# Faculty of Computing

**SE-314: Software Construction**

**Class: BESE 13AB**

# Lab 01: Introduction to UML – Use Case Diagrams and Class Diagrams

# CLO-02: Apply patterns, frameworks, and techniques for UML Diagrams CLO-04: Use modern tools such as Star UML, IBM Rational Rose etc. for UML Diagramss.

**Student Name:** Muhammad Saad Umer

**Class:** BESE 13-A

**CMS:** 408485

# Instructor: Dr. Mehvish Rashid

# Lab Engineer: Mr. Aftab Farooq

**Lab 01: Introduction to UML – Use Case Diagrams and Class Diagrams**

### **Introduction:**

Students have learned the use of UML and the theoretical knowledge of Use Case and Class diagrams in lectures. In this lab, students will learn how to use the UML notation to design use case diagrams and class diagrams.

### **Lab Objectives:**

After the completion of this lab, students will be able to draw use case diagrams and class diagrams according to the UML notation.

### Helping Material:

Please consult lectures slides and *Supporting Material* available on LMS.

## Tools/Software Requirement:

StarUML Download Link:  <https://staruml.io/download/>

StarUML Documentation: <https://docs.staruml.io/working-with-uml-diagrams/class-diagram>

**What is UML?**Unified Modeling Language (UML) is a collection of graphical notations, which are defined using a single meta-model. UML can be used for describing and designing software systems graphically, both at the requirements and design phases of a software life cycle. The focus is put on an object-oriented style, even if it is not limited to this paradigm.

**Use case Diagrams:**

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

The following topics describe model elements in use-case diagrams:

* [Use cases](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/cuc.html)  
  A use case describes a function that a system performs to achieve the user’s goal. A use case must yield an observable result that is of value to the user of the system.
* [Actors](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/cactor.html)  
  An actor represents a role of a user that interacts with the system that you are modelling. The user can be a human user, an organization, a machine, or another external system.
* [Subsystems](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/csubsys.html)  
  In UML models, subsystems are a type of stereotyped component that represent independent, behavioural units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modelling.
* [Relationships in use-case diagrams](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/crelsme_ucd.html)  
  In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behaviour between the model elements.

**Example:**

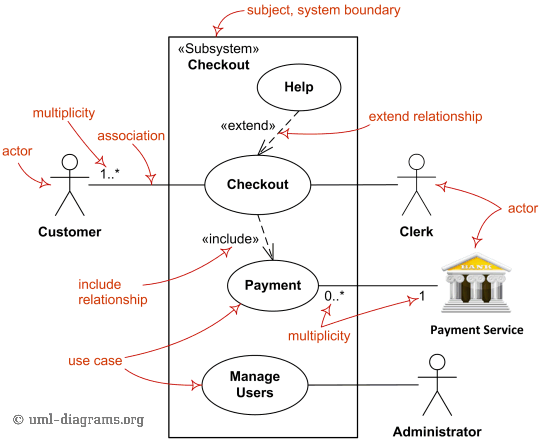


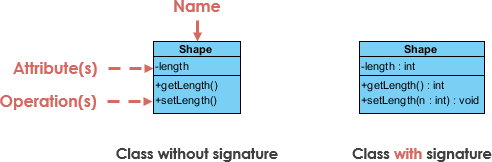
Figure 1- Major elements of UML use case diagram - actor, use case, subject, [**include**](https://www.uml-diagrams.org/use-case-include.html) and [**extend**](https://www.uml-diagrams.org/use-case-extend.html) relationships.

**Class Diagrams**

Class diagram is UML structure diagram which shows structure of the designed system at the level of classes and interfaces, shows their features, constraints, and relationships - associations, generalizations, dependencies, etc.

**UML Class Notation**

A class represent a concept which encapsulates state (**attributes**) and behaviour (**operations**). Each attribute has a type. Each **operation** has a **signature**. *The class name is the****only mandatory information***.



**Class Name:**

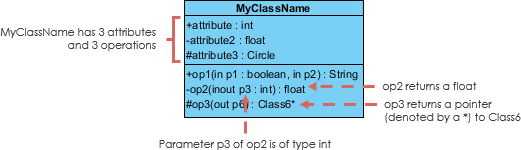
* The name of the class appears in the first partition.

**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.

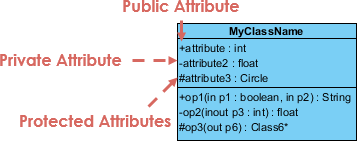
**Class Operations (Methods):**

* 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 are shown after the colon following the parameter name. Operations map onto class methods in code



**Class Visibility :**

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

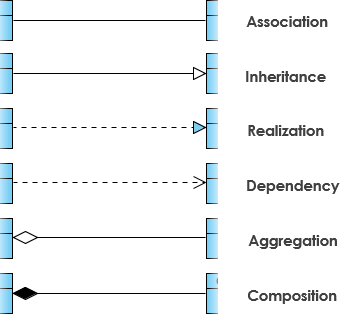


* **+** denotes public attributes or operations
* **-** denotes private attributes or operations
* **#** denotes protected attributes or operations

**Relationships between classes**

UML is not just about pretty pictures. If used correctly, UML precisely conveys how code should be implemented from diagrams. If precisely interpreted, the implemented code will correctly reflect the intent of the designer. Can you describe what each of the relationships mean relative to your target programming language shown in the Figure below?

If you can't yet recognize them, no problem this section is meant to help you to understand UML class relationships. A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

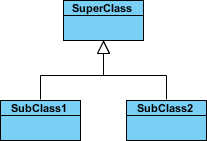


**Inheritance (or Generalization):**

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier.

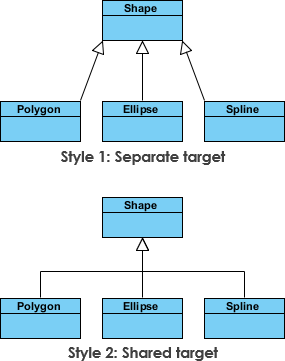
* Represents an "is-a" relationship.
* An abstract class name is shown in italics.
* SubClass1 and SubClass2 are specializations of SuperClass.

The figure below shows an example of inheritance hierarchy. SubClass1 and SubClass2 are derived from SuperClass. The relationship is displayed as a solid line with a hollow arrowhead that points from the child element to the parent element.



Inheritance Example - Shapes

The figure below shows an inheritance example with two styles. Although the connectors are drawn differently, they are semantically equivalent.



**Association**

Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes. Associations are typically named using a verb or verb phrase which reflects the real-world problem domain.

**Simple Association**

* A structural link between two peer classes.
* There is an association between Class1 and Class2

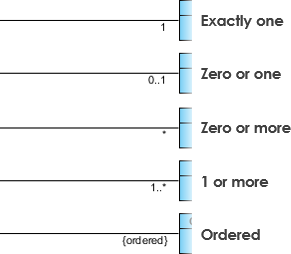
The figure below shows an example of simple association. There is an association that connects the <<control>> class Class1 and <<boundary>> class Class2. The relationship is displayed as a solid line connecting the two classes.

Simple Association

**Cardinality :**

Cardinality is expressed in terms of:

* one to one
* one to many
* many to many



**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.

The figure below shows an example of aggregation. The relationship is displayed as a solid line with a unfilled diamond at the association end, which is connected to the class that represents the aggregate.

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.

The figure below shows an example of composition. The relationship is displayed as a solid line with a filled diamond at the association end, which is connected to the class that represents the whole or composite.

Composition

**Dependency :**

An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modeled as a dependency relationship.

* A special type of association.
* Exists between two classes if changes to the definition of one may cause changes to the other (but not the other way around).
* Class1 depends on Class2

The figure below shows an example of dependency. The relationship is displayed as a dashed line with an open arrow.

Dependency

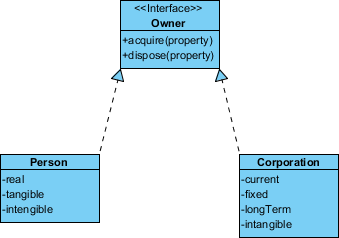
The figure below shows another example of dependency. The Person class might have a hasRead method with a Book parameter that returns true if the person has read the book (perhaps by checking some database).

Dependency

**Realization :**

Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class. In other words, you can understand this as the relationship between the interface and the implementing class.

For example, the Owner interface might specify methods for acquiring property and disposing of property. The Person and Corporation classes need to implement these methods, possibly in very different ways.



### **Lab Tasks:**

Draw the **Use case diagrams** and **Class Diagrams** for the following scenarios.

**Important note:** You may get help from the internet, however in case of plagiarism and copy paste of the images, no marks would be given.

### **Task 1: Smart Restaurant Electronic Kiosks for Ordering food**

* 1. Draw the **Use case Diagram** for an Electronic kiosks of a smart restaurant, you may give a brief description of the actor and use cases involved in the diagram. You may also name your restaurant and give a simple menu.
  2. Draw the **Class Diagram** for the ordering system of the restaurant using the electronic kiosks. Also give a paragraph describing the system.

**This news may help**: <https://koreajoongangdaily.joins.com/2021/02/23/business/industry/kiosk/20210223182200375.html>

*The domestic market for electronic kiosks has taken off during the coronavirus pandemic, as restaurant and store owners look for contactless alternatives to keep businesses running.*

*Commonly seen at small restaurants in Korea, electronic kiosks receive orders and take payments, printing out receipts with an order number that customers can later use to pick up their food. For the owner, the system saves labor costs, while customers save time standing in queues to place orders.*

|  |
| --- |
| Solution |
| (A)    Restaurant: CreekFoodPoint  Menu:    (B)    System Description:  The Smart Restaurant Electronic Kiosks for ordering food system is designed to streamline the food ordering process:   1. Item: With attributes like itemID, name, price, and description. 2. MenuItem: Inherits from Item categories such as Desserts, Beverages, Main Courses, and Appetizers. 3. Order: Manages customer orderID, orderDate, and status, and methods to add or remove MenuItems. 4. Payment: Handles the financial transactions with details like paymentID, amount, and paymentMethod. 5. Customer: Tracks customer details, including customerID, name, and orderHistory.   The system ensures that customers can efficiently place orders. |

### **Task 2: Ticket vending machine of a Subway/Train System**

Purpose: Show that ticket vending machine allows commuters to buy different kind of tickets.

Summary: The ultimate goal of a Commuter in relation to our ticket vending machine is to buy a ticket. The ticket may be of different types. (It can be a short distance (within city) ticket or can be a long-distance ticket (Out of City)). Ticket vending machine is a subject of the example use case diagram. Commuter and Bank are our actors, both participating in the Purchase Ticket use case. It is encouraged that you incorporate your own ideas.

In this task, you may target a specific country and research on how ticketing systems differ and incorporate those ideas in your use case diagram.

|  |
| --- |
| Solution |
| (A)    (B) |

### **Deliverables:**

Compile a single word document by filling in the solution part and submit this Word file on LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).