جامعة
الملك سعود
King Saud University

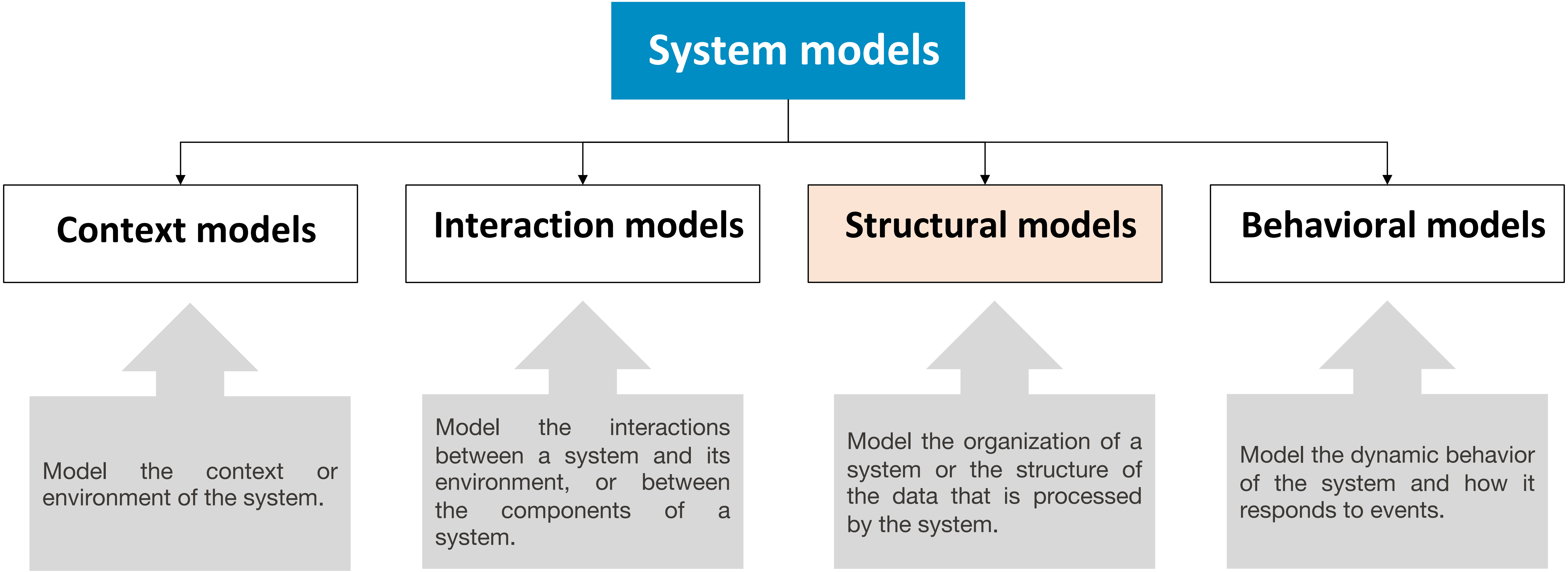
College of Computer and
Information Sciences

System Modeling

Section 5.2 (Sommerville)

Fall Semester 2021
1st Semester 1443 H

System modeling



System models

Context models

Interaction models

Structural models

Behavioral models

Model the context or environment of the system.

Model the interactions between a system and its environment, or between the components of a system.

Model the organization of a system or the structure of the data that is processed by the system.

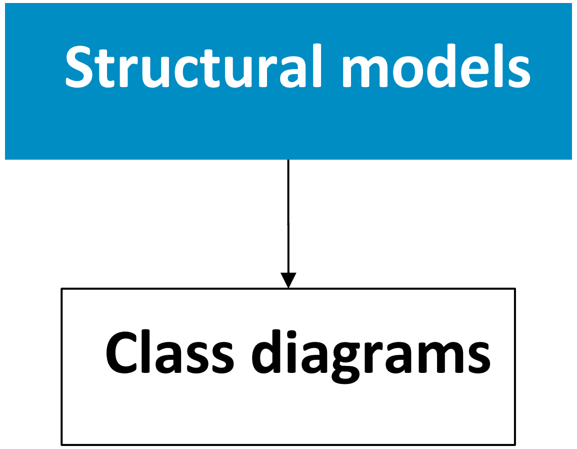
Model the dynamic behavior of the system and how it responds to events.

Structural models

- Display the organization of a system in terms of the ***components*** that make up that system and their ***relationships***.

- Structural models may be
  - **Static models:** show the structure of the system design.
  - **Dynamic models:** show the organization of the system when it is executing.

- You create structural models of a system when you are discussing and designing the system architecture.



Structural models

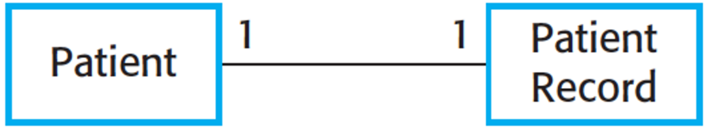
Class diagrams

Class diagrams

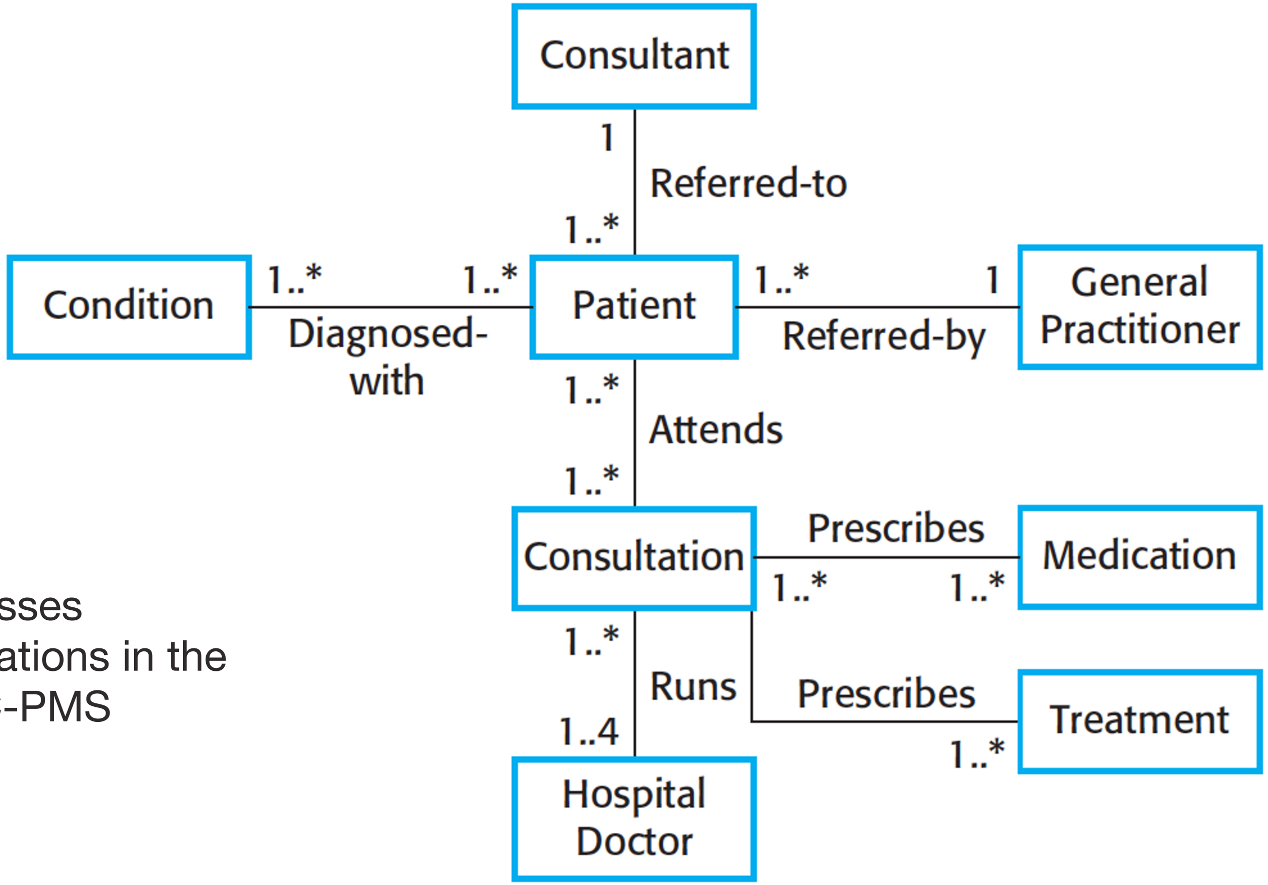
- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between them.

- A ***class*** can be thought of as a general definition of a system object.
  - The object represents something in the real world, such as a patient, a prescription, doctor, etc.

- An ***association*** is a link (relationship) between classes.

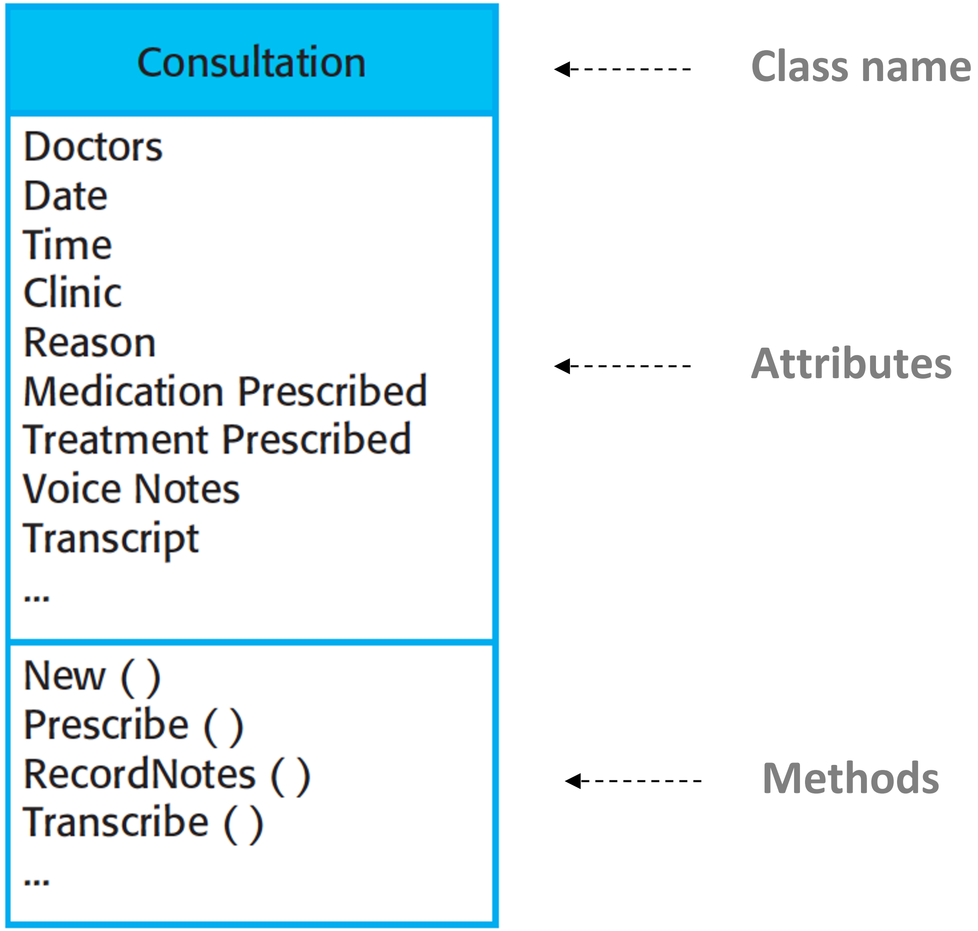


Class diagrams



Classes and associations in the MHC-PMS

Class diagrams



Consultation

Doctors
Date
Time
Clinic
Reason
Medication Prescribed
Treatment Prescribed
Voice Notes
Transcript
...

New ( )
Prescribe ( )
RecordNotes ( )
Transcribe ( )
...

Class name

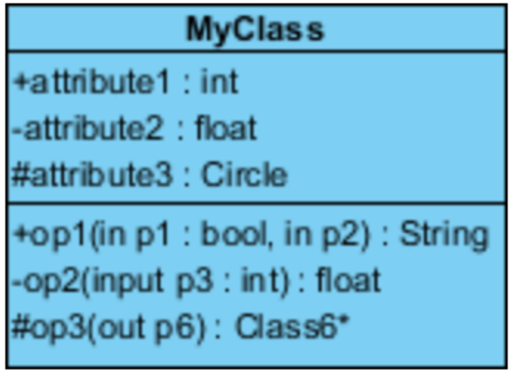
Attributes

Methods

Consultation class

Visibility

- ***Public*** attributes and methods are denoted with +.
- ***Private*** attributes and methods are denoted with -.
- ***Protected*** attributes and methods are denoted with #.



MyClass

+attribute1 : int
-attribute2 : float
#attribute3 : Circle

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

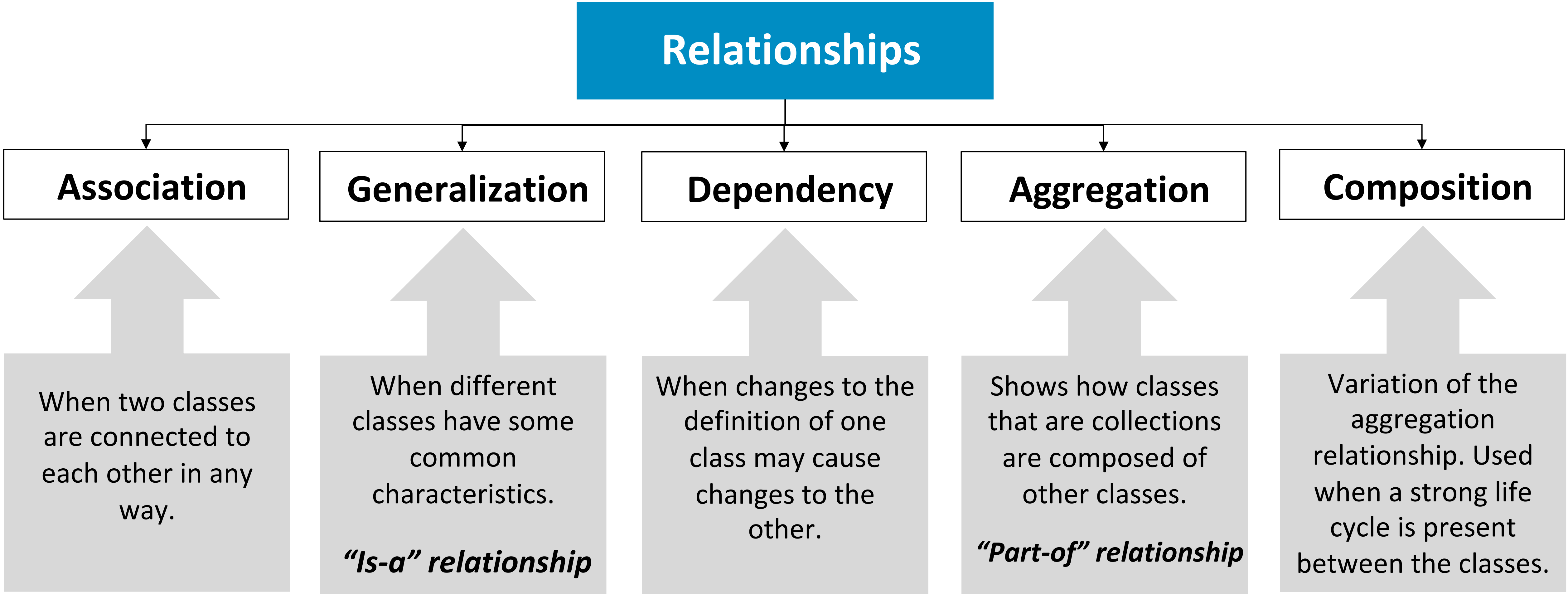
Access right	Public	Private	Protected
Members of the same class	√	√	√
Members of derived classes	√		√
Members of any other class	√		

Multiplicity

- How many objects of each class take part in the relationship.

Multiplicity	Notation
Exactly one	1
Zero or one	0..1
Many	0..* or *
One or more	1..*
Exact number	2..3 or 2

Class relationships



Relationships

Association

When two classes are connected to each other in any way.

Generalization

When different classes have some common characteristics.

"Is-a" relationship

Dependency

When changes to the definition of one class may cause changes to the other.

Aggregation

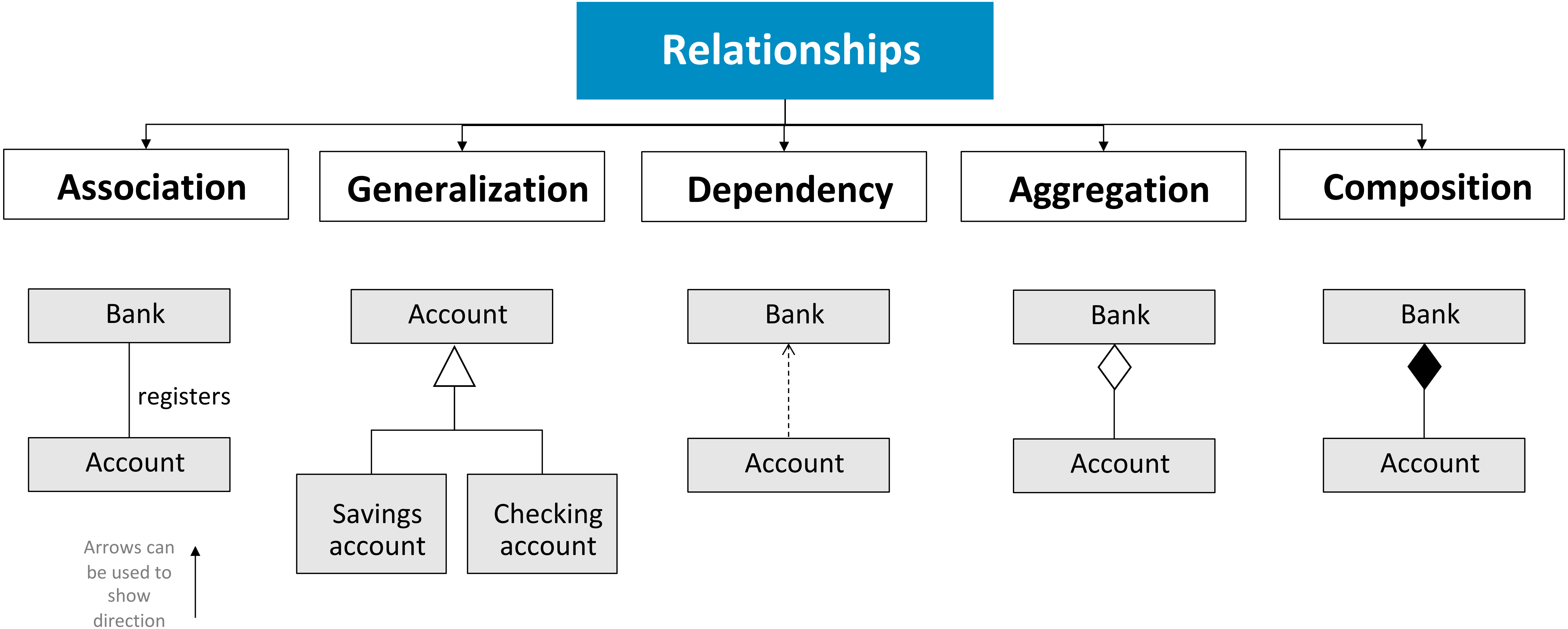
Shows how classes that are collections are composed of other classes.

"Part-of" relationship

Composition

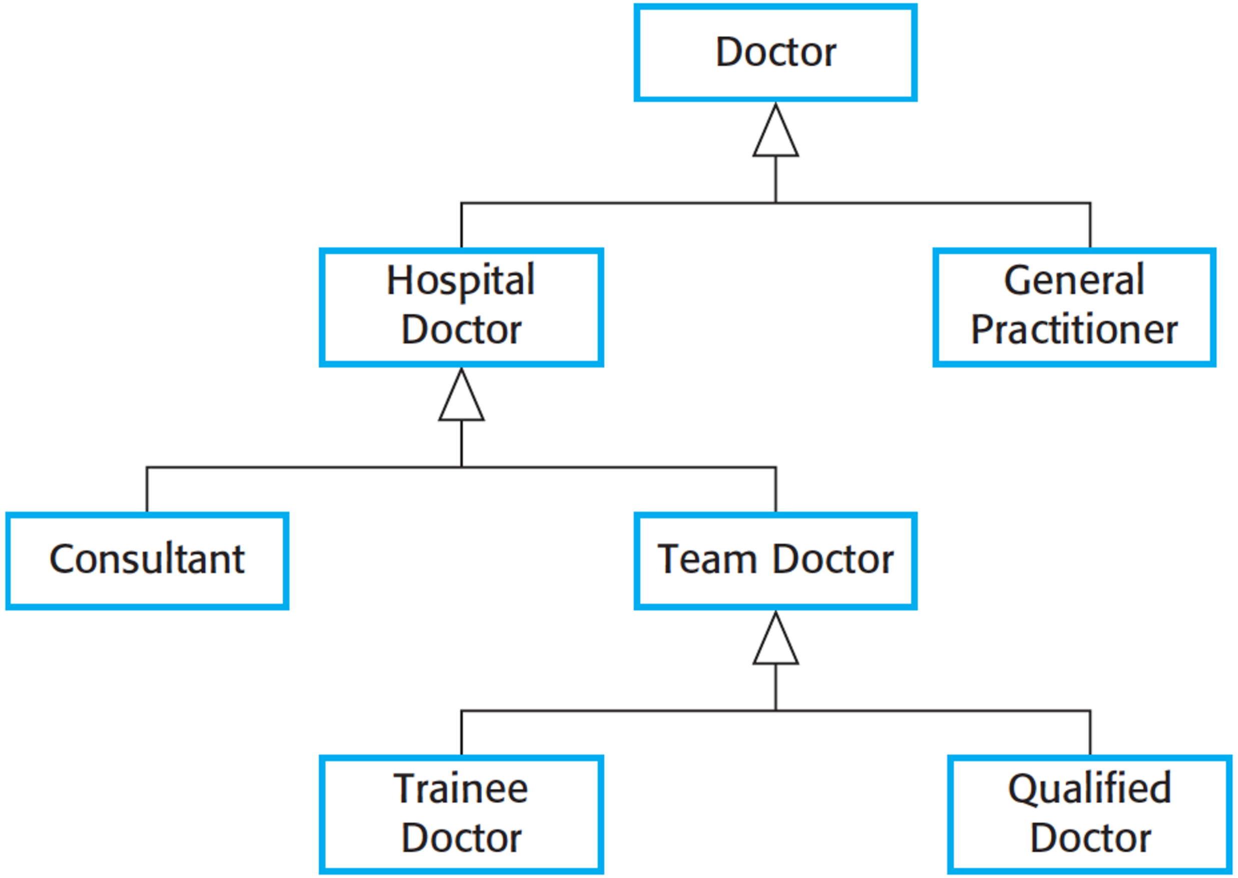
Variation of the aggregation relationship. Used when a strong life cycle is present between the classes.

Class relationships - *notations*

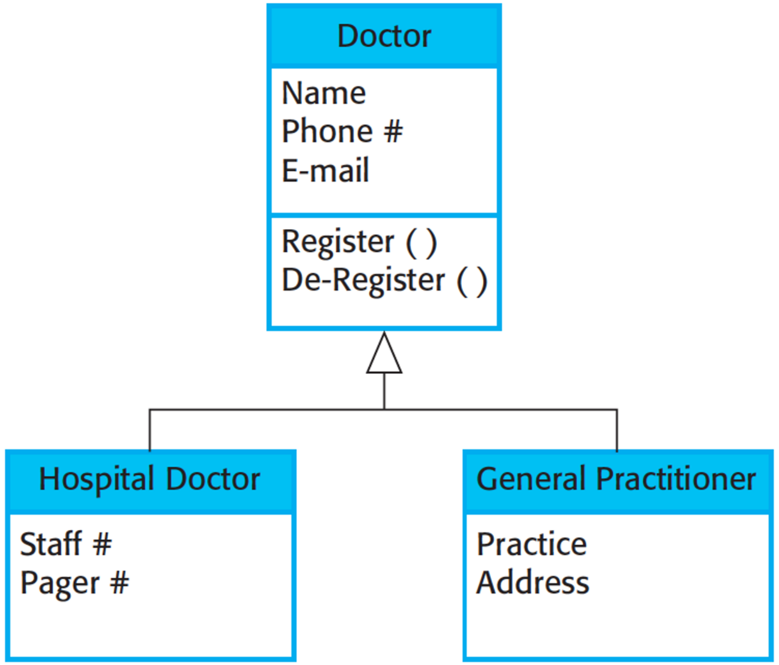


Arrows can be used to show direction

Generalization

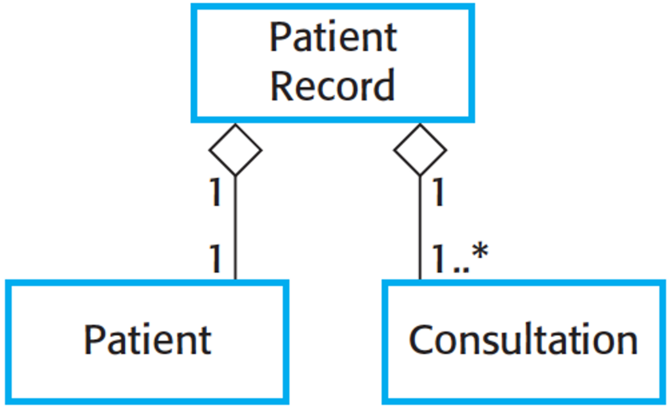


A generalization hierarchy



A generalization hierarchy with added detail

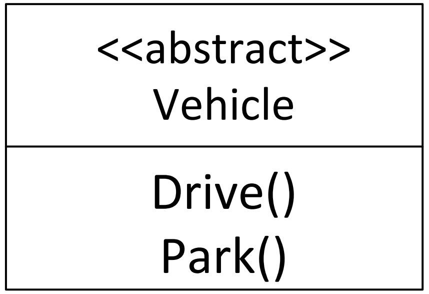
Aggregation



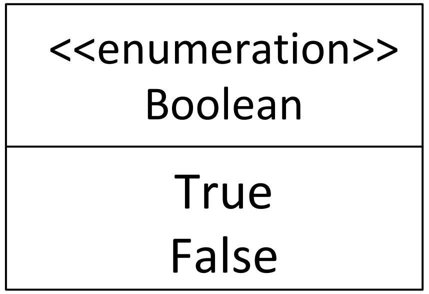
The aggregation association

Other class types

- **Abstract class** cannot be instantiated, but it can be sub-classed. It is used when an inheritance relationship serves only to model shared attributes and operations. The name of the abstract class is written in an italic font.



- **Enumeration class** is a user-defined data type that consists of a name and an ordered list of enumeration literals.

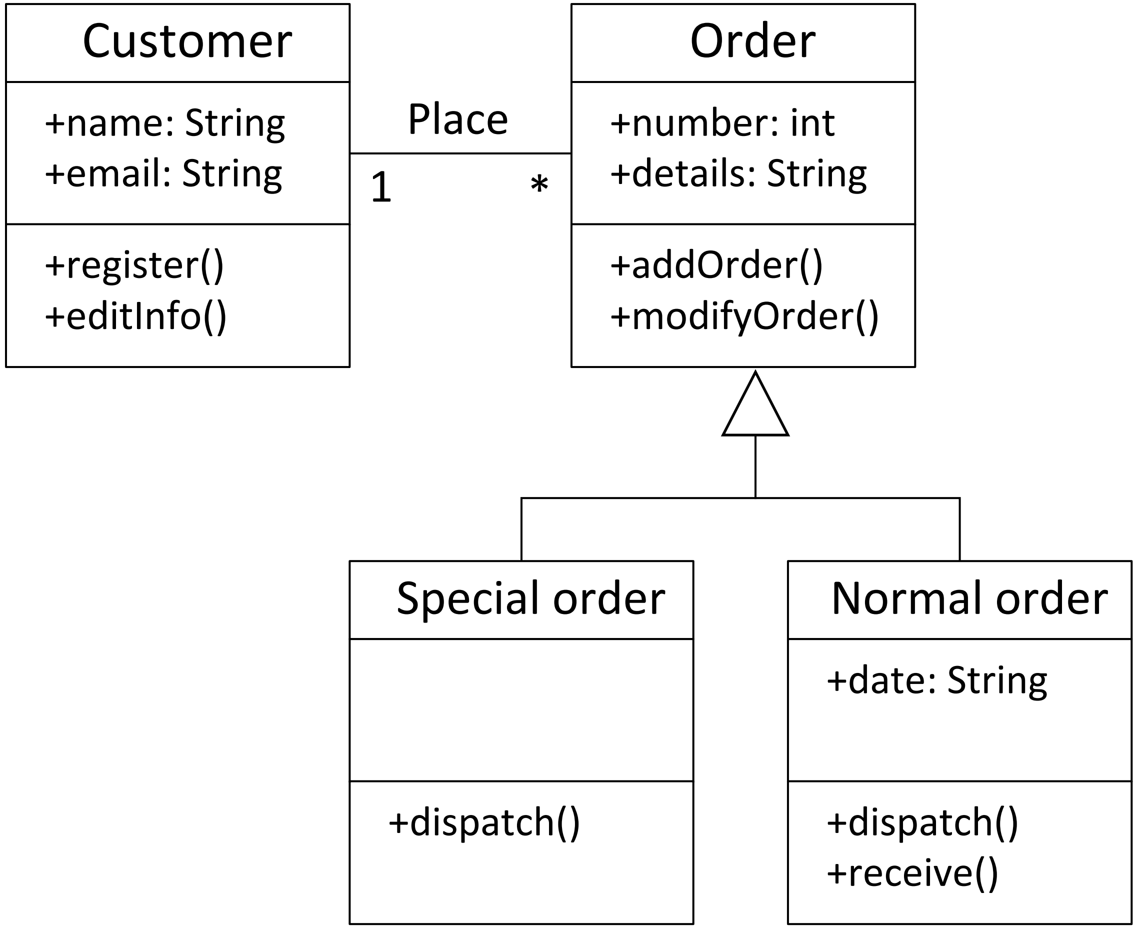


Class diagram

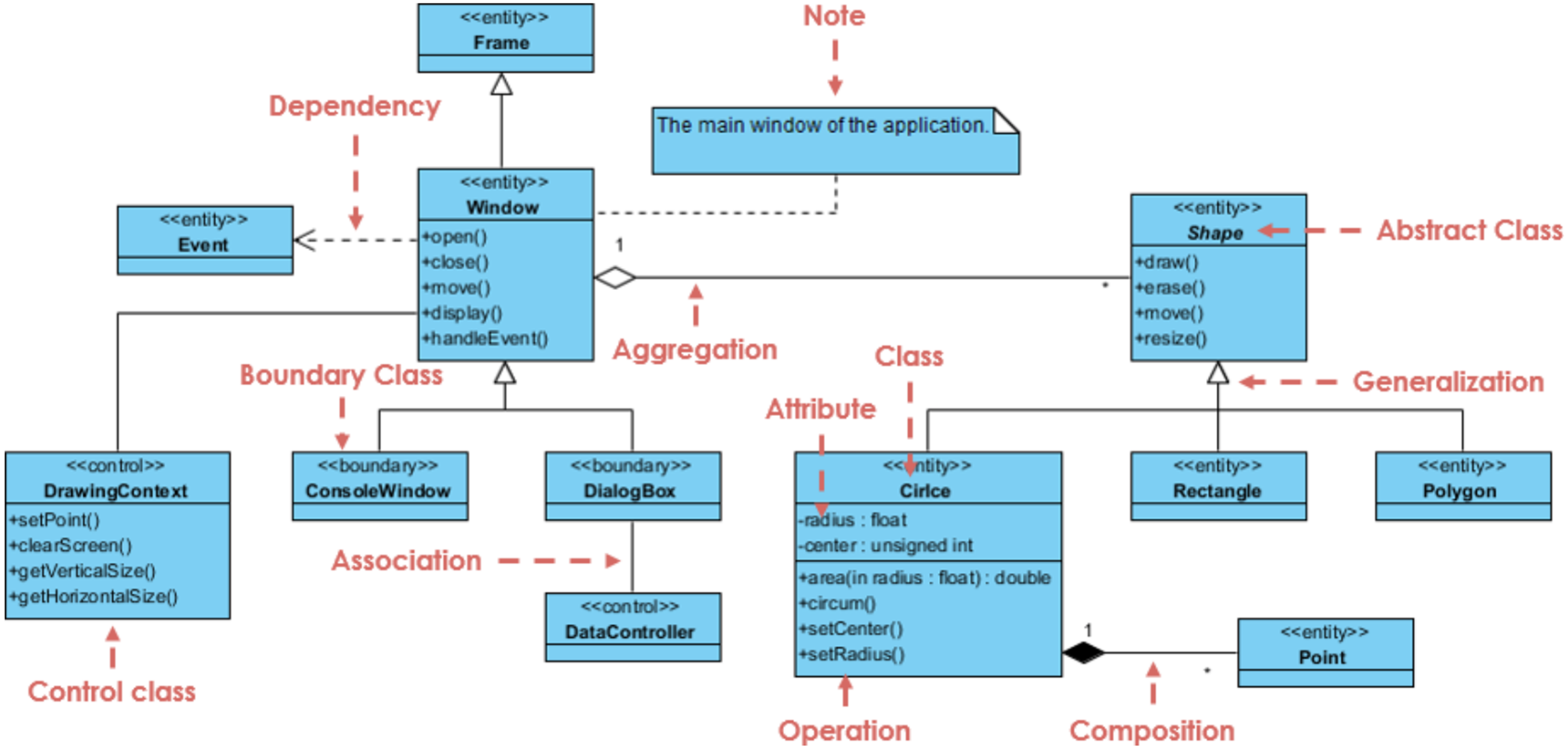
The following diagram is an example of an *Order System* of an application.

The system has two main elements: the customer and the order. A customer can have multiple orders. There are two types of orders: special order and normal order.

Both types have all properties of the order. In addition, they have additional functions like dispatch and receive.



Class diagram – *GUI tool example*


<<entity>>
Frame
Note
Dependency
The main window of the application.
<<entity>>
Window
+open()
+close()
+move()
+display()
+handleEvent()
<<entity>>
Event
1
<<entity>>
Shape
+draw()
+erase()
+move()
+resize()
Abstract Class
*
Aggregation
Class
Boundary Class
Generalization
Attribute
<<control>>
DrawingContext
+setPoint()
+clearScreen()
+getVerticalSize()
+getHorizontalSize()
<<boundary>>
ConsoleWindow
<<boundary>>
DialogBox
<<entity>>
Circle
-radius : float
-center : unsigned int
+area(in radius : float) : double
+circum()
+setCenter()
+setRadius()
<<entity>>
Rectangle
<<entity>>
Polygon
Association
<<control>>
DataController
1
<<entity>>
Point
*
Control class
Operation
Composition