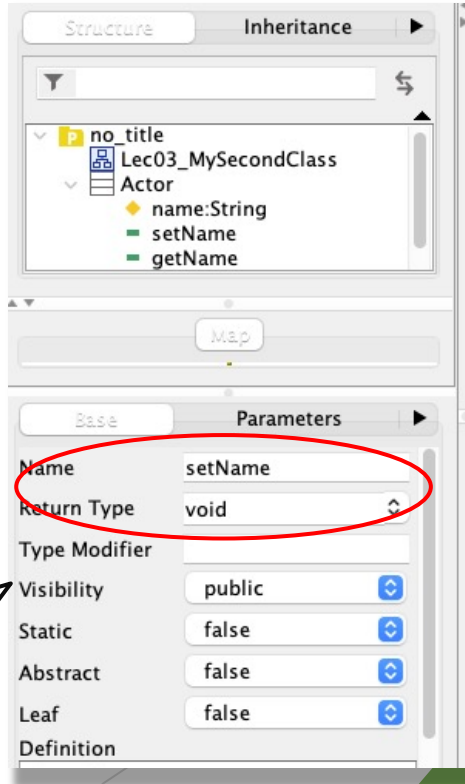
- Find the textbox by “Base -> Type”, then type “String”, finally the attribute is completed.
  - There is only one attribute, whose name is “name” and whose type is “String”
  - The character “-” means the attribute is specified as “private”.

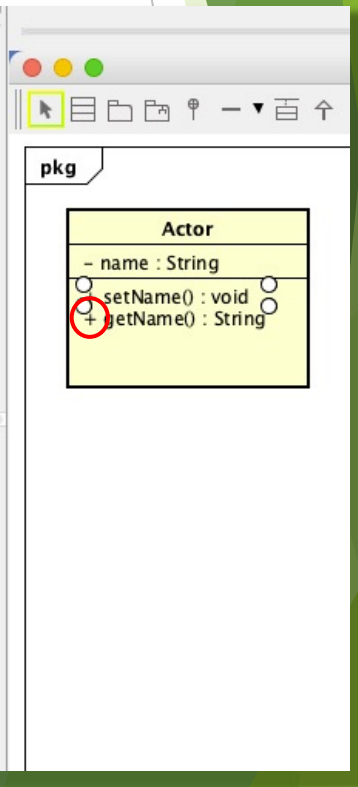


Step 5: Add Two Operations

- Right click on “Actor” class box (the popup menu will appear), then select “Add Operation”.
- Rename the operation “operation0” to “setName” and keep its returning type as “void”.
- Add one more operation “getName” with type “String”.



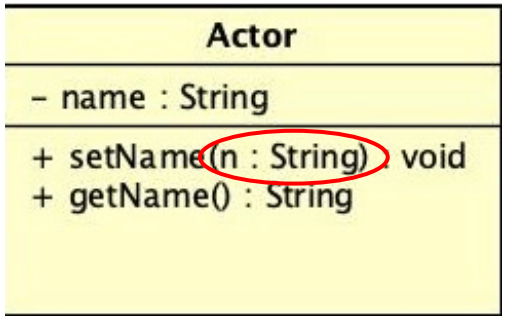




The character “+” means the operation is specified as “public”.

Final Class View

- Add “n: String” inside the round brackets of “setName”, then our first class will be completed.
- After designing, we can start coding.




```
// An actor with “name ” attribute
Public class Actor{
        // Fields
        private String name;
        // Create a new actor with the given name
        public Actor(){
                name = “<None>”;
        }
        // Get the name
        public String getName () {
                return name;
        }
        // set the name
        public void setName (String n){
                name = n;
        }
}
```

Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

Class Vocabularies

Class, Composition, Generalization, Instance and instantiation, Method, Message, Member, Constructor

Summary

- We considered the concept of class in OOP and UML design
- We studied class hierarchy and new UML connector – generalization
- Designing for a simple class (with Astah) was done
- Members of a class were studied

Outline

- The Definition of a Class
- Relationships Between Classes
- Depicting a Class
- Something More to Learn About a Class
- An Object, A Class
- Design and Code Our First Class
- Summary and Class Vocabularies
- Exercise 03

Exercise 03

- Deadline: **2022/04/28 (Mon.) 9:00**
- Please submit your answer file to “Exercise 03” under “Assignments” tab in Manaba +R
- Please put all of the answers in one “.pdf” file. The file name will be “**UML_Ex03_Your name.pdf**”
- The maximum points for “Exercise 03” will be **11p**
- If you put a wrong file name or wrong file format, your assignment will not be evaluated. Please be careful!

Tasks

- Task #01: In a UML diagram, how are objects distinguished from classes ? Choose only one option (1p)
  1) Object labels are shown in italics.
  2) Class labels are underlined.
  3) Object labels are underlined.
- Task #02: Consider the following diagrams (Diagram 1, Diagram 2) and answer the following question: "What do Diagrams 1 and 2 illustrate?" Choose only one option. (1p)
  1) Diagram 1: An aggregation, Diagram 2: Class with attribute
  2) Diagram 1: Class with an attribute, Diagram 2: An aggregation
  3) Diagram 1: Class with an attribute, Diagram 2: An association

Diagram 1

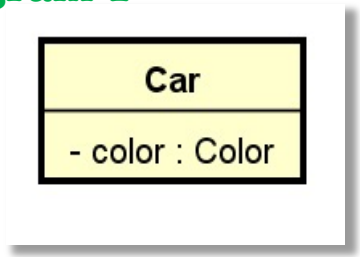
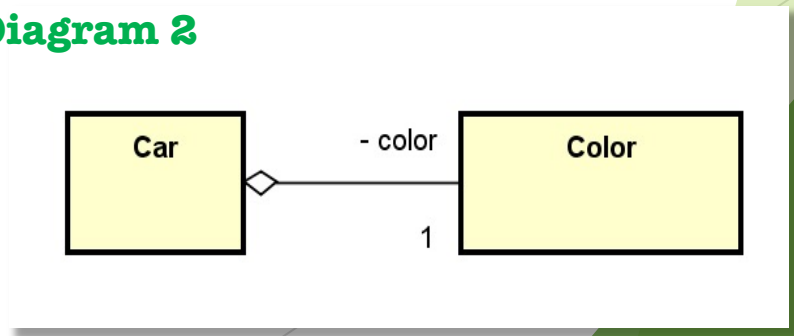
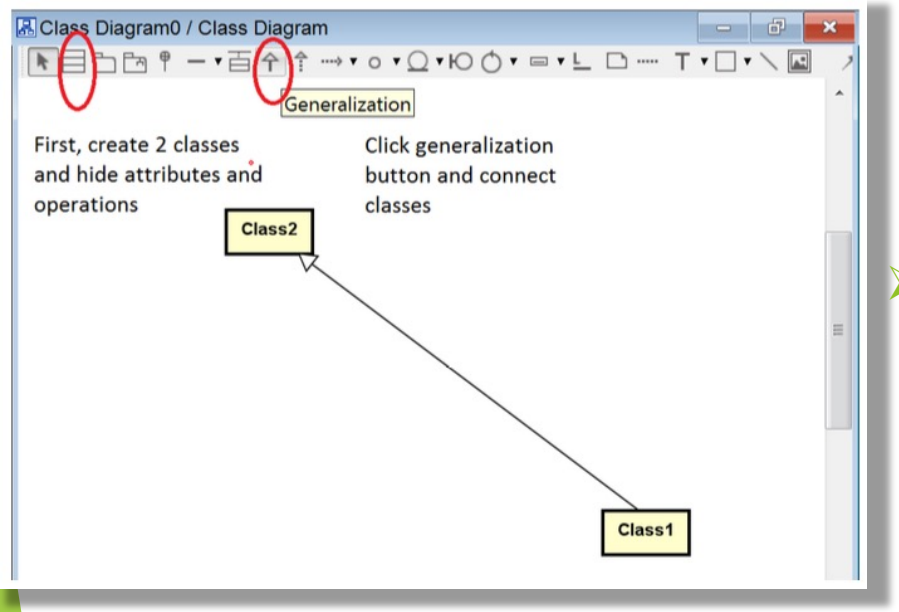


Diagram 2



Tasks

- Task #03: Build a simple class diagram (with Astah UML) based on the following instructions/image, then export your diagram as a PNG/JEPG image and insert your diagram image in your report (1p)



- Requirements:
  - ✓ 2 class blocks (1p)
  - ✓ Connection type: "Generalization" (1p)
  - ✓ Add 1 or 2 attributes into each class block (2p)
  - ✓ Add 1 or 2 operations into each class block (2p)
  - ✓ Change the color of one of the class blocks (1p)
  - ✓ Add your name into one of the class blocks as the class name (1p)