Department of
Computer Science and Engineering

Lab No. 10
Title: Develop UML Class Diagram (part 1: basic class diagram) for the given project

Integrated Design Project I Lab
CSE 324

Green University of Bangladesh

1 Objective(s)

- To describes the structure of an object-oriented system by showing the classes in that system and the relationships between the classes.

1.1 Sub-Objective(s)

- To describe the static view of the system.
- To model the collaboration among the elements of the static view.
- To describe the functionalities performed by the system.
- To construct software applications using object oriented languages.
- To construct software applications using object oriented languages.

2 UML Class Diagram

2.1 What are UML diagrams?

UML, which stands for Unified Modeling Language, is a way to visually represent the architecture, design, and implementation of complex software systems. When you're writing code, there are thousands of lines in an application, and it's difficult to keep track of the relationships and hierarchies within a software system. UML diagrams divide that software system into components and subcomponents.

2.2 What is Class Diagram?

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. A Class Diagram in Software engineering is a static structure that gives an overview of a software system by displaying classes, attributes, operations, and their relationships between each other. Class Diagram helps construct the code for the software application development. Class Diagram defines the types of objects in the system and the different types of relationships that exist among them. It gives a high-level view of an application. This modeling method can run with almost all Object-Oriented Methods. A class can refer to another class. A class can have its objects or may inherit from other classes.

2.3 Purpose of Class Diagrams

Most of the UML diagrams can't be mapped directly with any object-oriented programming languages except class diagrams. In other words, class diagram ideally can have one to one mapping to UML class diagrams. Class diagrams are the main building blocks of every object-oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has an appropriate structure to represent the classes in their context. It describes various kinds of objects and the static relationship between them.

3 Methodology

3.1 Essential elements of A UML class diagram

There are several diagram components that can be efficiently used while making/editing the model. These are as follows:

Class:

- Class name
- Attribute
- Operation

Class:

The UML representation of a class is a rectangle containing three compartments Name, Attributes and Operations stacked vertically, as shown in the Fig.1.

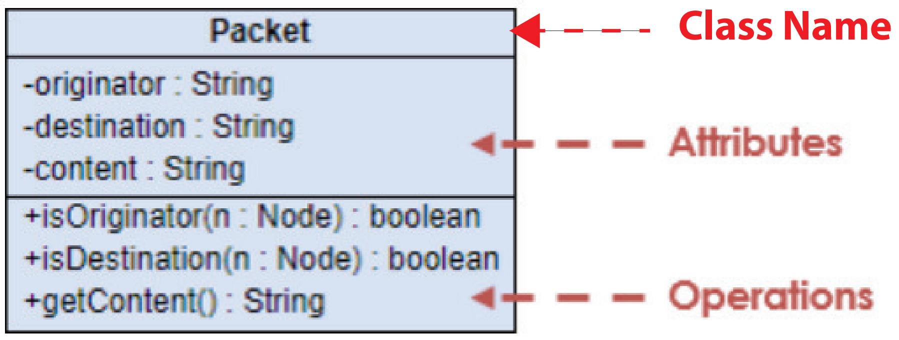

Figure 1: Symbols of a class

- **(i) Name:**

  Each class is represented by a rectangle in the diagram, with the name of the class in the top part. Class names are usually centered, written in bold, and start with a capital letter.

- **(ii) Attribute:**

  Attributes are data elements that are contained in every object of a class and have a value for each of these objects. Attributes are shown in the middle of the class rectangle, are aligned left, and the first letter is lowercase. The attribute section of a class lists each of the class's attributes on a separate line. The attribute section is optional, but when used it contains each attribute of the class displayed in a list format. The line uses this format: name : attribute type (e.g. cardNumber : Integer).

  Visibility:

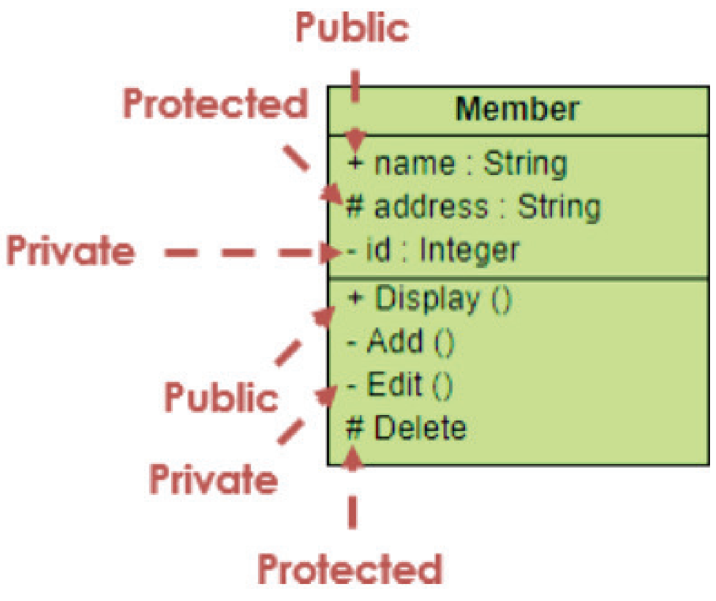

Figure 2: Symbols of Visibility

Visibility is used to signify who can access the information contained within a class denoted with +, -, and as shown in the Fig.2.

- **(iii) Operation:**

  The operations are documented in the bottom compartment of the class diagram's rectangle, which also is optional. Like the attributes, the operations of a class are displayed in a list format, with each operation on its own line. Operations are documented using this notation: name (parameter list) : type of value returned (e.g. calculateTax (Country, State) : Currency).

3.2 Relationships between Classes

There are mainly three kinds of relationships in UML:

- Dependencies
- Generalizations
- Associations

There are different types of relationships in the class diagram are described below.

(i) Dependency A dependency means the relation between two or more classes in which a change in one may force changes in the other. However, it will always create a weaker relationship. Dependency indicates that one class depends on another.

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

-Class B is used as the type of a local variable in one or more methods of class A.

-Class B is used as the type of parameter for one or more methods of class A.

-Class B is used as the return type for one or more methods of class A.

-One or more methods of class A invoke one or more methods of class B.

In the following UML class diagram examples, Student has a dependency on College as shown in Fig.3

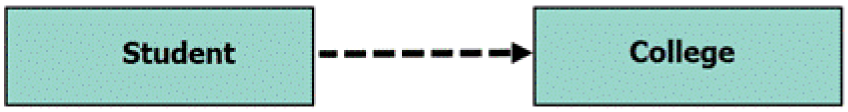

Figure 3: Symbols of Dependency

(ii) Generalization

inheritance relationship (Inheritance between classes, interface implementation) Will discuss in the 11th lab.

(iii) Association

objects are made up of other objects. Association specifies a "has-a" or "whole/part" relationship between two classes. In an association relationship, an object of the whole class has objects of part class as instance data.

a class diagram, an association relationship is rendered as a directed solid line. There are two types of associations, Unidirectional and Bidirectional association as shown in Fig.4

Unidirectional association - In a unidirectional association, two classes are related, but only one class knows that the relationship exists. A unidirectional association is drawn as a solid line with an open arrowhead pointing to the known class.

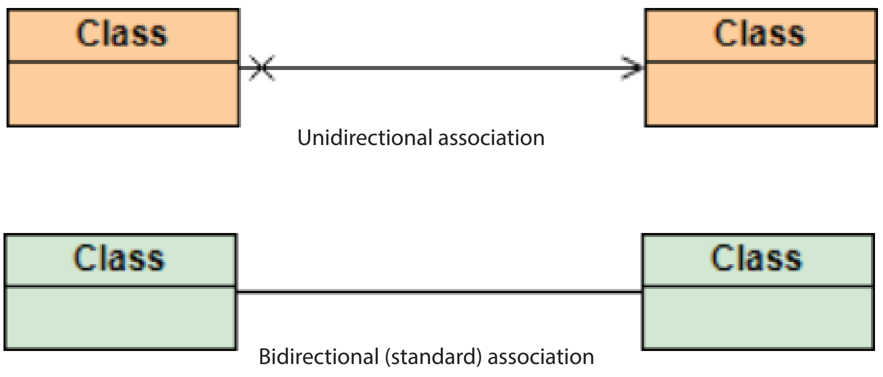
Class
Class
Unidirectional association
Class
Class
Bidirectional (standard) association

Figure 4: Symbols of Association

Bidirectional (standard) association - The default relationship between two classes. Both classes are aware of each other and their relationship with the other. This association is represented by a straight line between two classes. A bi-directional association is indicated by a solid line between the two classes.

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 as shown in Fig.5.

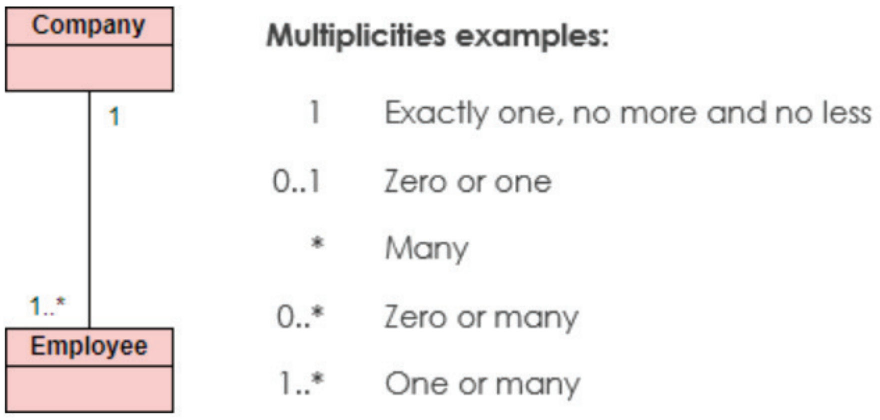

Figure 5: Symbols of Multiplicity

Aggregation

Aggregation is a special type of association that models a whole- part relationship between aggregate and its parts as shown in Fig.6. The relationship is displayed as a solid line with an unfilled diamond at the association end, which is connected to the class that represents the aggregate.

For example, the class college is made up of one or more student. In aggregation, the contained classes are never totally dependent on the lifecycle of the container. Here, the college class will remain even if the student is not available shown in figure 6.

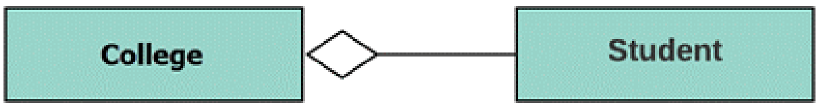

Figure 6: Symbols of Aggregation

Aggregation indicates a relationship where the child can exist separately from their parent class. Example: Automobile (Parent) and Car (Child). So, if you delete the Automobile, the child Car still exist.

Composition

The composition is a special type of aggregation which denotes strong ownership between two classes when one class is a part of another class as shown in Fig.7.

For example, if college is composed of classes student. The college could contain many students, while each student belongs to only one college. So, if college is not functioning all the students also removed.

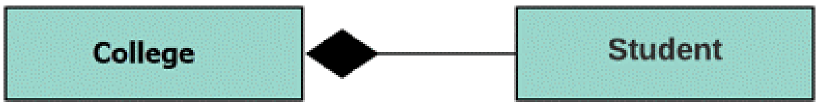

Figure 7: Symbols of Composition

Composition display relationship where the child will never exist independent of the parent. Example: House (parent) and Room (child). Rooms will never separate into a House.

3.3 Required Software

To design your Class diagram, you may use platforms and editing tools online. These tools are helpful since they already have the needed symbols to illustrate your class diagram. You just have to plot the included classes, relationships, and visibility. Then you will put the appropriate attributes that the system requires.

platforms or online tools that you may use are:

1. Visual Paradigm for UML 8.2 (online link: https://online.visual-paradigm.com/)
2. StartUML
3. Lucidchart and other drawing tools

3.4 Design Procedure

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

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

1. Identify the objects in the problem domain, and create classes for each of them. (e.g. Teacher, Student, Course for an enrollment system)
2. Add attributes for those classes (e.g. name, address, telephone for the Student class)
3. Add operations for those classes (e.g. addStudent(student) for the Course class)
4. Connect the classes with appropriate relationships (e.g. Relate Teacher and Course with an association)
5. Optionally specify the multiplicities for association connectors' ends (e.g. Input 0..3 for the Course side of the association that connects Teacher and Course, to signify that one teacher can teach multiple up to three courses)

4 Implementation

You can draw class diagrams in VP-UML. A class diagram describes the structure of an object-oriented system by showing the classes in that system and the relationships between the classes. A class diagram also shows constraints, and attributes of classes. Go through the following link and follow procedure step by step to draw a class diagram. https://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190drawingclass.html

The Class diagram of the Telephone communication system is designed using vp-UML, shown in fig.8

5 Output

The Class diagram of the Telephone communication system

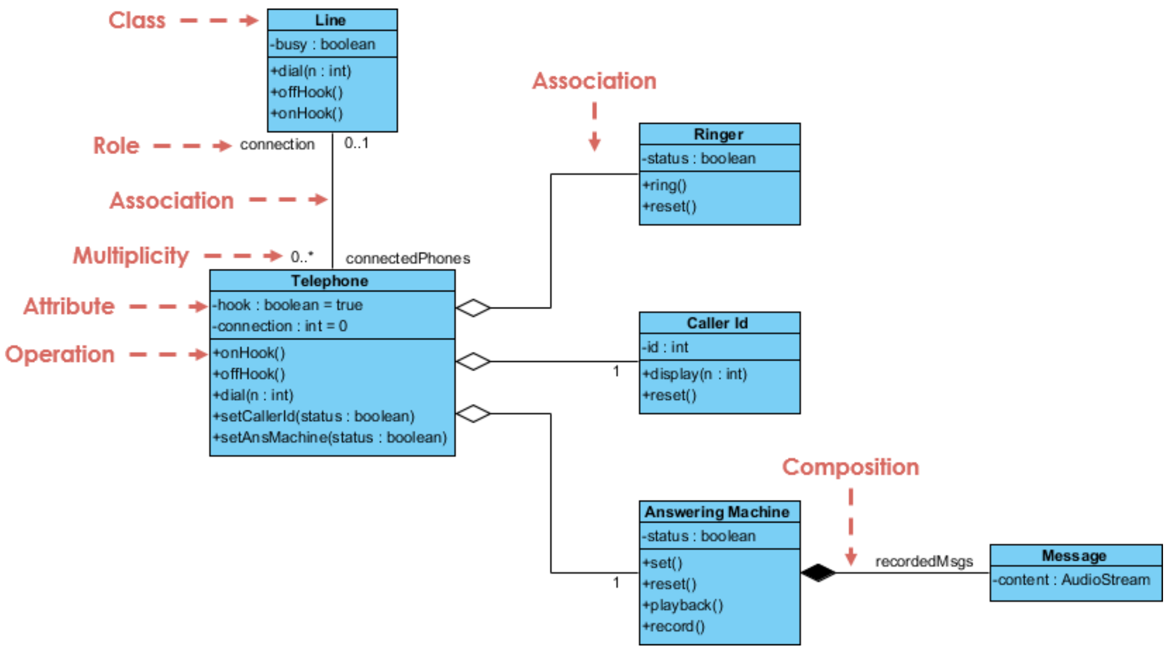

Figure 8: Class diagram of the Telephone communication system

The telephone communication System is a modeled diagram that explain its classes and relationships. In the diagram classes are represented as boxes with three rectangles inside each box. The top rectangle has the class's name; the middle rectangle contains the class's properties; and the bottom rectangle contains the class's methods, commonly known as operations.

As you can see through the illustration, the classes were determined which is symbolized by boxes. They were designated with their corresponding attributes and shows the class' methods. Their relationships are also plotted to show the connections between classes and their multiplicity. You should also look into the visibility symbols displayed in the diagram. These are important because it declares the status of attributes in your Class Diagram. Some of the Class' attributes are for public (+) which means that they can be accessed by the classes connected to them. While the protected () symbols, means that the attributes of the data can be accessed by the same classes or subclass and the (-) symbol means it cannot be accessed by other class.

6 Discussion & Conclusion

To design and visualize the software system artifacts, the standard language used is the UML. The relationship between the different objects is described by the class diagram, which ensures the design and analysis of an application and views it in its static form. Being the most important UML diagram, the class diagram consists of class, attributes, and relationships, which are its essential elements. To get an idea of the application structure, the class diagram is used, which helps in reducing the maintenance time.

7 Lab Task (Please implement yourself and show the output to the instructor)

1. Draw a class diagram for the given project **'Student Registration System'**.

7.1 Problem analysis

A Student Registration System (SRS) is a software application for educational establishments to manage student data. Student Registration systems provide capabilities for entering registered student list, enrolled courses, payment status and managing many other student related data needs in a school, college or university.

Develop a Student Registration System in which student gets enroll in a semester. The semester contains courses. Each course has a title and course Code. The course can be registered, dropped, withdraw, passed and failed by the students. Also, a semester may be freezed or attended. There are two kinds of students; Undergraduate Students (for BCS MCS) and Postgraduate Students (for MPhil PhD). Each type of students enrolls in different ways.

The classes that must include in an Student Course Registration System would be the students, courses, registration, payment, requirements, and transaction. The mentioned classes were just general. If you want more complex or wider scope of your Student Registration system, then you can add your desired classes.

8 Lab Exercise (Submit as a report)

- Draw an Class diagram for the given project for your team.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.