- Systems Analysis and Design
- Course Code: CSE 305

Object-Oriented Systems Analysis and Design Using UML (Class Diagram)

Instructor:
Dr. Shah Murtaza Rashid Al Masud
Assoc. Prof. CSE Dept., UAP

Class Diagrams

In software engineering, a class diagram in the Unified Modeling Language (UML) 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.

A class diagram is a UML diagram type that describes a system by visualizing the different types of objects within a system and the kinds of static relationships that exist among them. It also illustrates the operations and attributes of the classes.

Purpose of Class Diagrams

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.

How to Draw a Class Diagram?

The following points should be remembered while drawing a class diagram –

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

How to Draw a Class Diagram?

Step 1: Identify the class names

The first step is to identify the primary objects of the system.

Step 2: Distinguish relationships

Next step is to determine how each of the classes or objects are related to one another.

Step 3: Create the Structure

First, add the class names and link them with the appropriate connectors. You can add attributes and functions/ methods/ operations later.

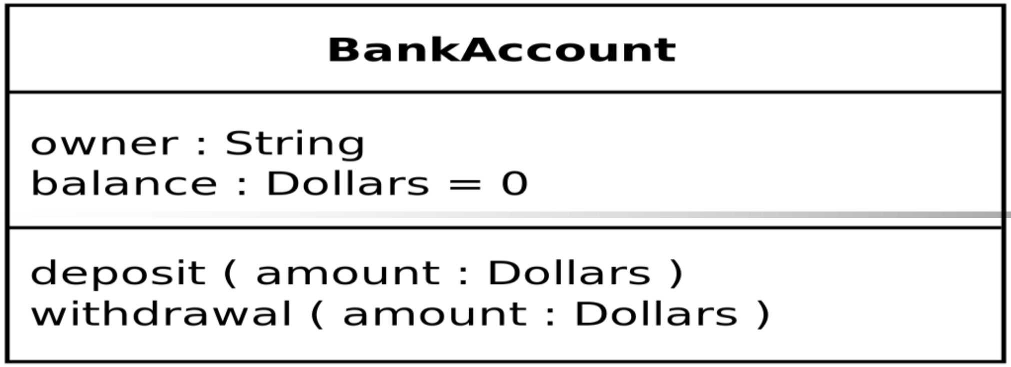
How to Draw a Class Diagram?

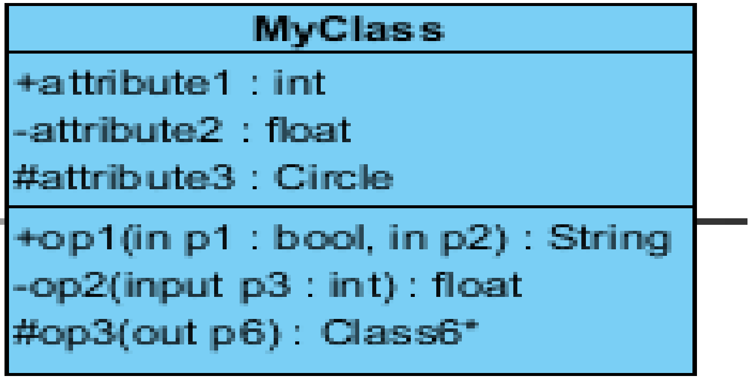
- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase. The attribute type is shown after the colon.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

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.

How to Draw a Class Diagram?





The graphical representation of the class - MyClass as shown above:

•MyClass has 3 attributes and 3 operations

•op2 returns a float

•Parameter p3 of op2 is of type int

•op3 returns a pointer (denoted by a *) to Class6

How to Draw a Class Diagram?

Visibility of Class attributes and Operations

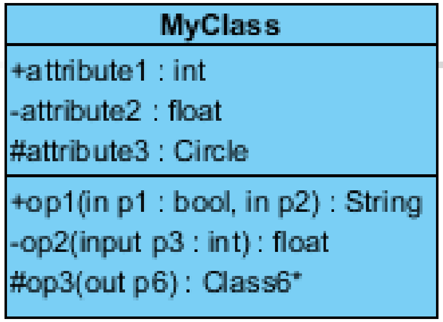
In object-oriented design, there is a notation of visibility for attributes and operations. UML identifies four types of visibility: **public**, **protected**, **private**, and **package**.

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

How to Draw a Class Diagram?

Visibility of Class attributes and Operations



In the example above:

•attribute1 and op1 of MyClassName are public

•attribute3 and op3 are protected.

•attribute2 and op2 are private.

Class Diagrams example

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

- First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
- Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().

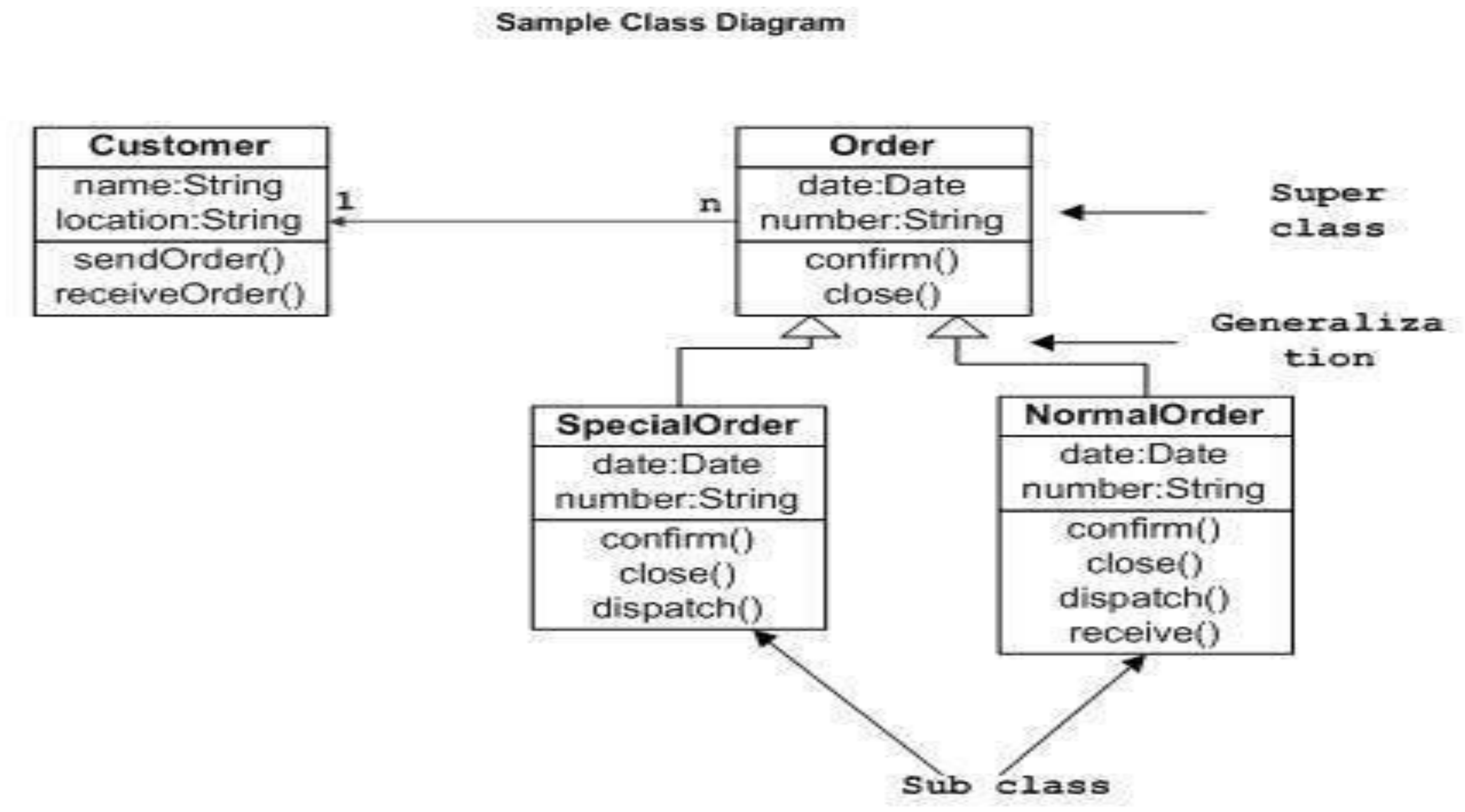
Class Diagrams example

An abstract class is meant to be used as the base class from which other classes are derived.

The derived class is expected to provide implementations for the member functions that are not implemented in the base class.

A derived class that implements all the missing functionality is called a *concrete class* .

Class Diagrams example



Class Diagrams example

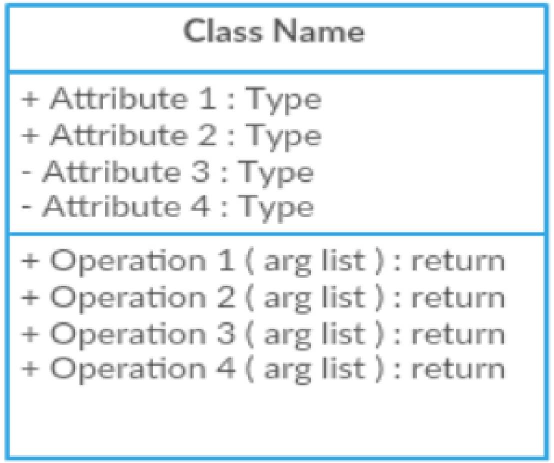
Where to Use Class Diagrams?

- Class diagram is a static diagram and it is used to model the static view of a system.
- Showing the collaboration among the elements of the static view.
- Describing the functionalities performed by the system.
- Construction of software applications using object oriented languages.

Class Diagram Notations with Examples

There are several class diagram notations that are used when drawing UML class diagrams. We've listed below the most common class diagram notations.

Class

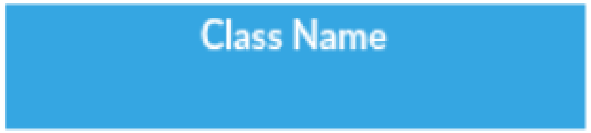

Class Name
+ Attribute 1 : Type
+ Attribute 2 : Type
- Attribute 3 : Type
- Attribute 4 : Type
+ Operation 1 (arg list) : return
+ Operation 2 (arg list) : return
+ Operation 3 (arg list) : return
+ Operation 4 (arg list) : return

Classes represent the central objects in a system. It is represented by a rectangle with up to 3 compartments.

The first one shows the class's name, while the middle one shows the class's attributes which are the characteristics of the objects. The bottom one lists the class's operations, which represents the behavior of the class.

Class Diagram Notations with Examples

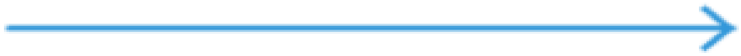
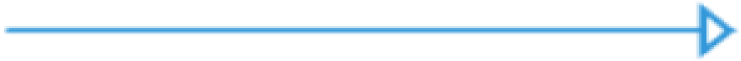
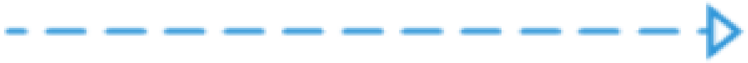
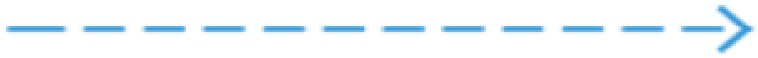
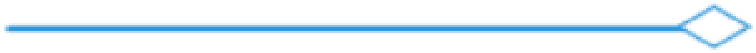
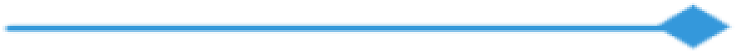
The first one shows the class's name, while the middle one shows the class's attributes which are the characteristics of the objects. The bottom one lists the class's operations, which represents the behavior of the class.



Simple Class

The last two compartments are optional. The class notation without the last two compartments is called a simple class and it only contains the name of the class.

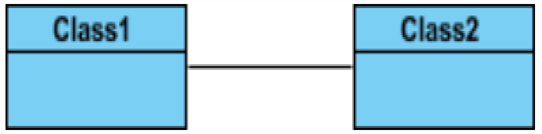
Class Diagram Relationships

Class Diagram Relationship Type	**Notation**
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	

Class Diagram Relationships

Simple Association:

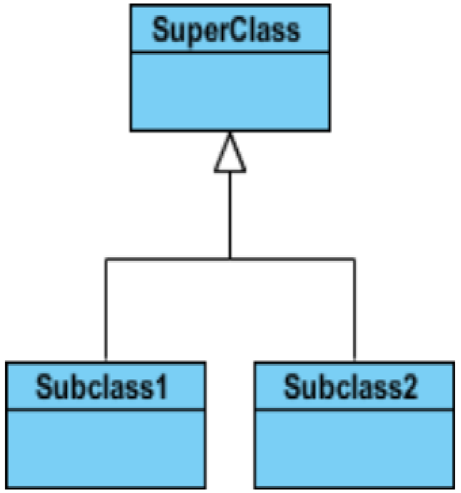
- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes

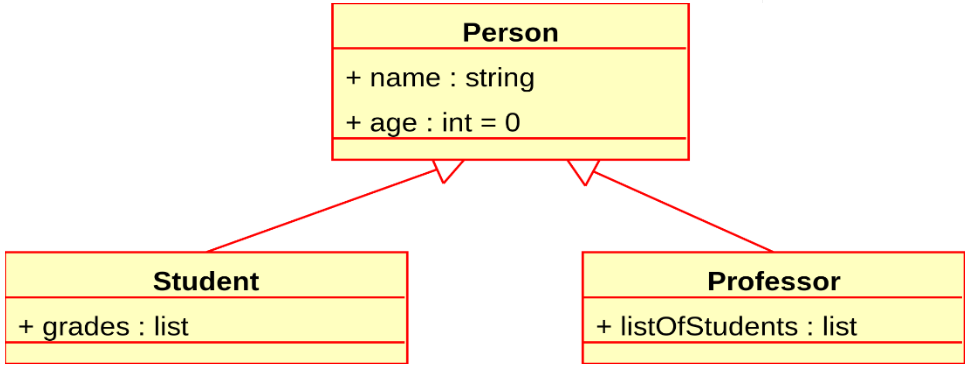


Class Diagram Relationships

Inheritance (or Generalization):

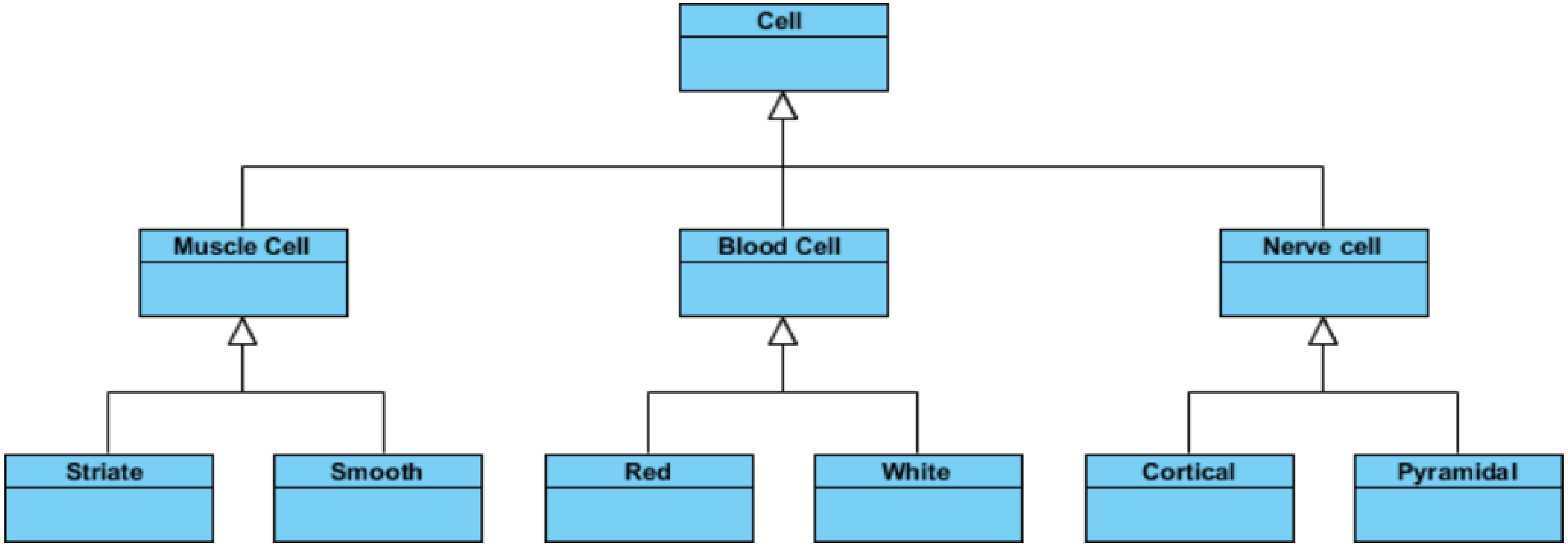
- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class





Inheritance Example - Cell Taxonomy

- Inheritance is another special case of an association denoting a "kind-of" hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The child classes inherit the attributes and operations of the parent class.

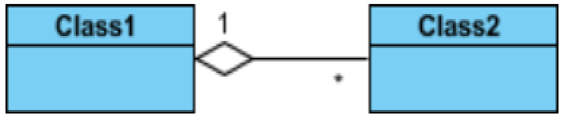


Class Diagram Relationships

Aggregation:

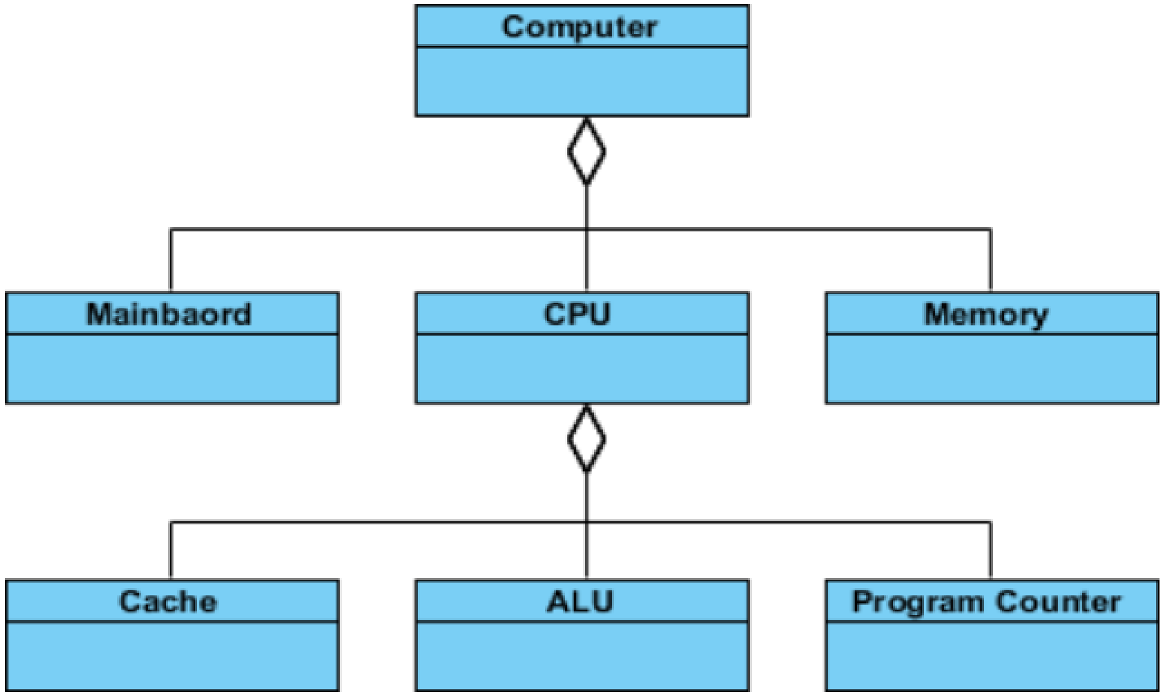
A special type of association. It represents a "part of" relationship.

- 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 Example - Computer and parts

- An aggregation is a special case of association denoting a "consists-of" hierarchy
- The aggregate is the parent class, the components are the children classes

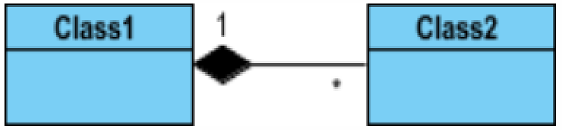


Class Diagram Relationships

Composition:

A special type of aggregation where parts are destroyed when the whole is destroyed.

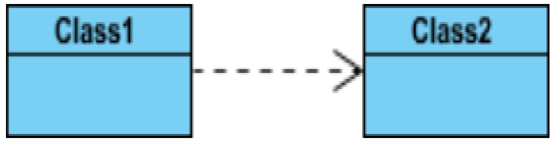
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite

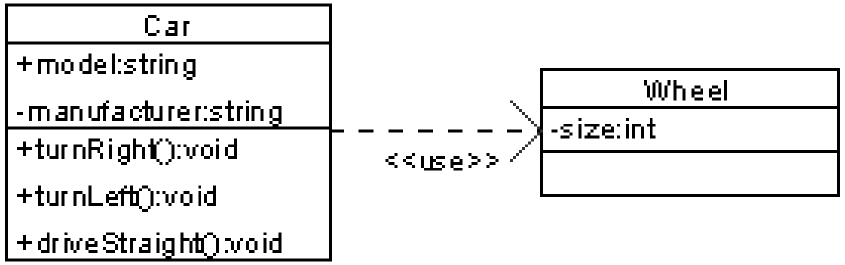


Class Diagram Relationships

Dependency:

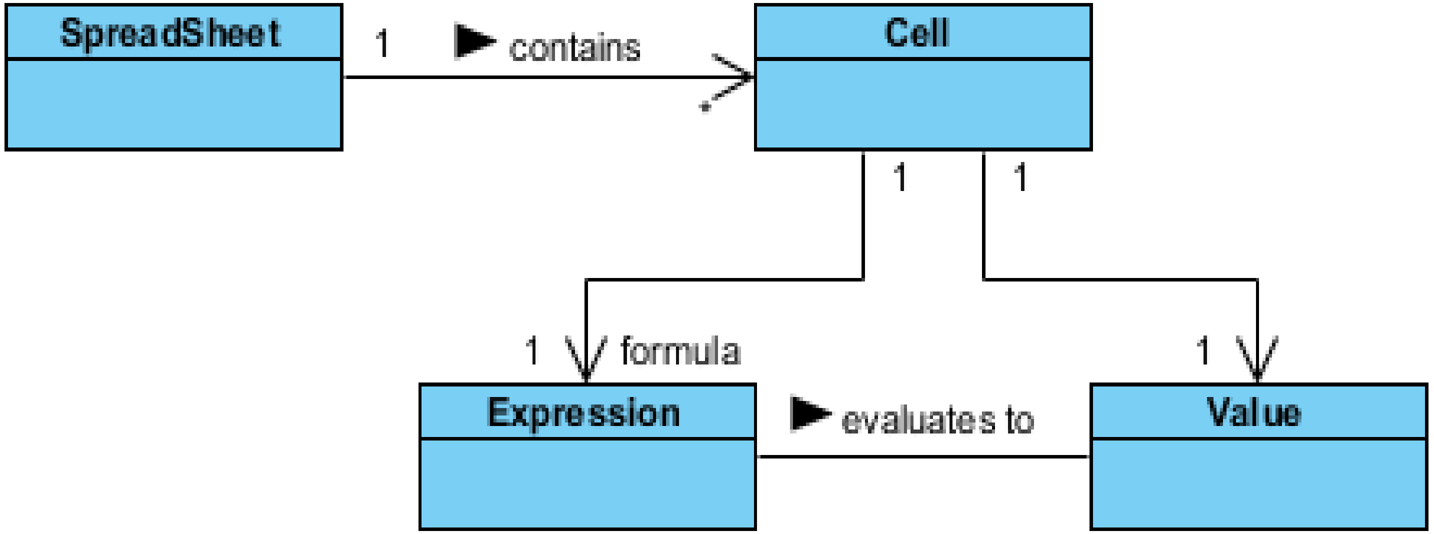
- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2
- A dashed line with an open arrow





Relationship Names

- Names of relationships are written in the middle of the association line.
- Good relation names make sense when you read them out loud:
  - "Every spreadsheet **contains** some number of cells",
  - "an expression **evaluates to** a value"
- They often have a **small arrowhead to show the direction** in which direction to read the relationship, e.g., expressions evaluate to values, but values do not evaluate to expressions.



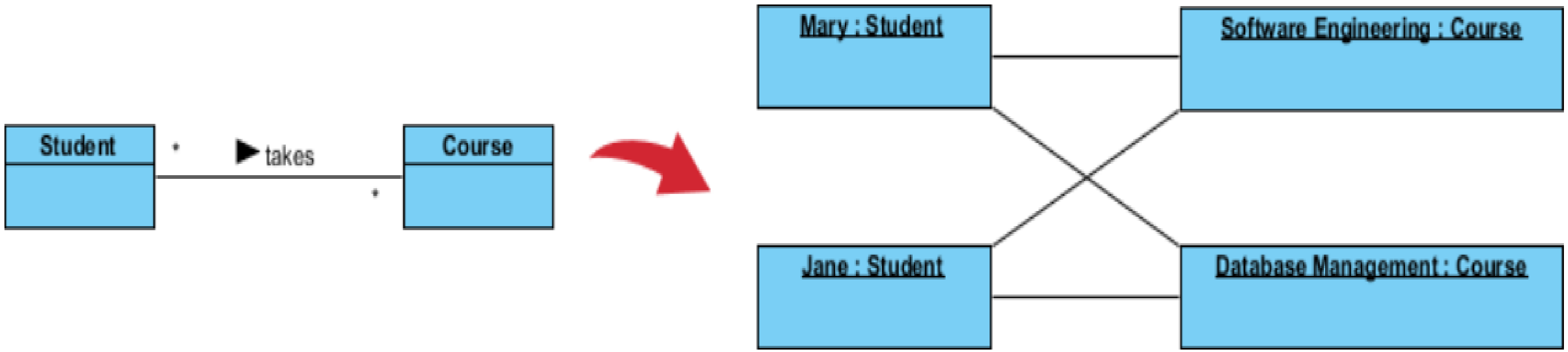
Multiplicity

How many objects of each class take part in the relationships and multiplicity can be expressed as:

- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One or more - 1..*
- Exact Number - e.g. 3..4 or 6
- Or a complex relationship - e.g. 0..1, 3..4, 6.* would mean any number of objects other than 2 or 5

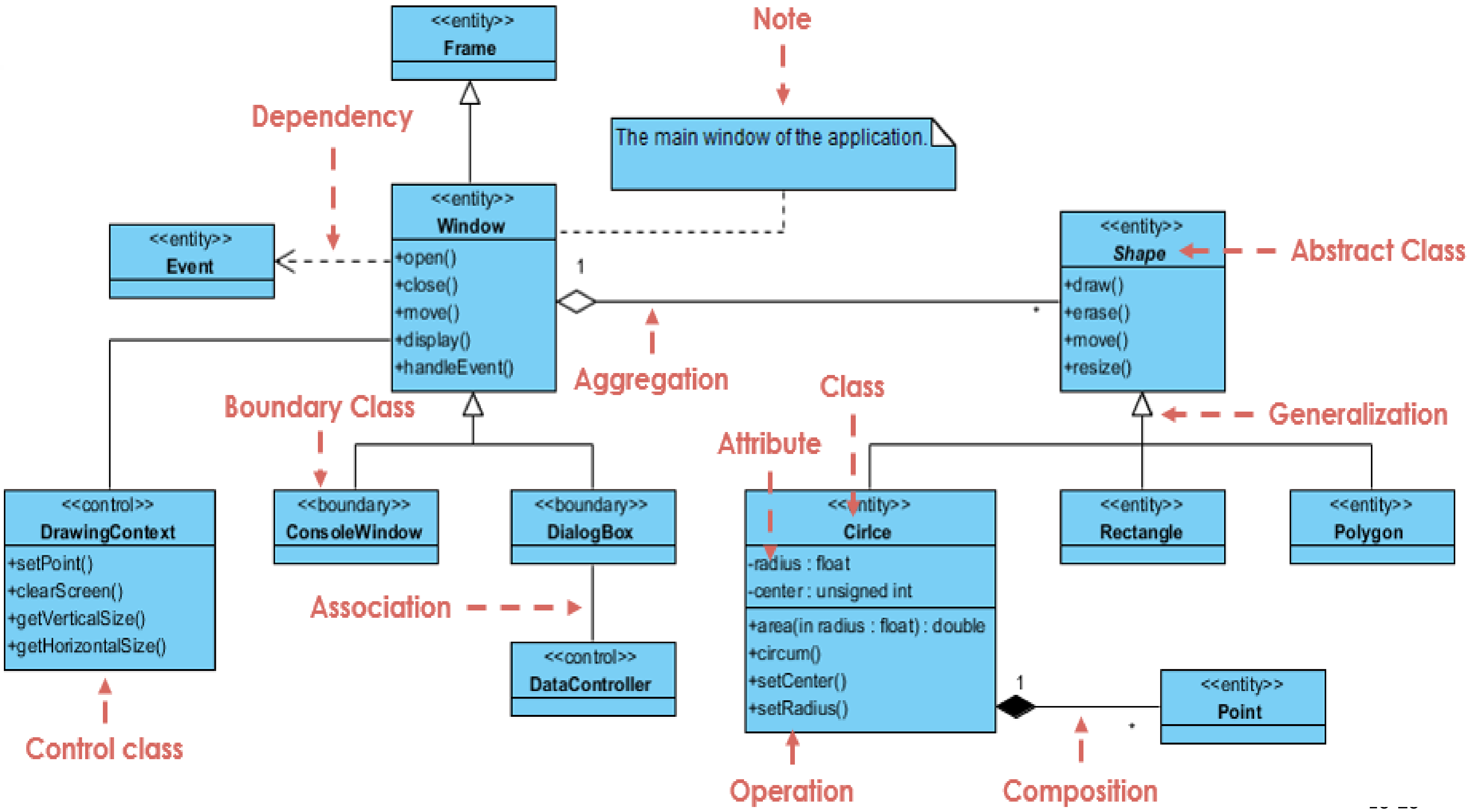
Multiplicity example

- Requirement: A Student can take many Courses and many Students can be enrolled in one Course.
- In the example below, the **class diagram** (on the left), describes the statement of the requirement above for the static model while the object diagram (on the right) shows the snapshot (an instance of the class diagram) of the course enrollment for the courses Software Engineering and Database Management respectively)



Class Diagram - Diagram Tool Example

A class diagram may also have notes attached to classes or relationships. Notes are shown in grey.

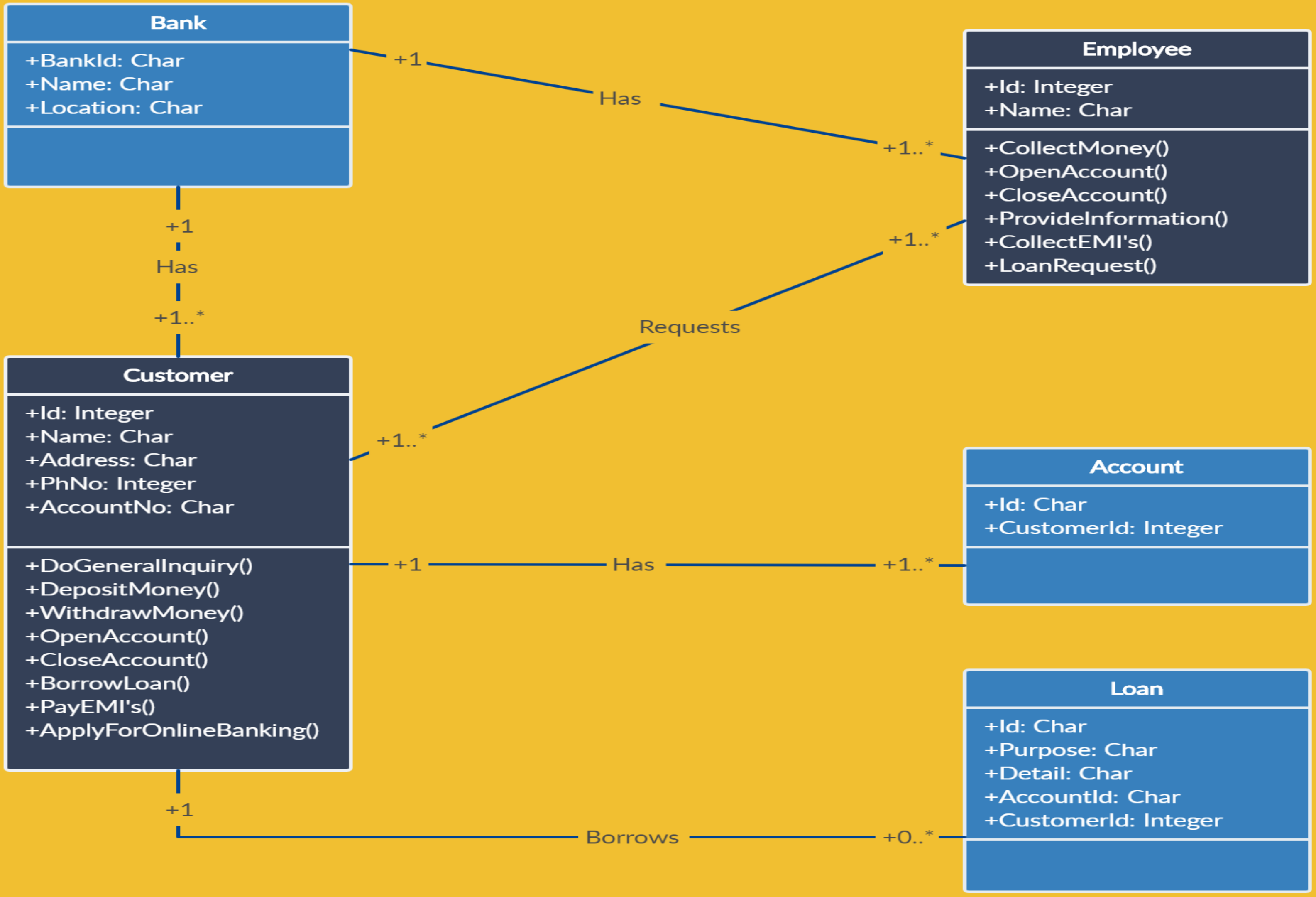


Class Diagram - Diagram Tool Example

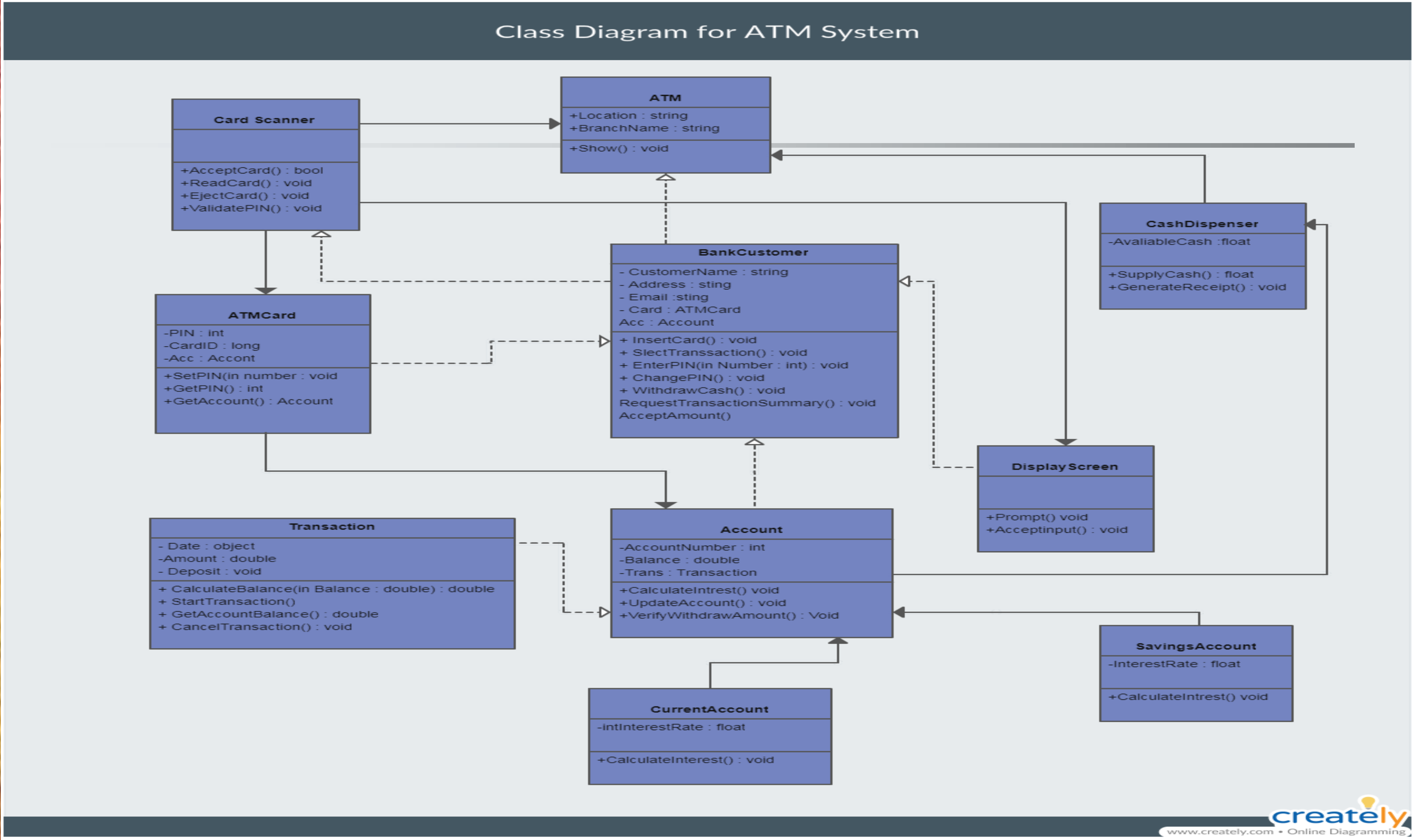
In the example above:

- We can interpret the meaning of the above class diagram by reading through the points as following.
- Shape is an abstract class. It is shown in Italics.
- Shape is a superclass. Circle, Rectangle and Polygon are derived from Shape. In other words, a Circle is-a Shape. This is a generalization / inheritance relationship.
- There is an association between DialogBox and DataController.
- Shape is part-of Window. This is an aggregation relationship. Shape can exist without Window.
- Point is part-of Circle. This is a composition relationship. Point cannot exist without a Circle.
- Window is dependent on Event. However, Event is not dependent on Window.
- The attributes of Circle are radius and center. This is an entity class.
- The method names of Circle are area(), circum(), setCenter() and setRadius().
- The parameter radius in Circle is an in parameter of type float.
- The method area() of class Circle returns a value of type double.
- The attributes and method names of Rectangle are hidden. Some other classes in the diagram also have their attributes and method names hidden.

Class Diagram more example



Class Diagram more example



Steps Used in UML

The steps used in UML are:

- Define the use case model.
- Continue UML diagramming to model the system. during the systems analysis phase.
- Develop the class diagrams.
- Draw statechart diagrams.
- Begin systems design by refining the UML diagrams.
- Document your system design in detail.