Brunel University London

Software Development and Management
CS2002

Dr Giuseppe Destefanis
giuseppe.destefanis@brunel.ac.uk
@GiuseppeDes

Lecture 4

UML Class diagrams

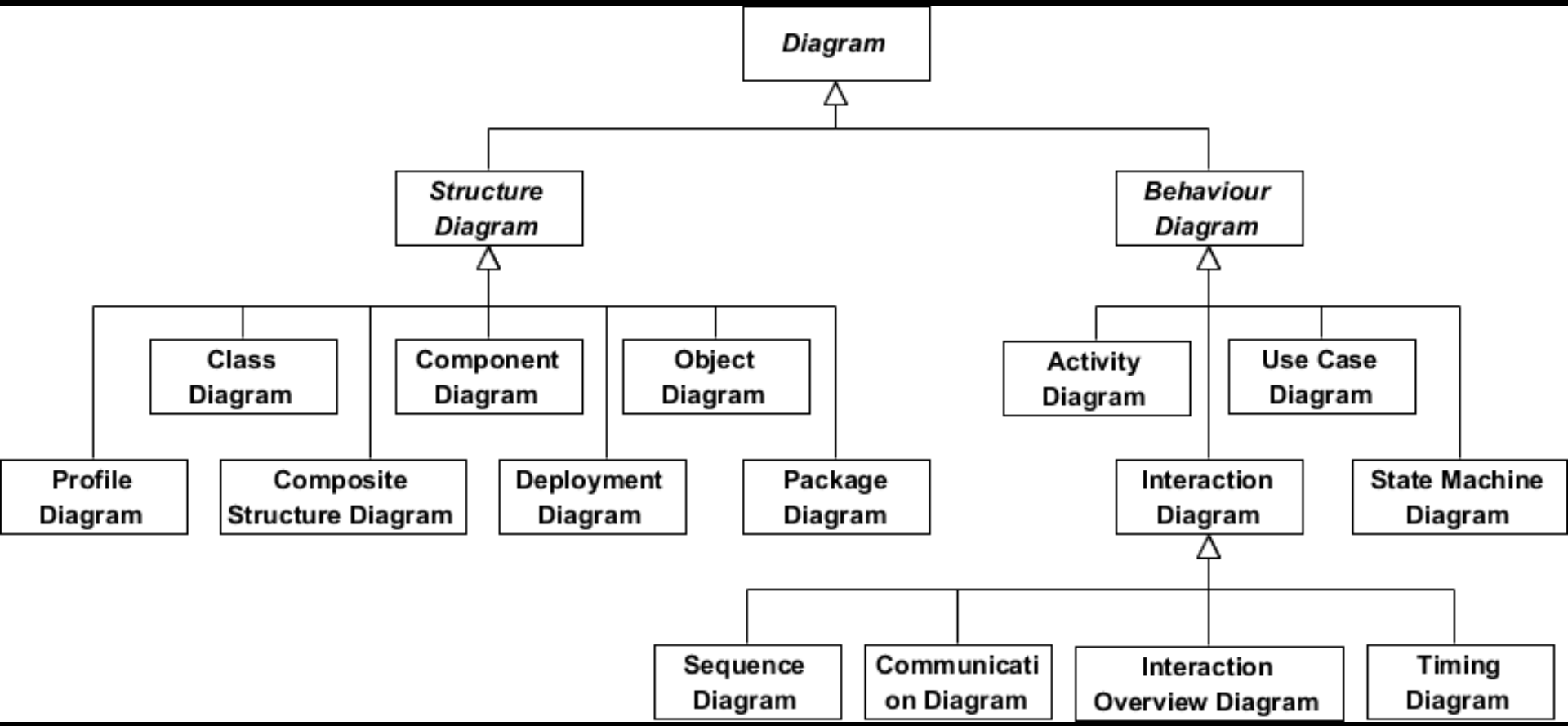
At the end of this lecture, and after a period of independent study:

- You should be able to define what a Class Diagram is
- You should be able to identify classes given a problem statement

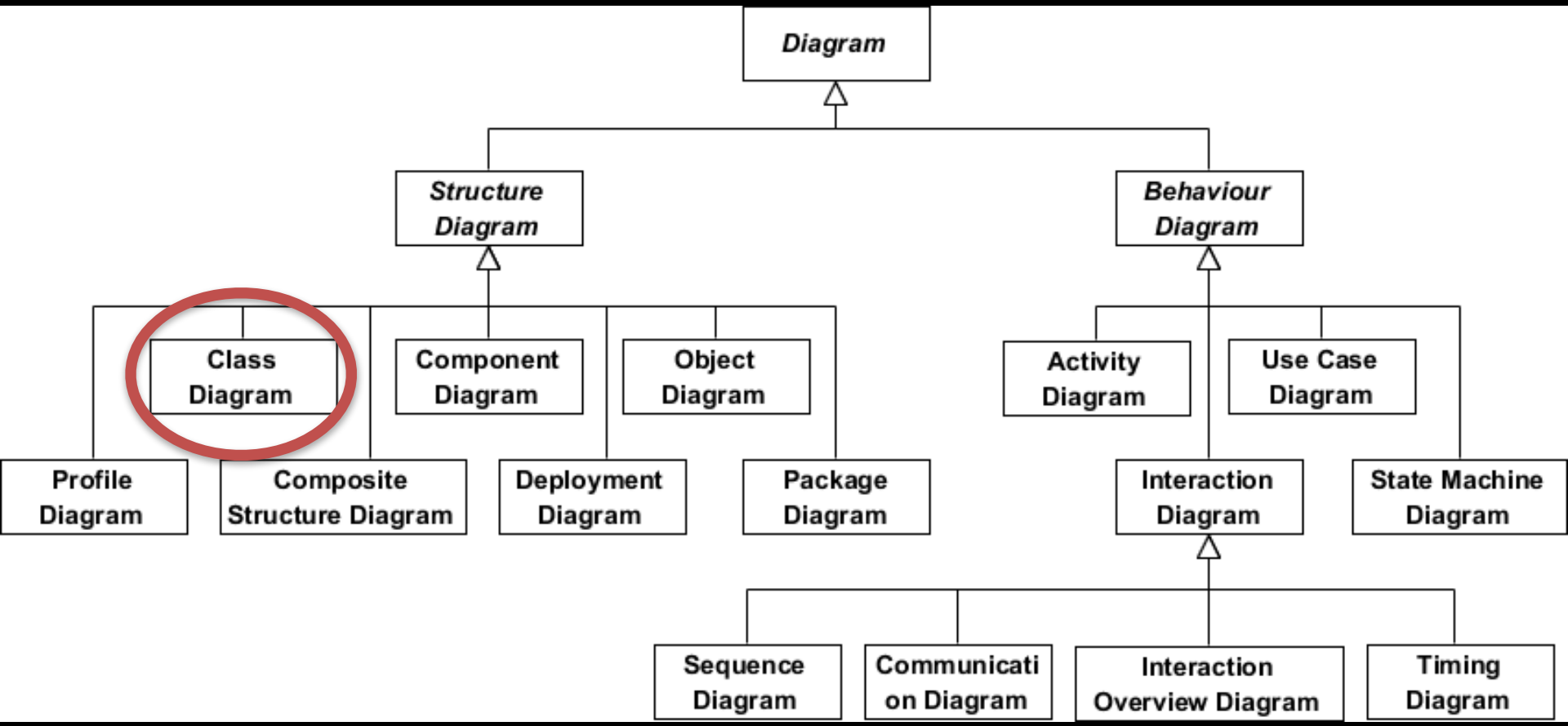
What is a Class Diagram?

in the Unified Modeling Language, is a type of static structure diagram that describes the structure of a system by showing **the *system*'s *classes, their attributes, operations (or methods), and the relationships among objects***

Different diagrams for different viewpoints



Different diagrams for different viewpoints



Purpose of a Class Diagram

- Shows static structure of classifiers in a system
- Diagram provides a basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members
- Business Analysts can use class diagrams to model systems from a business perspective

A UML Class Diagram contains:

- A set of classes
- A set of relationships between classes

A class:

A description of a group of objects all with similar roles in the system

Structural features (attributes)

- define what objects of the class "know"
- Represent the state of an object of the class
- Are descriptions of the structural or static features of a class

Behavioural features (operations)

- define what objects of the class "can do"
- Define the way in which objects may interact
- Operations are descriptions of behavioral or dynamic features of a class

A class notation consists of three parts:

Class Name

- The name of the class appears in the first partition

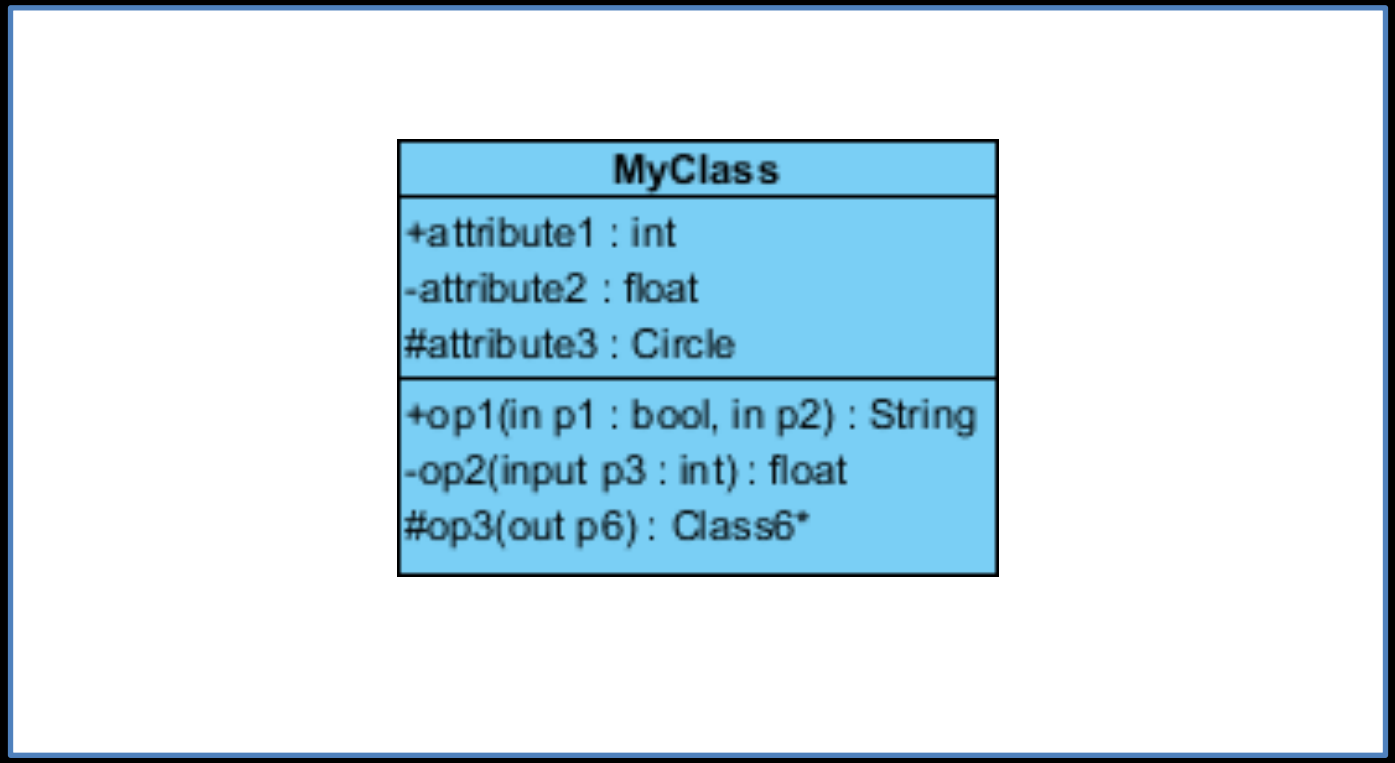
Class Attributes

- Attributes are shown in the second partition
- The attribute type is shown after the colon
- Attributes map onto member variables (data members) in code

Class Operations (Methods)

- Operations are shown in the third partition. They are services the class provides
- The return type of a method is shown after the colon at the end of the method signature
- The return type of method parameters is shown after the colon following the parameter name

UML Class representation

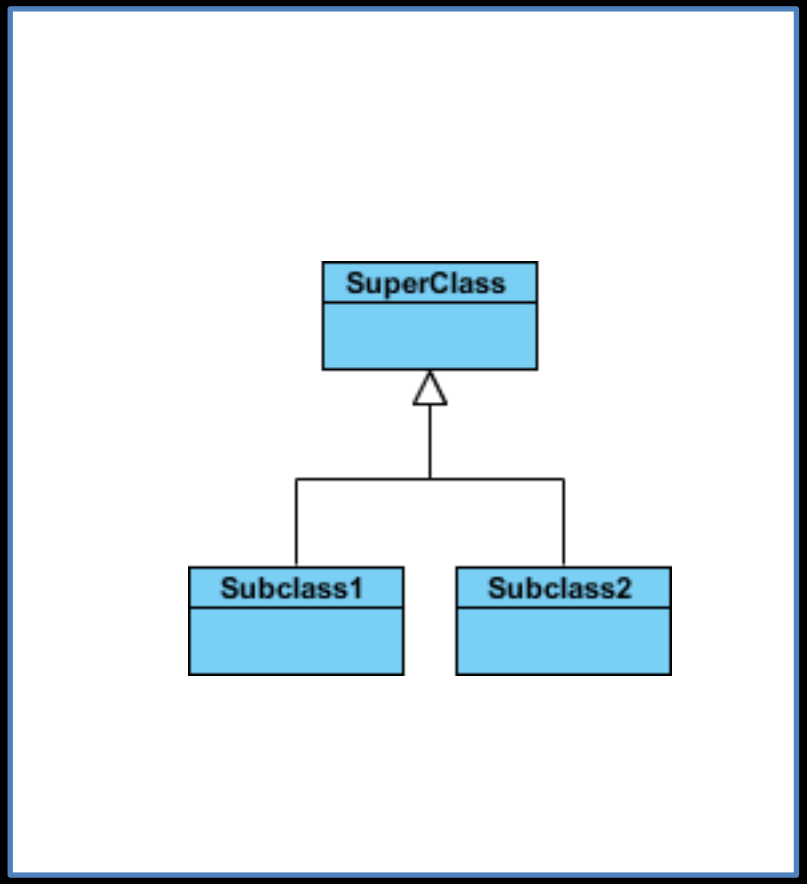


MyClass

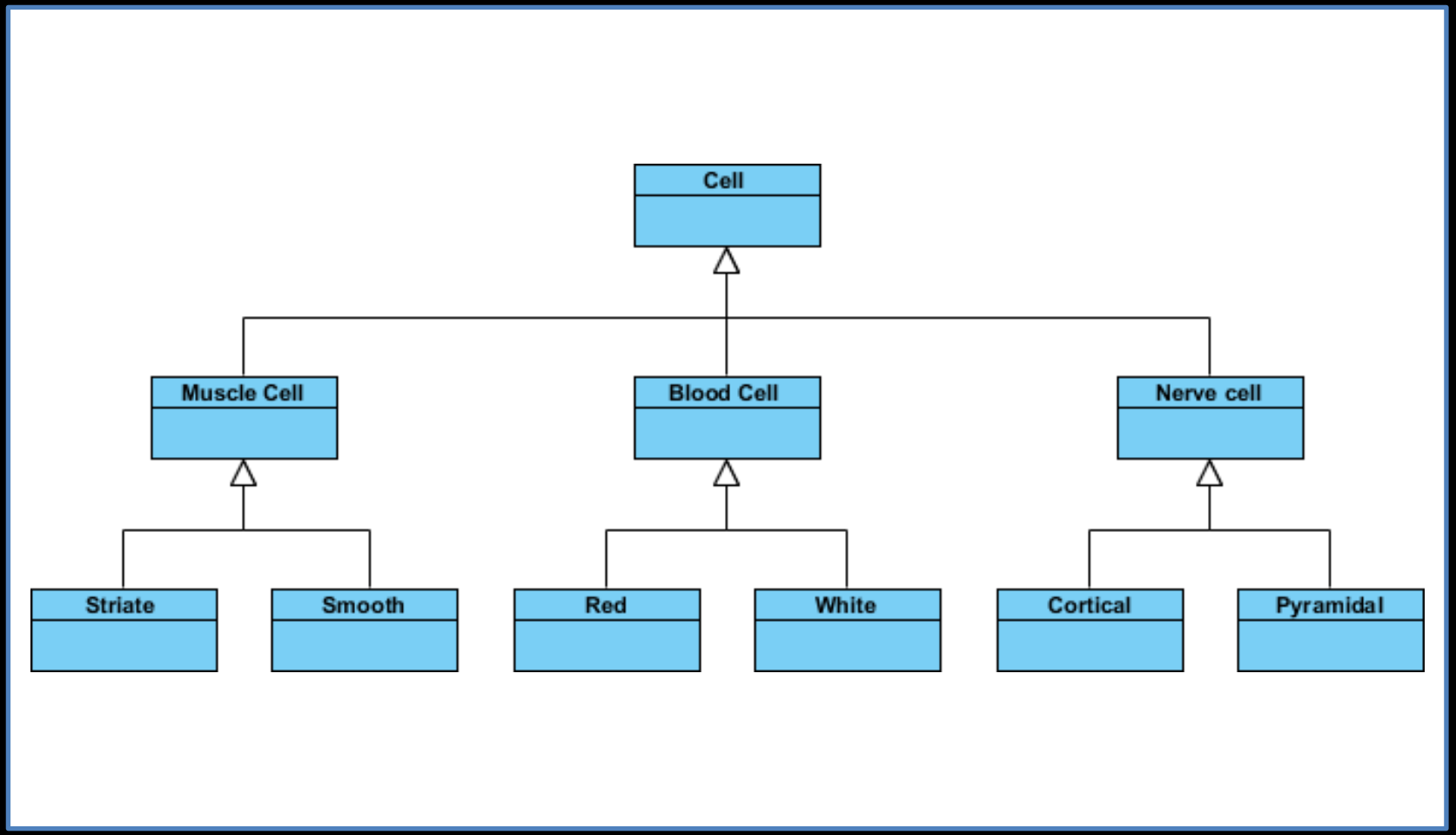
- MyClass has 3 attributes and 3 operations
- Parameter p3 of op2 is of type int
- op2 returns a float
- op3 returns a pointer (denoted by a *) to Class6

Inheritance

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class

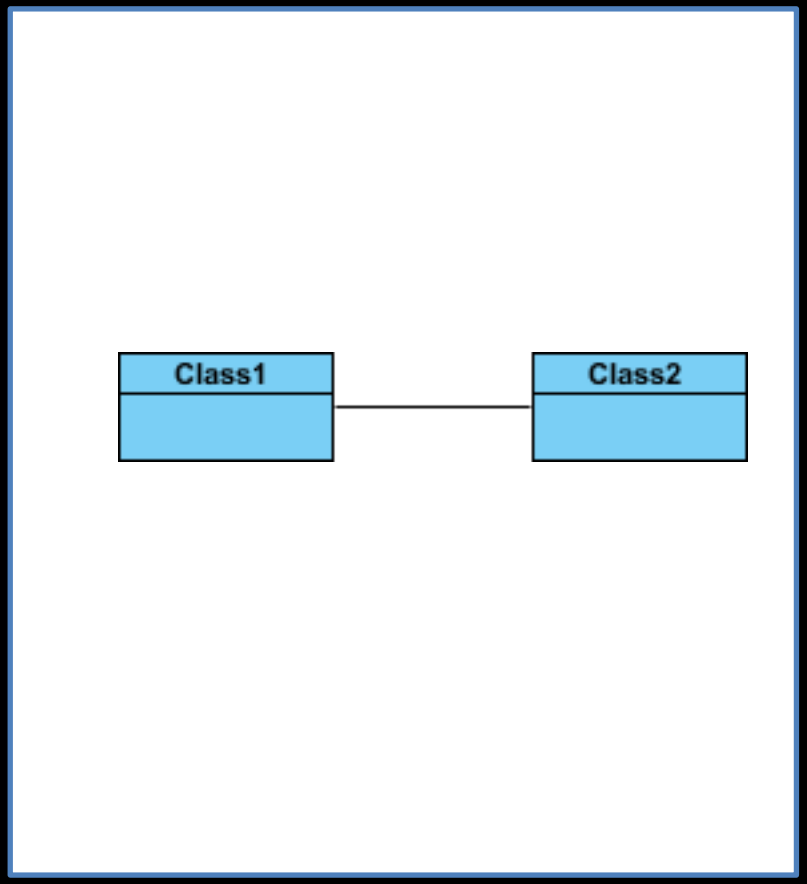


Inheritance



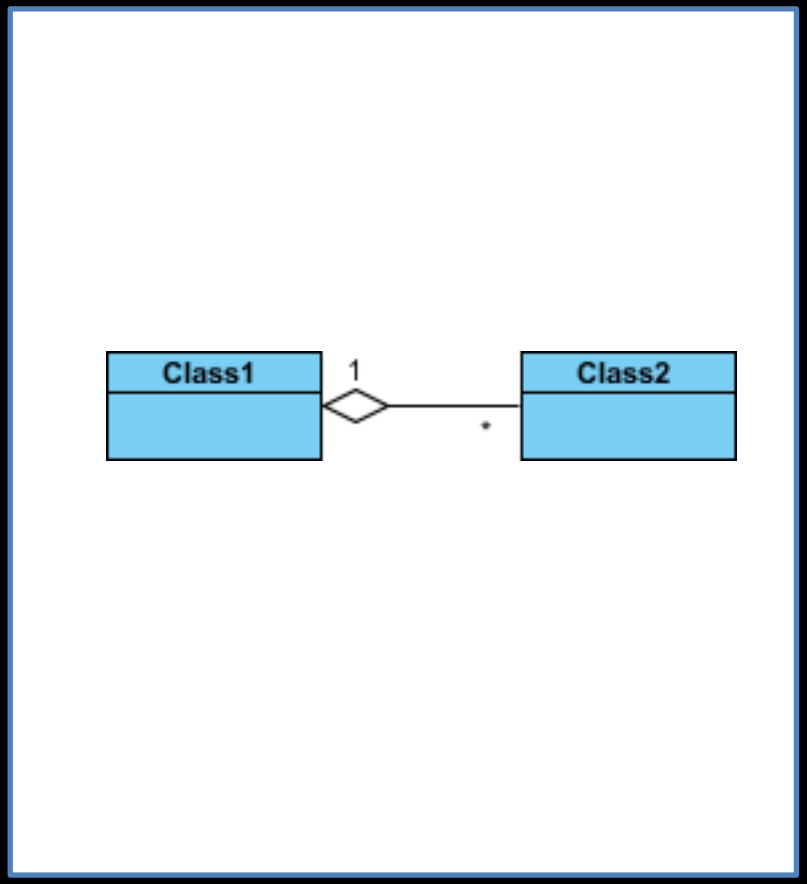
Simple Association

- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes

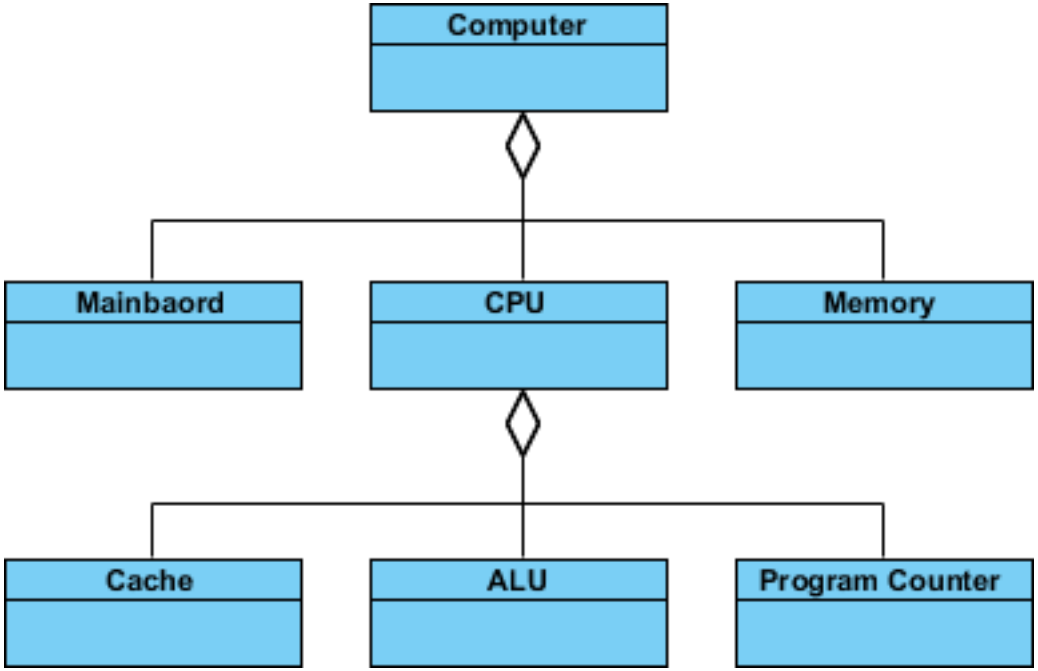


Aggregation

- A special type of association. It represents a "part of" relationship.
- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite

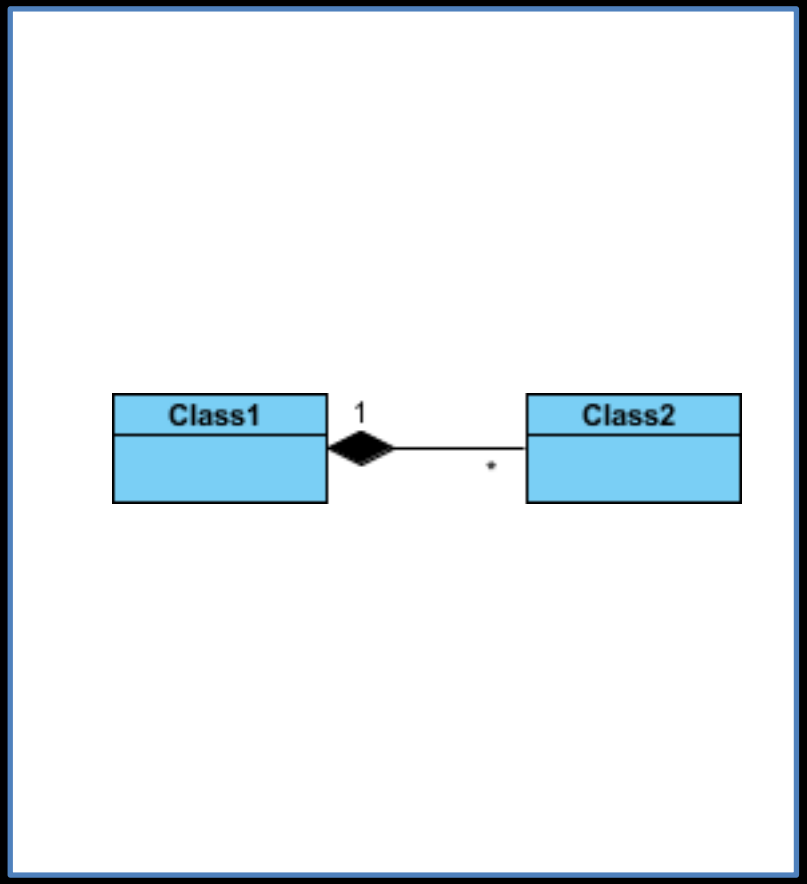


Aggregation



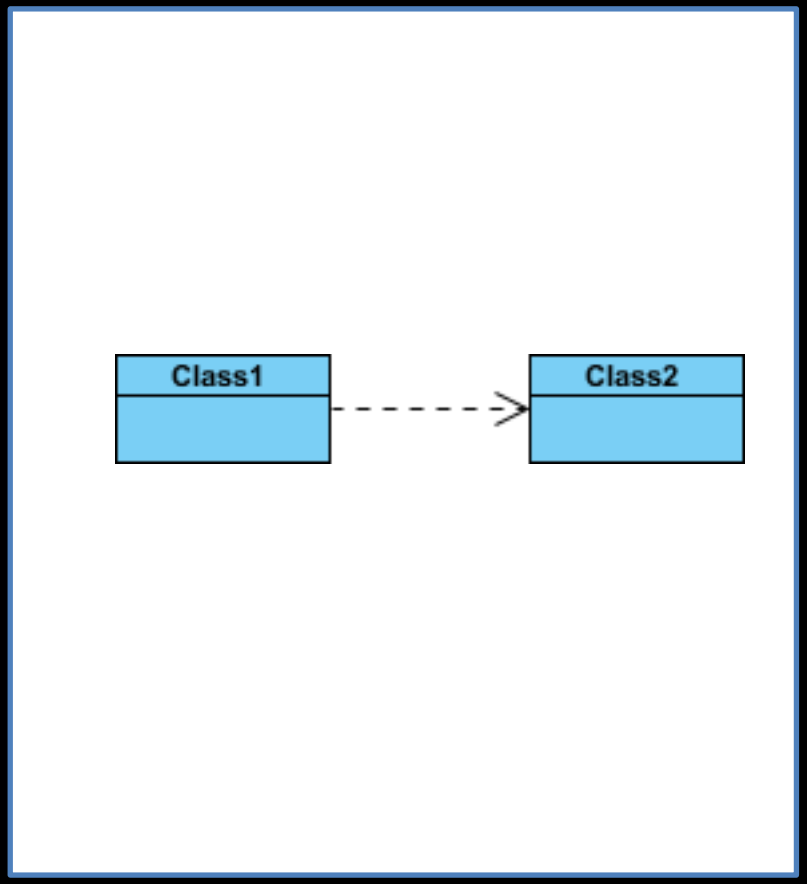
Composition

- A special type of aggregation where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite

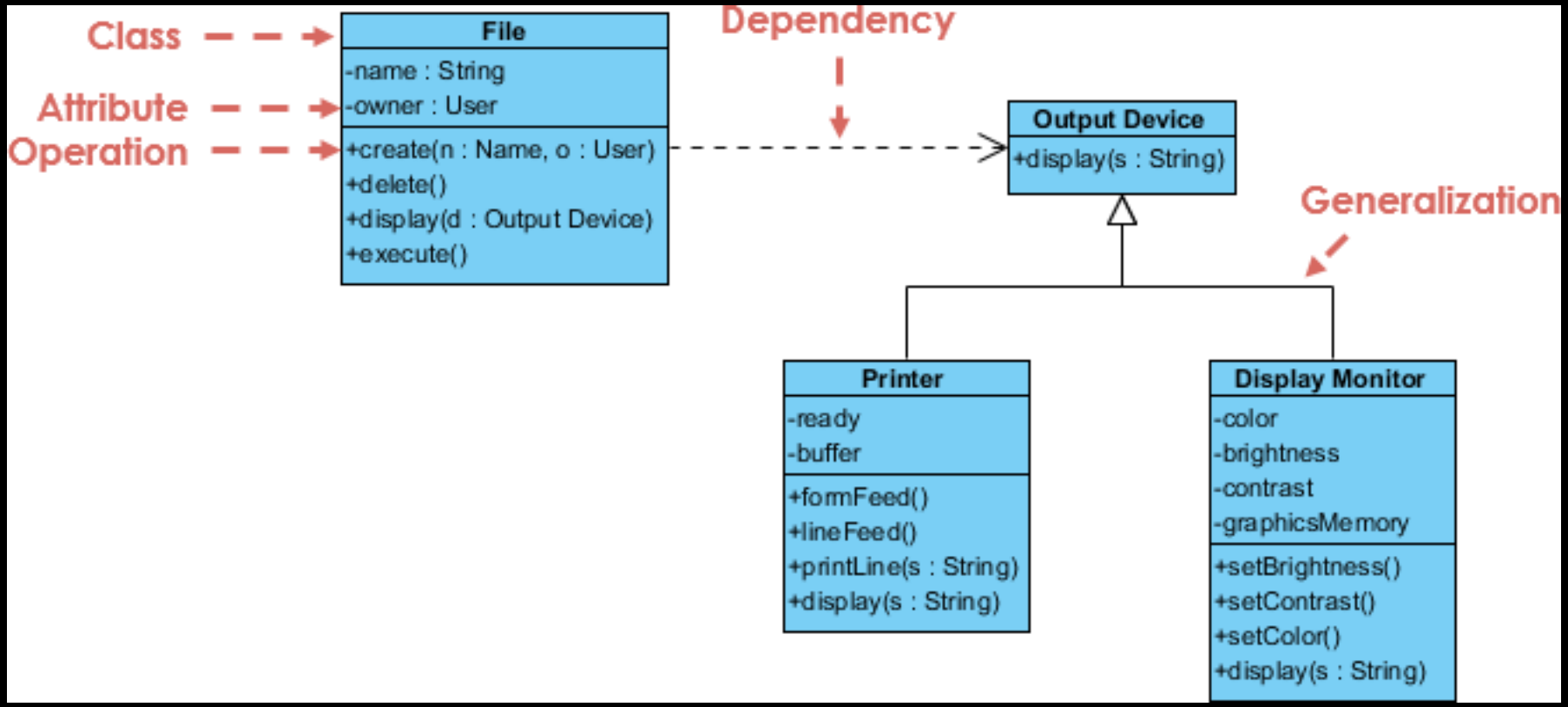


Dependency

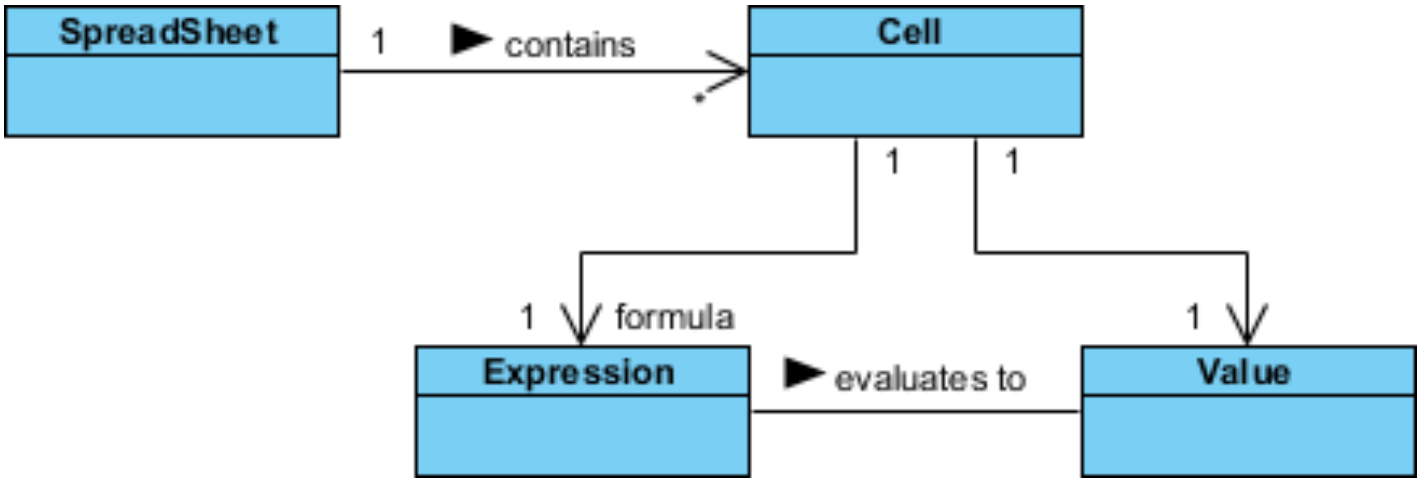
- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2
- A dashed line with an open arrow



Dependency


Class
Attribute
Operation
File
-name : String
-owner : User
+create(n : Name, o : User)
+delete()
+display(d : Output Device)
+execute()
Dependency
Output Device
+display(s : String)
Generalization
Printer
-ready
-buffer
+formFeed()
+lineFeed()
+printLine(s : String)
+display(s : String)
Display Monitor
-color
-brightness
-contrast
-graphicsMemory
+setBrightness()
+setContrast()
+setColor()
+display(s : String)

Relationship Names



Visibility of Class attributes and Operations

- In object-oriented design, there is a notation of visibility for attributes and operations. UML identifies four types of visibility: **public, protected, private, and package.**

- The +, -, # and ~ symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

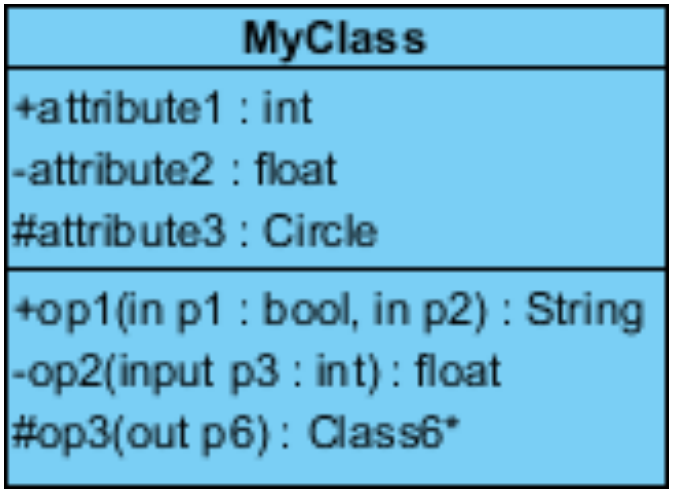
- + denotes public attributes or operations

- - denotes private attributes or operations

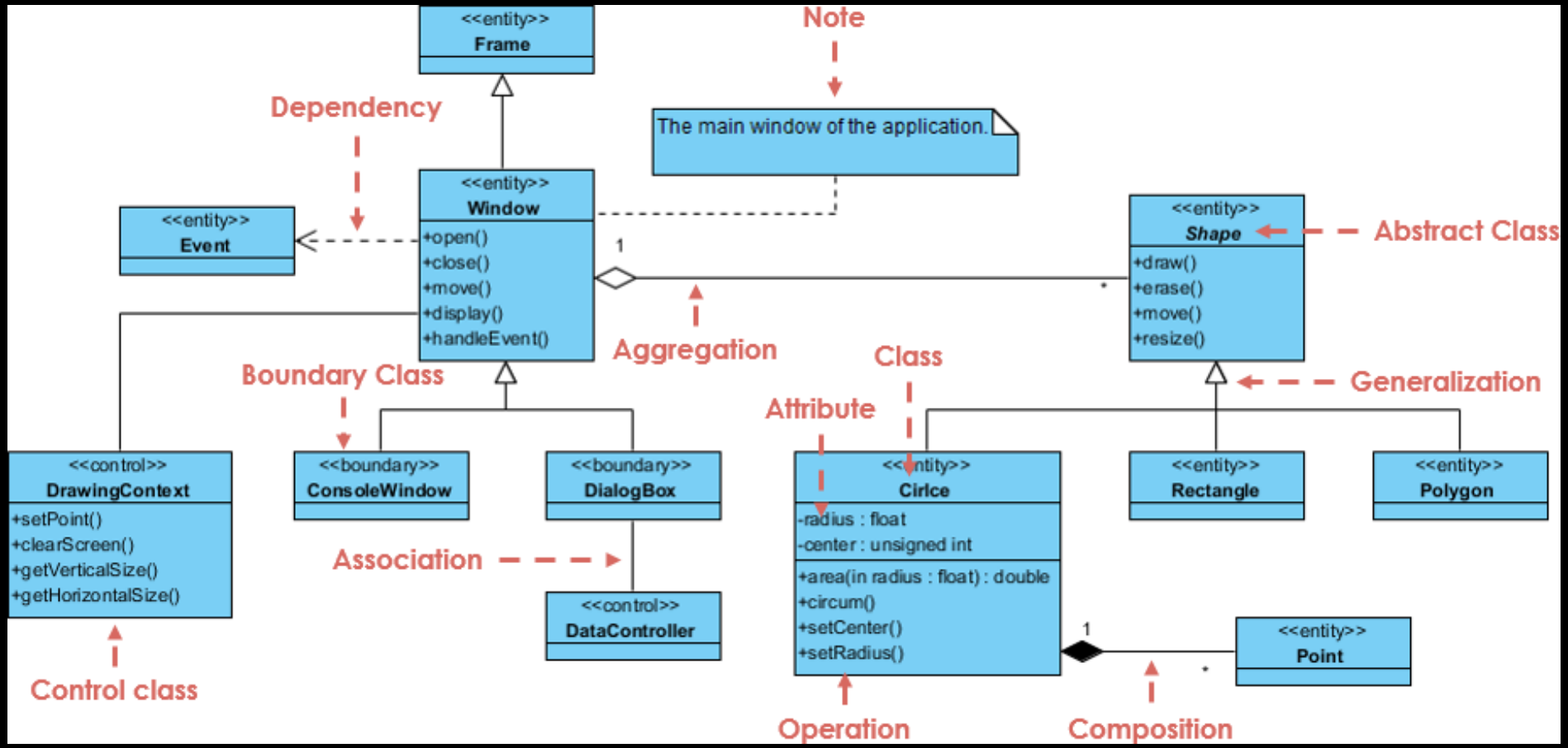
- # denotes protected attributes or operations

- ~ denotes package attributes or operations

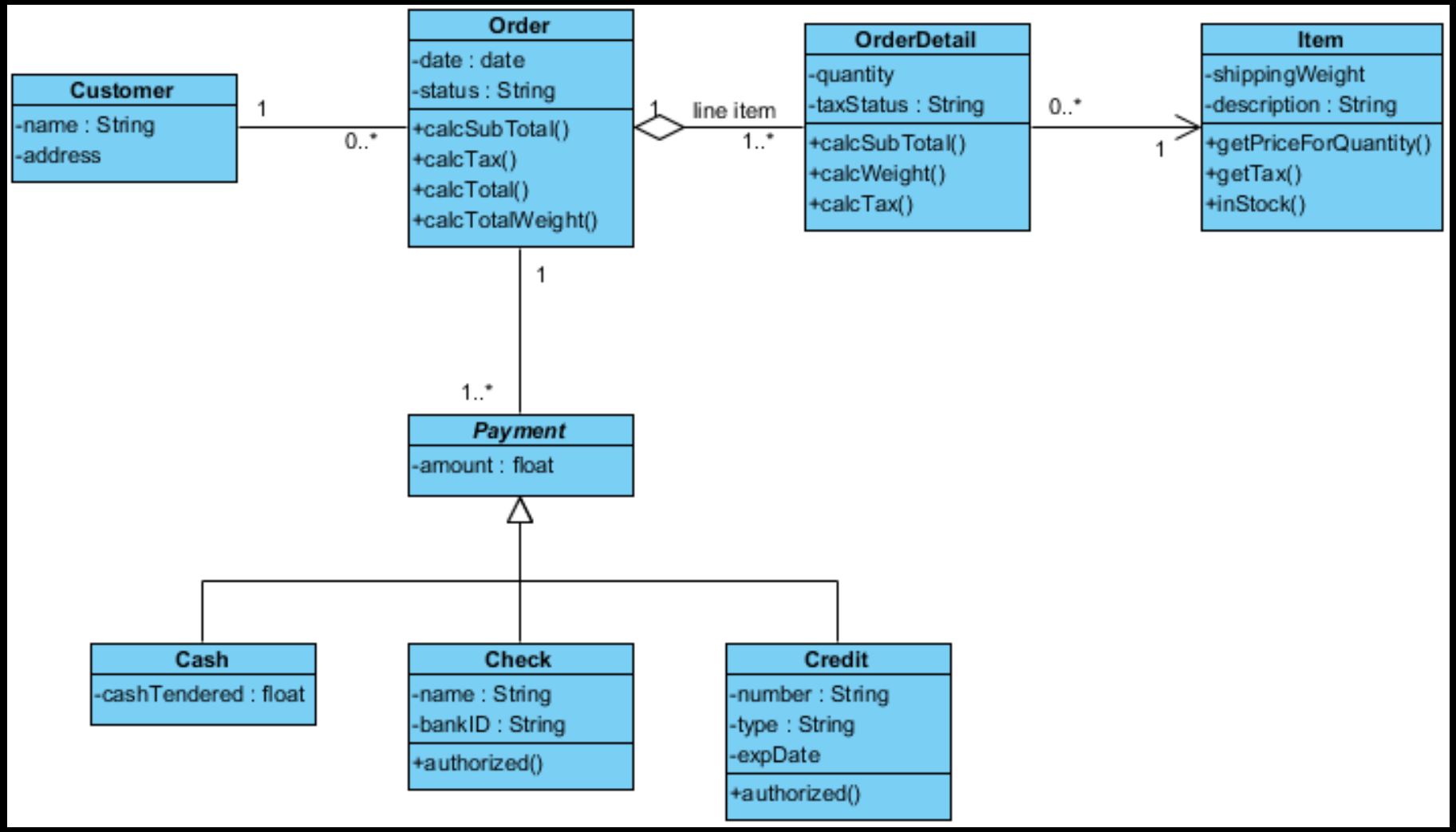
Visibility


MyClass
+attribute1 : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : bool, in p2) : String
-op2(input p3 : int) : float
#op3(out p6) : Class6*

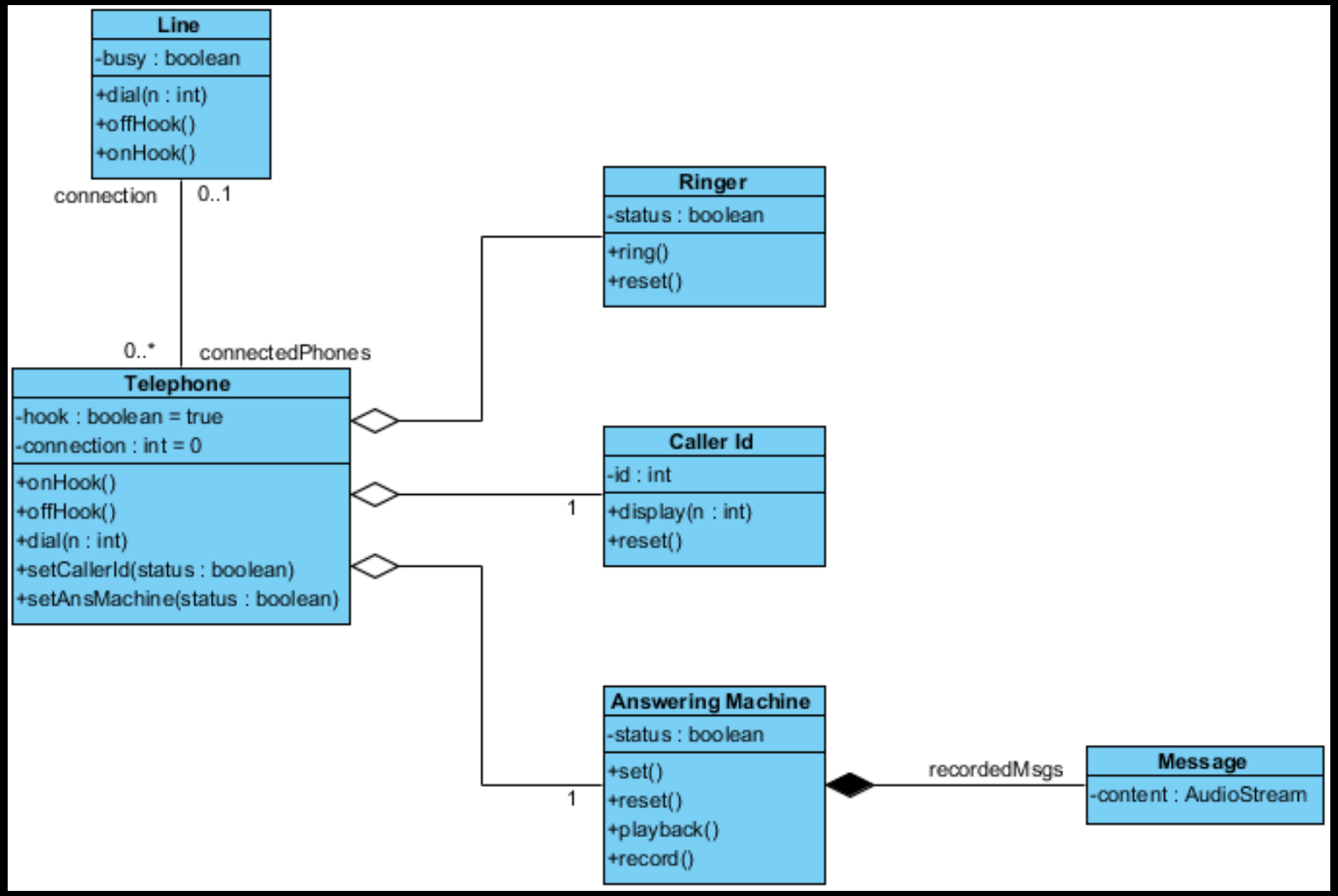
Class Diagram (example)



Class Diagram (II)



Class Diagram (III)



Reading

- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- https://circle.visual-paradigm.com/gallery/general/class-diagram/