Class Diagrams
System Analysis and Design

Adiba Shaira
Lecturer, Department of CSE,
United International University

Courtesy: Farhanaz Farheen

What are Class diagrams?

Class diagrams are diagrams that represent a system structure by denoting the classes, attributes, operations and the relationships between objects.

Format

- A class shape contains three rows separated by horizontal lines
- The top row will contain the name of the class under consideration
- The middle row contains all the attributes of the class
- And the bottom row contains all the methods

Access Specifiers

We can specify the level of access of each attribute and method in a class with symbols:

- Public (given by +)
- Private (given by -)
- Protected (given by #)
- Default/Package Access (given by ~)

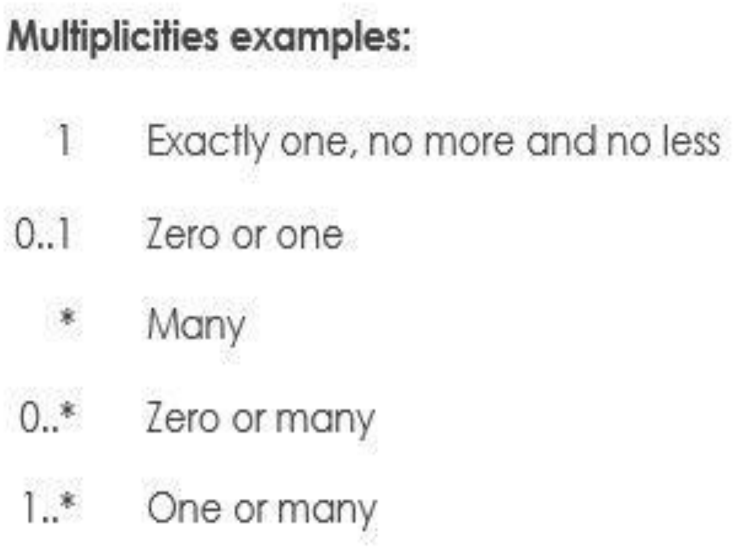
Components of class representation

- Top row: Contains the name of the class. This is compulsory: No matter what the internal contents of the class are, the name of the class should be written and this field cannot be empty.
- Middle row: Contains the attributes of a class. Can be empty.
- Last row: Contains the operations or methods of a class. Can be empty.

Relationships between classes

- Inheritance
- Association (Unidirectional or Bidirectional)
- Realization
- Dependency

Multiplicity



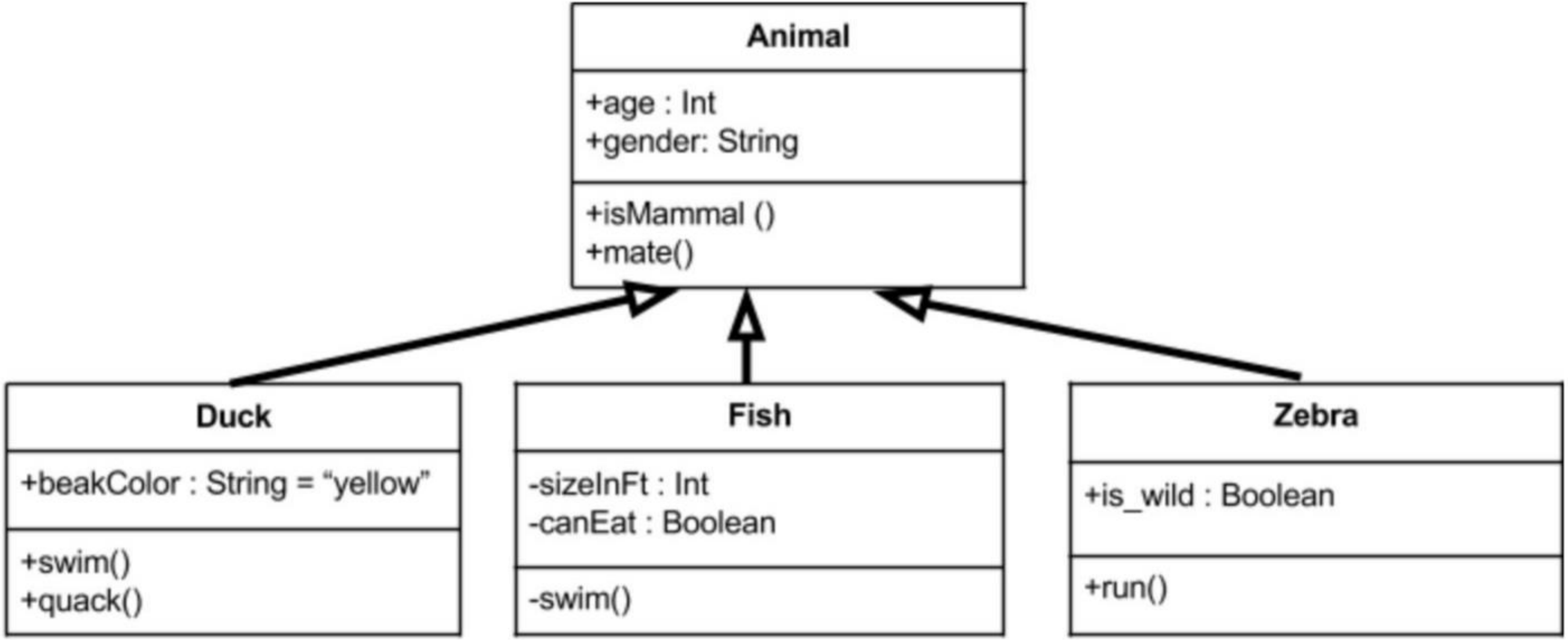
Multiplicities examples:

1	Exactly one, no more and no less
0..1	Zero or one
*	Many
0..*	Zero or many
1..*	One or many

Inheritance or Generalization

A child or subclass inherits a parent or superclass.

The rules are similar to those of Object Oriented Programming concepts.



Animal

+age : Int
+gender: String

+isMammal ()
+mate()

Duck

+beakColor : String = “yellow”

+swim()
+quack()

Fish

-sizeInFt : Int
-canEat : Boolean

-swim()

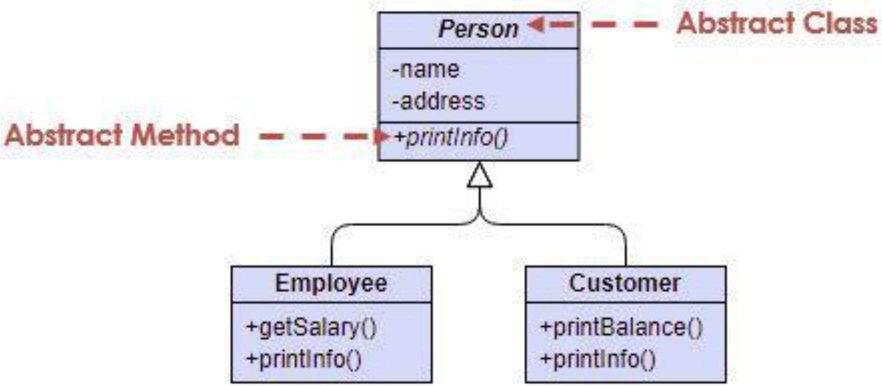
Zebra

+is_wild : Boolean

+run()

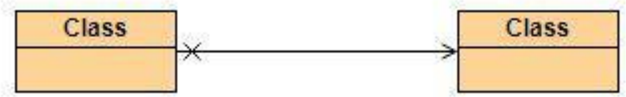
Abstract classes

Written in Italic font



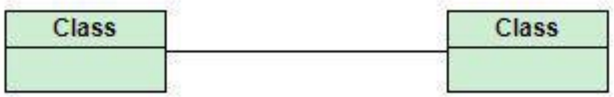
Unidirectional Association

Less common relationship. A class is aware of the other and interacts with it. Two classes are related but only one is aware that the relationship exists.



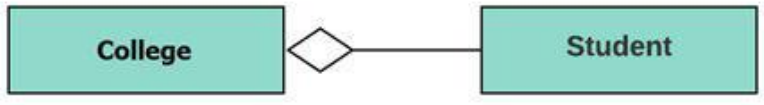
Bidirectional Association

Both classes are aware of the relationship between them. Associations are commonly always assumed to be bi-directional.



Aggregation

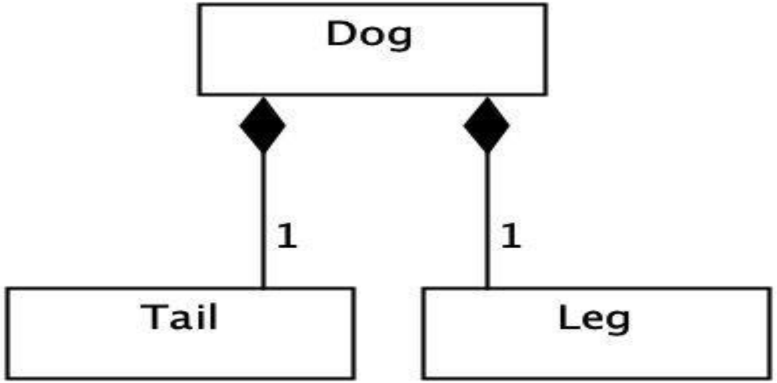
It models a whole-part relationship between an aggregate and its parts.



College is made up of one or more student. Students will still exist even if College is removed.

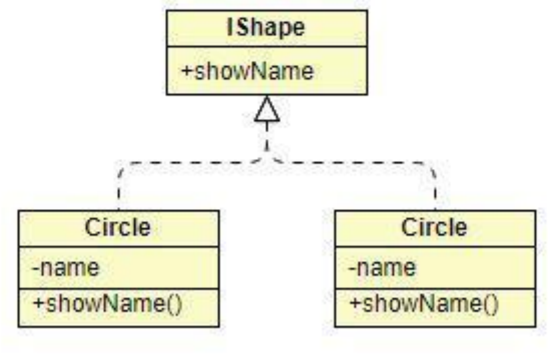
Composition

Special type of association. It denotes strong ownership between two classes.



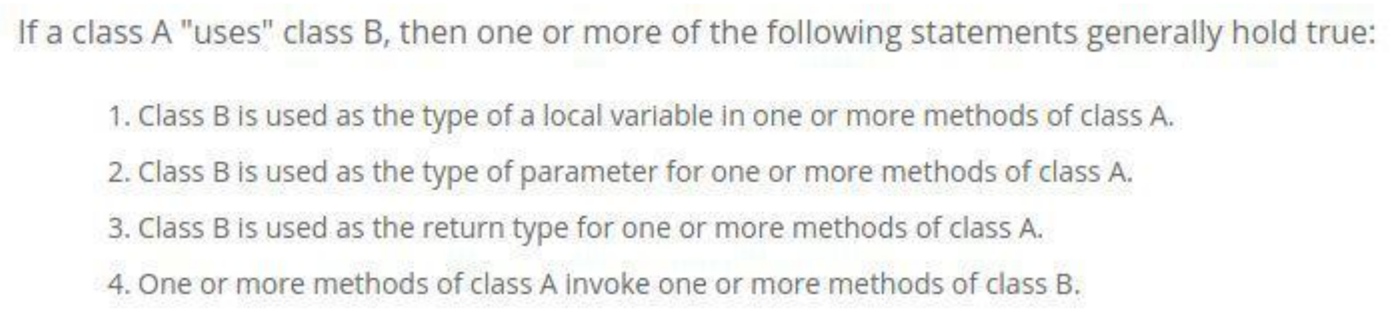
Realization

Used to represent Interfaces.



Dependency

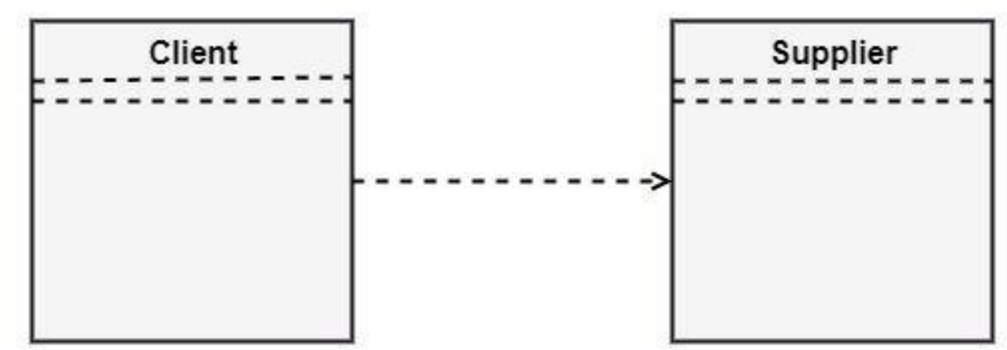
Indicates a “uses” relationship between two classes.

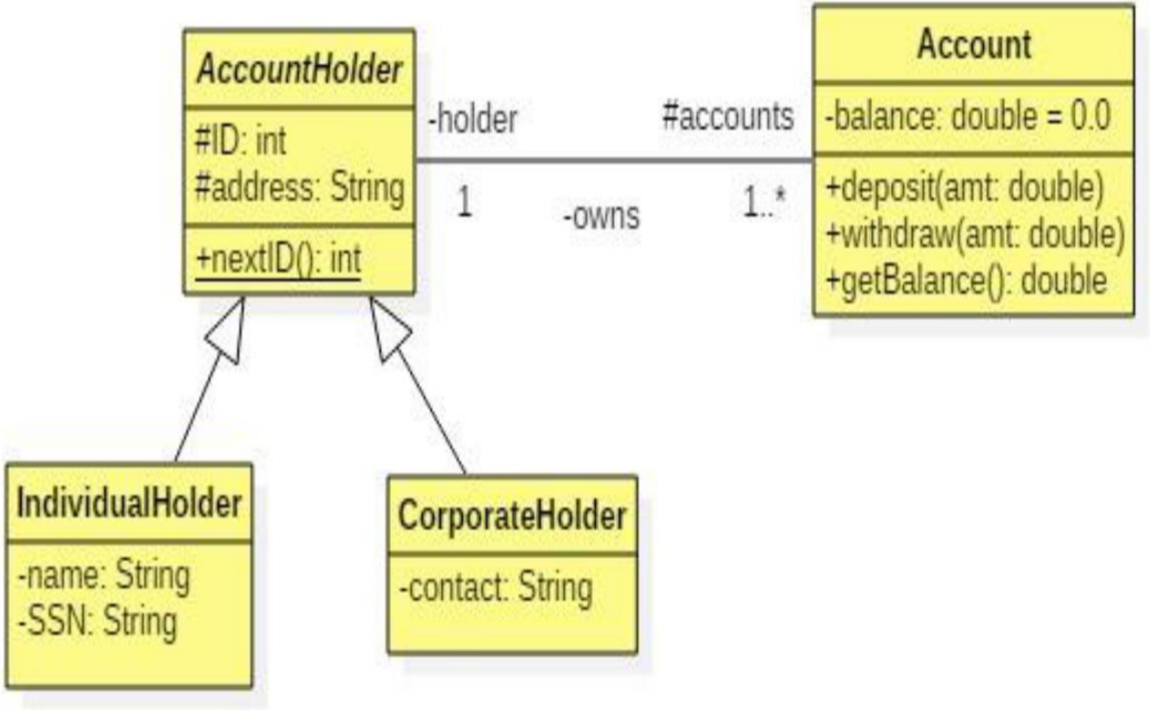


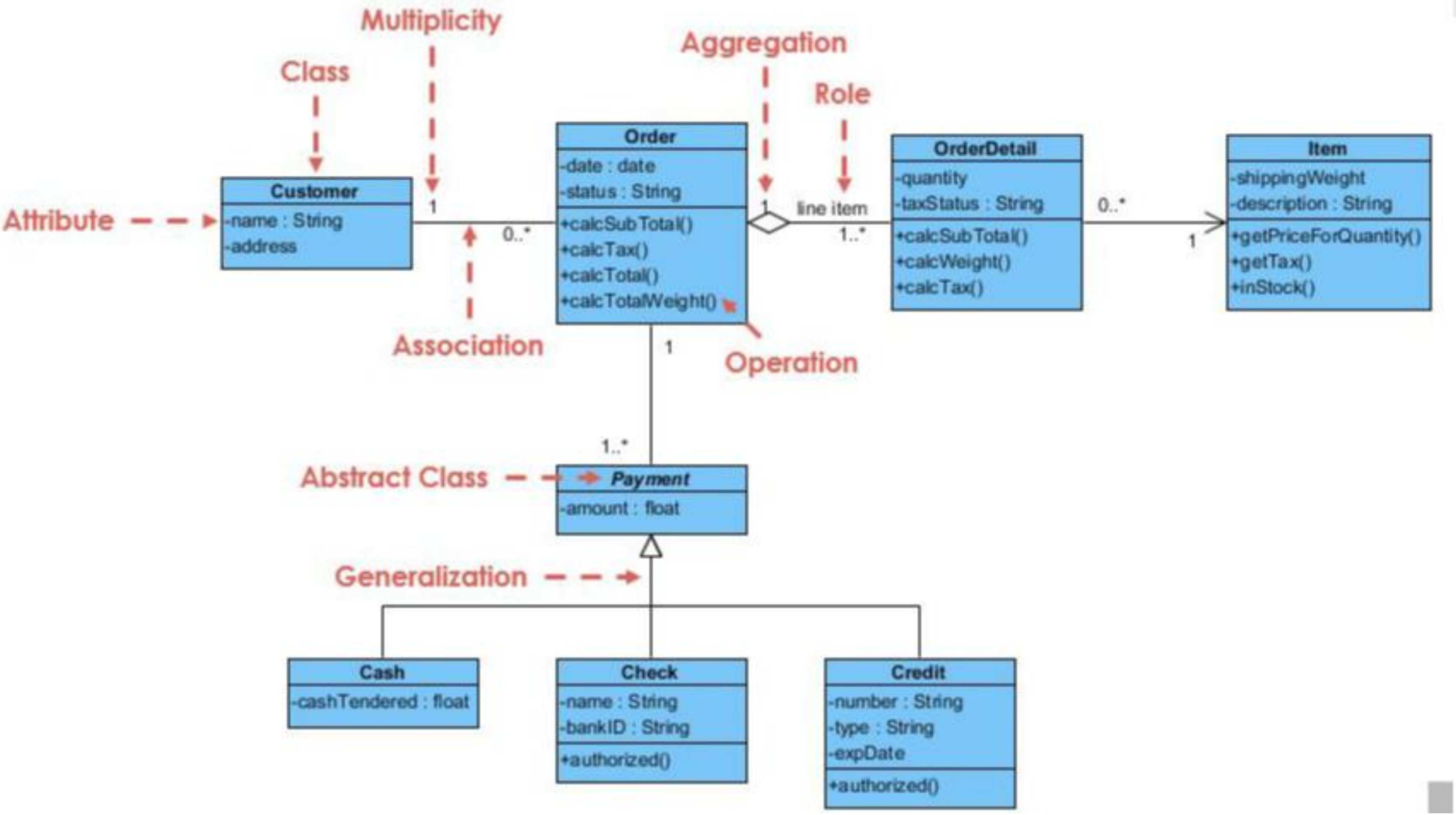
If a class A "uses" class B, then one or more of the following statements generally hold true:

1. Class B is used as the type of a local variable in one or more methods of class A.
2. Class B is used as the type of parameter for one or more methods of class A.
3. Class B is used as the return type for one or more methods of class A.
4. One or more methods of class A invoke one or more methods of class B.

Dependency




AccountHolder
#ID: int
#address: String
+nextID(): int
-holder
1
-owns
#accounts
1..*
Account
-balance: double = 0.0
+deposit(amt: double)
+withdraw(amt: double)
+getBalance(): double
IndividualHolder
-name: String
-SSN: String
CorporateHolder
-contact: String


Multiplicity
Class
Aggregation
Role
Attribute
Association
Operation
Abstract Class
Generalization
Customer
-name : String
-address
1
0..*
Order
-date : date
-status : String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight()
1
line item
1..*
OrderDetail
-quantity
-taxStatus : String
+calcSubTotal()
+calcWeight()
+calcTax()
0..*
1
Item
-shippingWeight
-description : String
+getPriceForQuantity()
+getTax()
+inStock()
1
1..*
Payment
-amount : float
Cash
-cashTendered : float
Check
-name : String
-bankID : String
+authorized()
Credit
-number : String
-type : String
-expDate
+authorized()

Advantages

- Help to understand the general overview of application schematics
- They demonstrate data models for information systems
- Help to visually express the requirements or needs of a system
- They allow the visualization of the system without giving details of implementation

Resources

1. https://www.lucidchart.com/pages/uml-class-diagram
2. https://www.guru99.com/uml-class-diagram.html
3. https://online.visual-paradigm.com/diagrams/tutorials/class-diagram-t utorial/

Thank you!