Class diagram

Class diagram

- Class diagrams are the main building block in object-oriented modeling.
- They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

Class diagram

- In UML, a class represents an object or a set of objects that share a common structure and behaviour.
- They're represented by a rectangle that includes rows of the class name, its attributes, and its operations.

Upper section: Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.

Middle section: Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.

Bottom section: Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

Class diagram

Access specifier for an attribute and a method in a class diagram

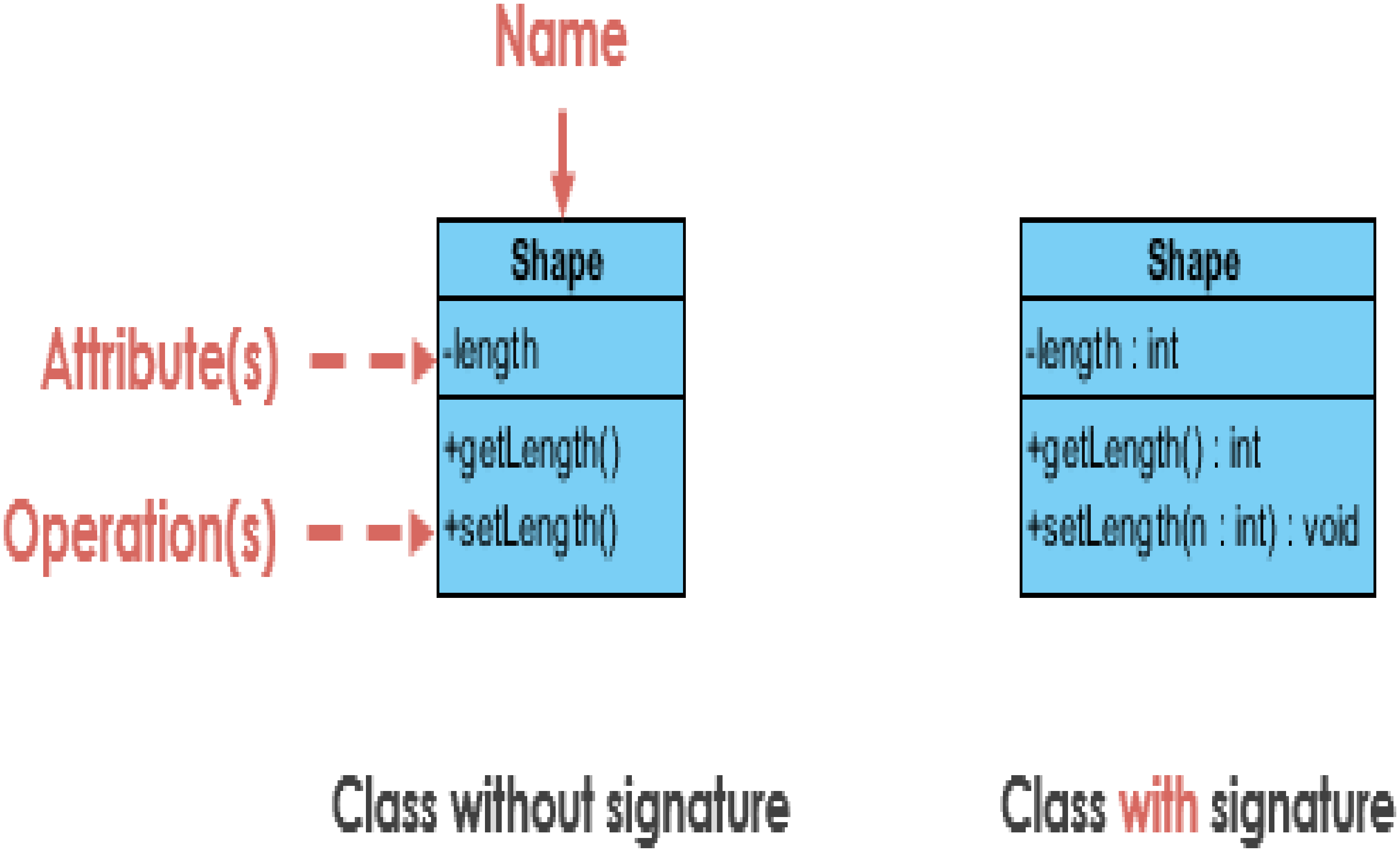
+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

~ denotes package attributes or operations

Class diagram



Class diagram

How to identify classes?

One of the approaches is:

Use noun form approach

Class diagram

- Relationships in UML are used to represent a connection between structural, behavioral, or grouping things.
- It is also called a link that describes how two or more things can relate to each other during the execution of a system. Type of UML Relationship are

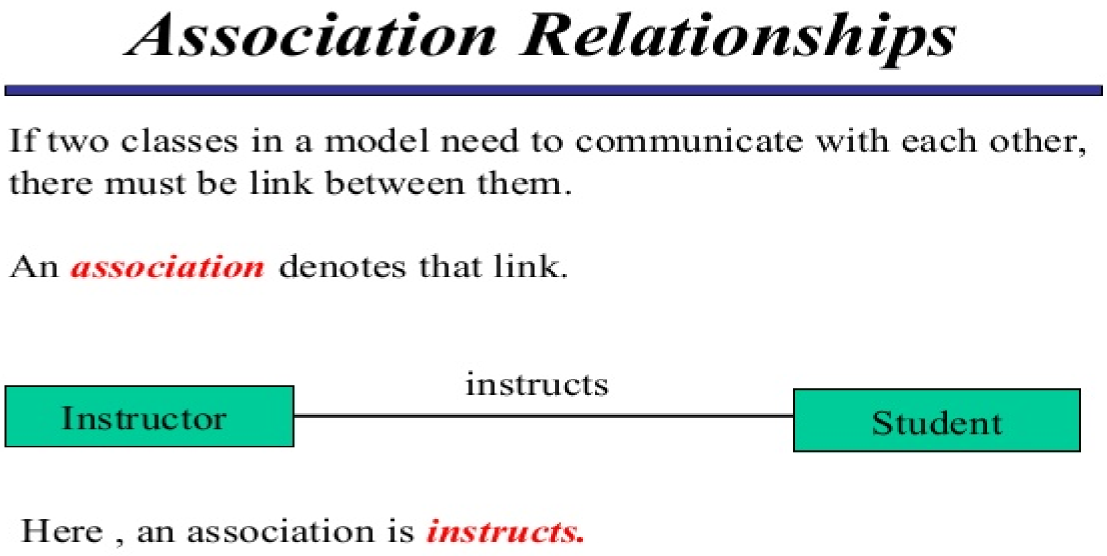
1. Association
2. Aggregation
3. Composition
4. Inheritance

Class diagram

Classes are having 4 relationships

1. Association:

- It is a set of links that connect classes of the class diagram.
- Read/write relationships from left to right and from top to bottom



Association Relationships

If two classes in a model need to communicate with each other, there must be link between them.

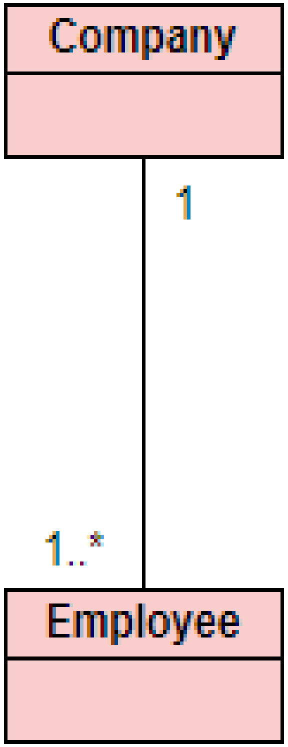
An ***association*** denotes that link.

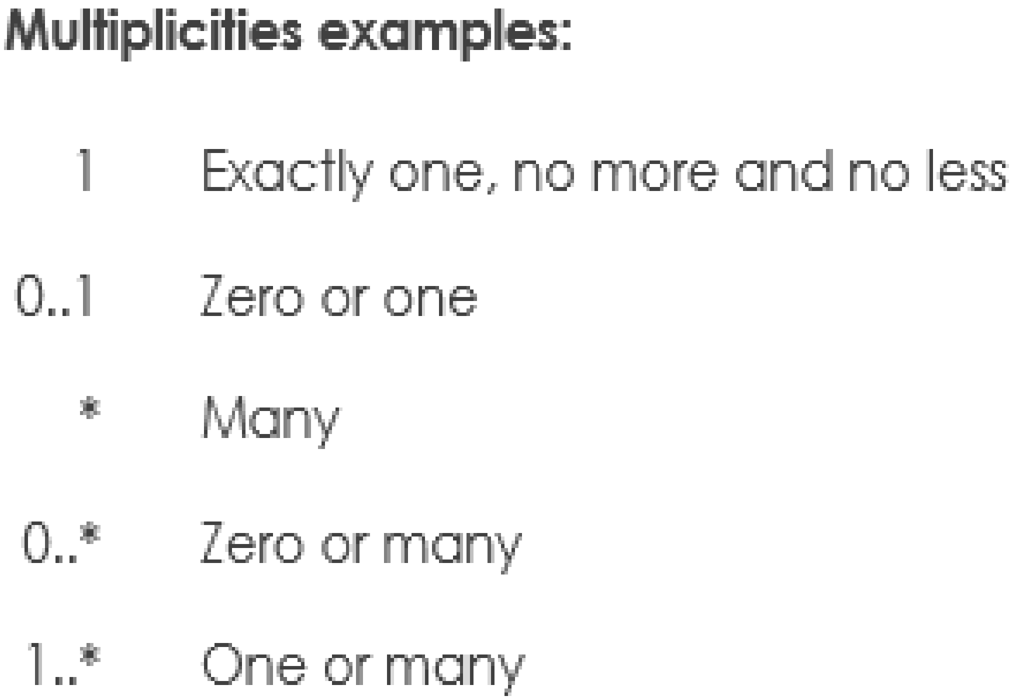
Here , an association is ***instructs.***

Class diagram

1. Association: multiplicity

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.





Multiplicities examples:

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

Class diagram

Classes are having 4 relationships

2.Aggregation:

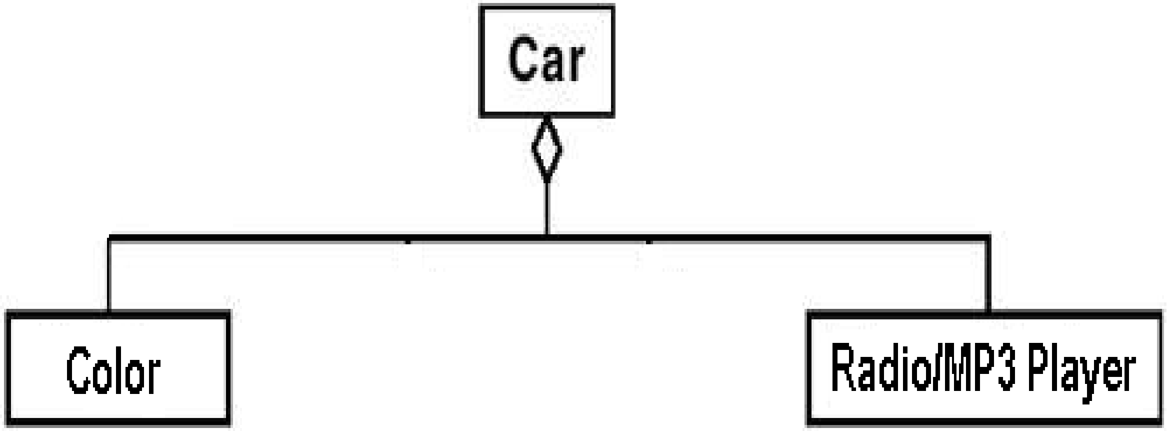
- Aggregation is identified by the phrase “is-part-of”.
- Aggregation implies a relationship where the child can exist independently of the parent.
- In Aggregation , parent and child entity maintain Has-A relationship but both can also exist independently.
- We can use parent and child entity independently. Any modification in the parent entity will not impact the child entity or vice versa.

Class diagram

Classes are having 4 relationships

2.Aggregation:

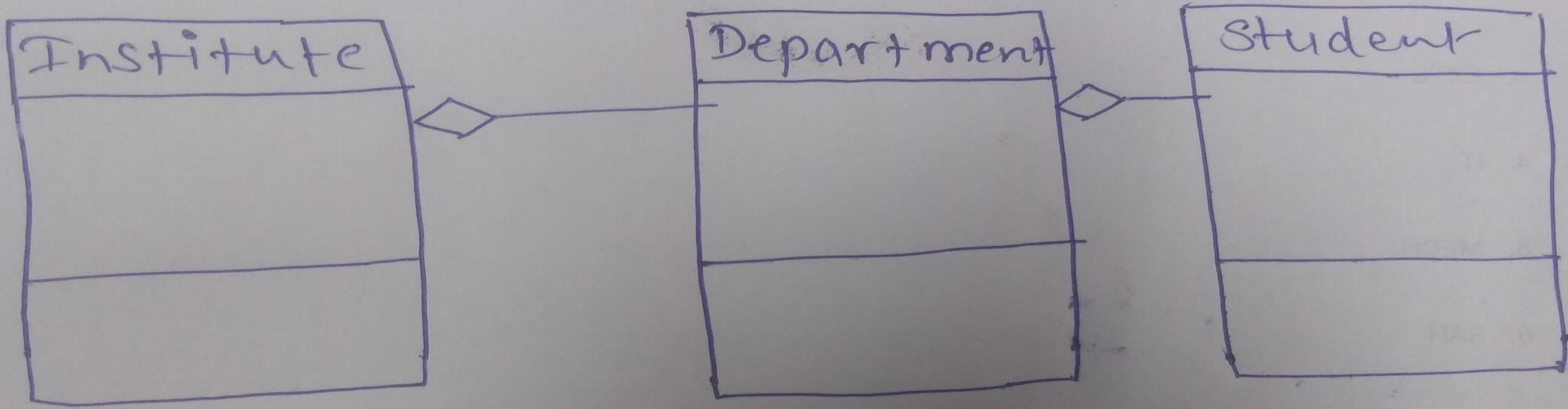
- In the UML diagram, aggregation is denoted by an empty diamond, which shows their obvious difference in terms of strength of the relationship.



Class diagram

Classes are having 4 relationships

2.Aggregation:



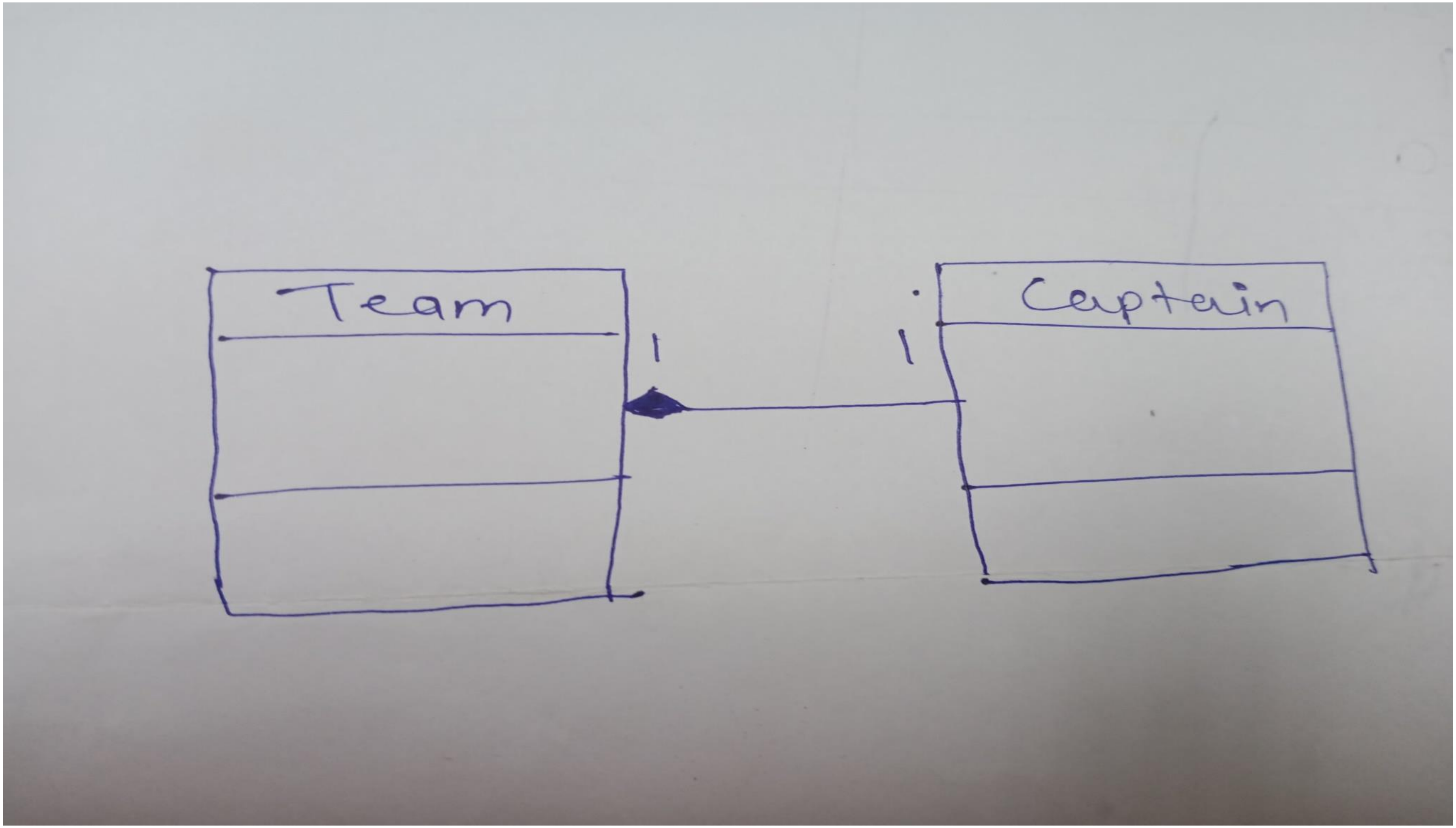
Class diagram

3.Composition:

- “is-whole-part-of” relationship.
- It is a two-way association between the objects.
- Composition implies a relationship where the child cannot exist independent of the parent.
- In Composition, parent owns child entity so child entity can’t exist without parent entity. We can’t directly or independently access child entity. In the UML diagram, composition is denoted by a filled diamond.

Class diagram

3.Composition:



Class diagram

Classes are having 4 relationships

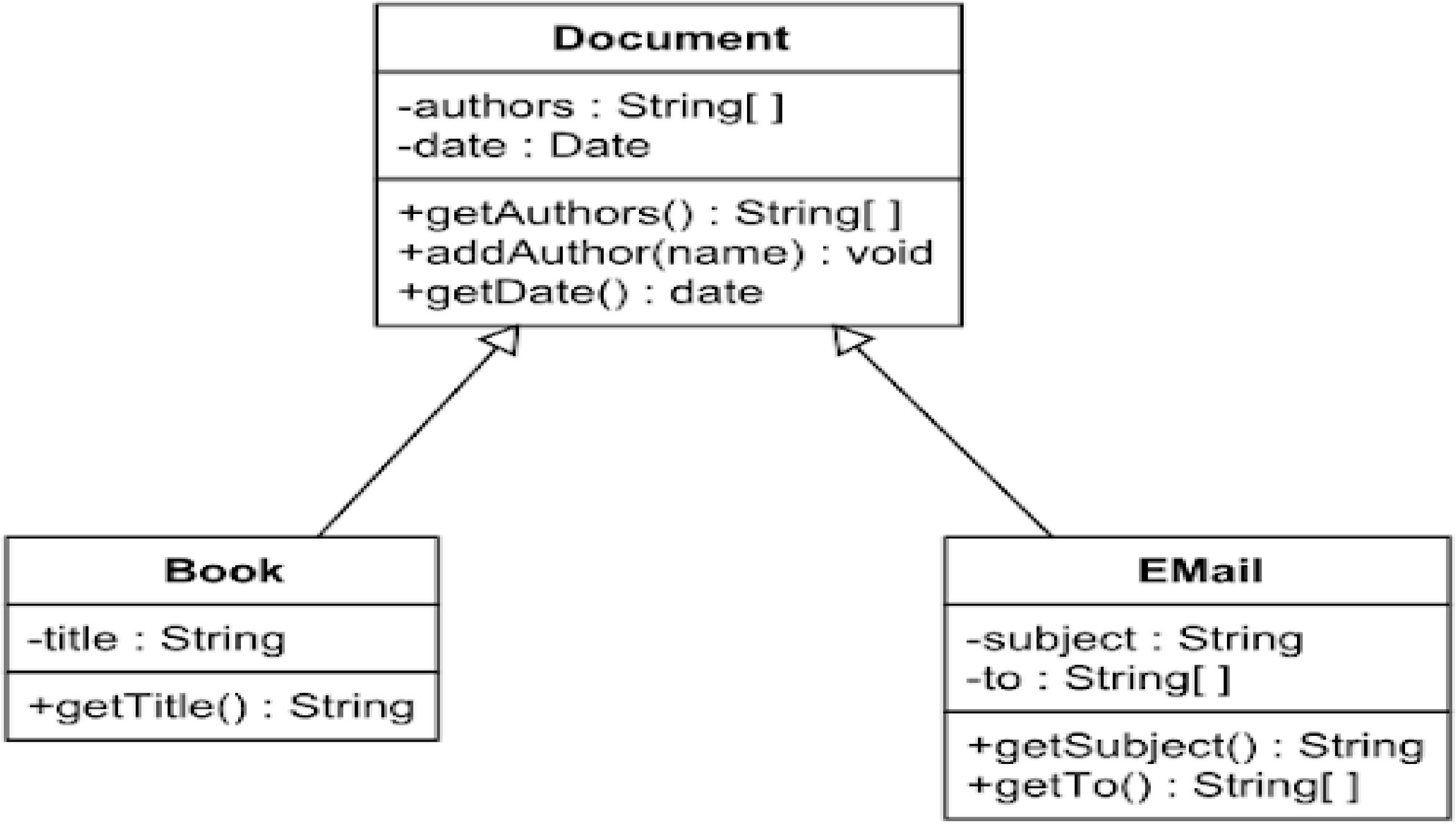
4. Generalization/Inheritance:

- Inheritance is a mechanism which allows a class A to inherit members (data and functions) of a class B.
- We say “A inherits from B”.
- Objects of class A thus have access to members of class B without the need to redefine them.
- refers to a type of relationship wherein one associated class is a child of another by virtue of assuming the same functionalities of the parent class
- a solid line from the child class to the parent class is drawn using an unfilled arrowhead.

Class diagram

Classes are having 4 relationships

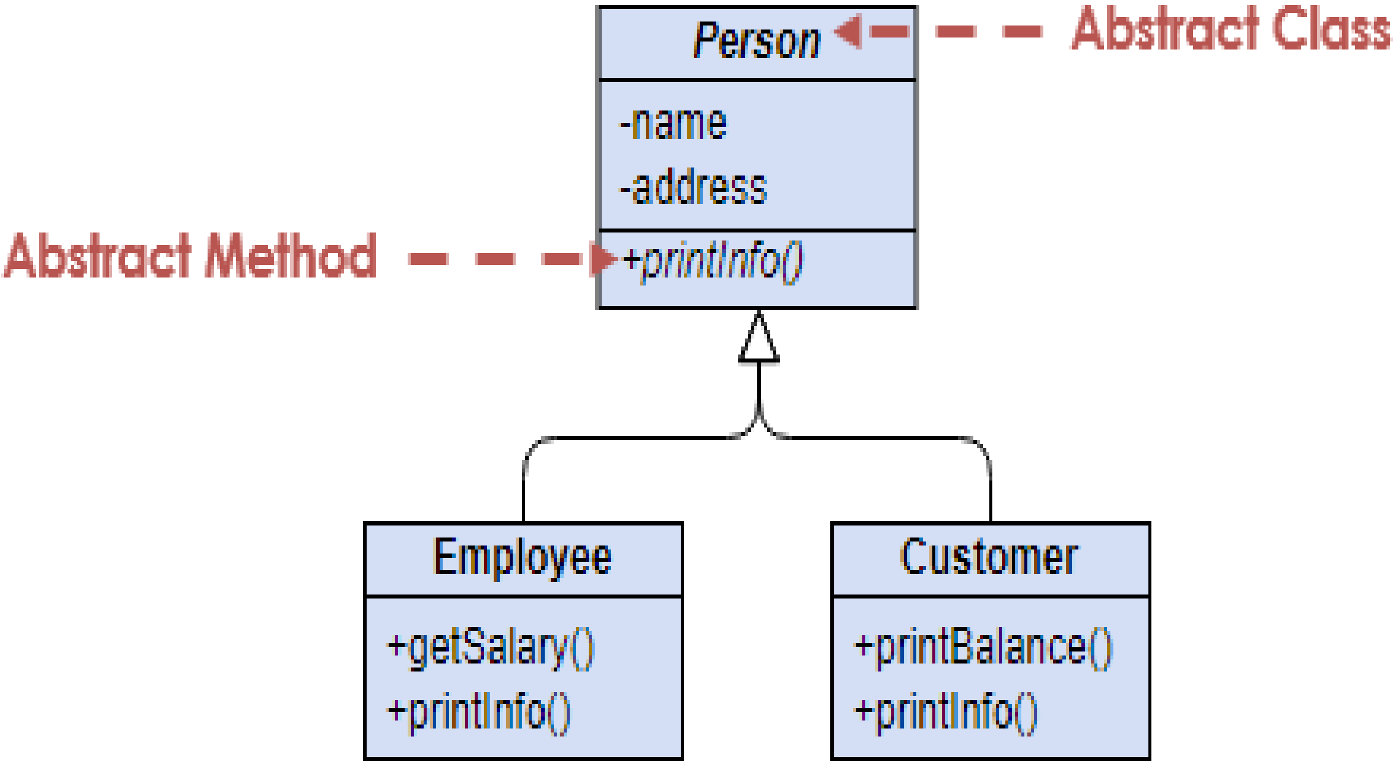
4. Generalization/Inheritance:



Class diagram

Classes are having 4 relationships

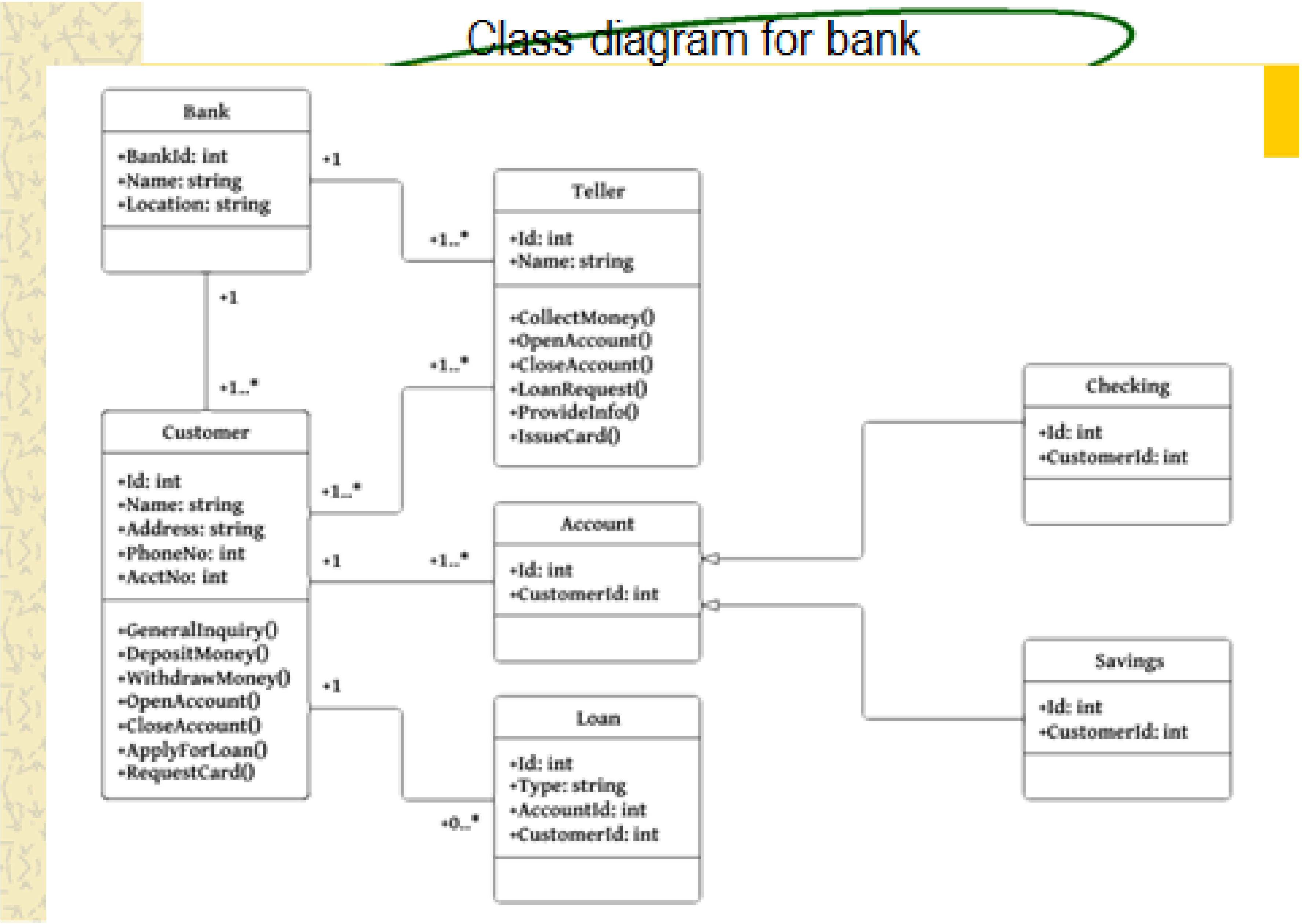
4. Generalization/Inheritance for abstract class/method:



Class diagram

Draw a class diagram for hospital management system.

Class diagram



Class diagram for bank

Bank
- +BankId: int
- +Name: string
- +Location: string

Teller
- +Id: int
- +Name: string
- +CollectMoney()
- +OpenAccount()
- +CloseAccount()
- +LoanRequest()
- +ProvideInfo()
- +IssueCard()

Customer
- +Id: int
- +Name: string
- +Address: string
- +PhoneNo: int
- +AcctNo: int
- +GeneralInquiry()
- +DepositMoney()
- +WithdrawMoney()
- +OpenAccount()
- +CloseAccount()
- +ApplyForLoan()
- +RequestCard()

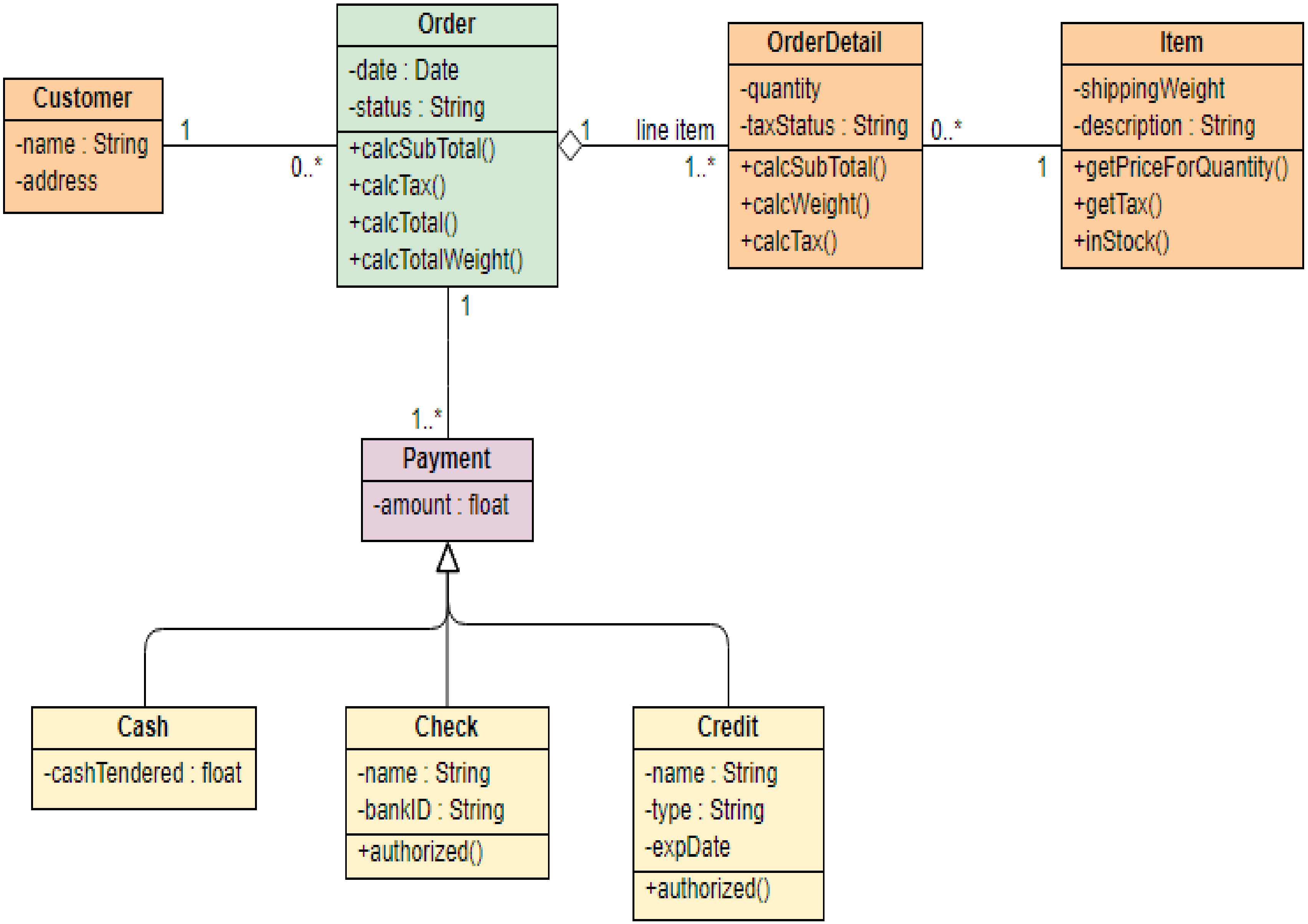
Account
- +Id: int
- +CustomerId: int

Checking
- +Id: int
- +CustomerId: int

Savings
- +Id: int
- +CustomerId: int

Loan
- +Id: int
- +Type: string
- +AccountId: int
- +CustomerId: int

Class diagram for online orders



Customer
-name : String
-address

Order
-date : Date
-status : String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight()

OrderDetail
-quantity
-taxStatus : String
+calcSubTotal()
+calcWeight()
+calcTax()

Item
-shippingWeight
-description : String
+getPriceForQuantity()
+getTax()
+inStock()

Payment
-amount : float

Cash
-cashTendered : float

Check
-name : String
-bankID : String
+authorized()

Credit
-name : String
-type : String
-expDate
+authorized()