LAB 11: Composition and Aggregation

Aggregation:

CONCEPT: Aggregation occurs when a class contains an instance of another class.

Q1:When designing software, it sometimes makes sense to create an object from other objects. For example, Suppose you need an object to represent a **Course** that you are taking in college. You decide to create a **Course** class, which will hold the following information:

• The course name
• The instructor's last name, first name, and office number
• The textbook's title, author, and publisher

In addition to the course name, the class will hold items related to the **instructor** and the **textbook**. You could put attributes for each of these items in the Course class. However, a good design principle is to separate related items into their own classes.

Note:

In this question, an Instructor class could be created to hold the instructor-related data and a TextBook class could be created to hold the textbook-related data. Instances of these classes could then be used as attributes in the Course class.

To keep things simple, the **Instructor class** will have only the following functions:

• A default constructor that assigns empty strings to the instructor's last name, first name, and office number.
• A constructor that accepts arguments for the instructor's last name, first name, and office number
• A set function that can be used to set all of the class's attributes
• A print function that displays the object's attribute values

the **TextBook** class will have only functions are:

a default constructor,
a constructor that accepts arguments,
a set function, and a print function.

the Course class has an Instructor object and a TextBook object as member variables. Those objects are used as attributes of the Course object. Making an instance of one class an attribute of another class is called object aggregation.

the relationships that exist among the Course , Instructor , and TextBook classes can be described as follows:

• The course has an instructor.
• The course has a textbook.

Test the functionality of all classes using main by Creating a Course object and passing it the course name ,instructor information and textbook title and author,publisher and finally prints the course information.

Composition is again specialize form ofAggregation. It is a strong type of Aggregation. Here the Parent and Child objects have coincident lifetimes. Child object dose not have it's own lifecycle and if parent object gets deleted, then all of it's child objects will also be deleted.

Q2:A university owns various departments (e.g., CS, Electrical Engineering), and each department has number of professors. If the university closes, the departments will no longer exist, but the professors in those departments will continue to exist. Therefore, a University can be seen as a composition of departments, whereas departments have an aggregation of professors. In addition, a Professor could work in more than one department, but a department could not be part of more than one university.

Write a Class named **Professor** having following attributes:

- name of type string

- employeeID of type int
- designation of type string

Write a Class named **Department** having following attributes:

- name of type string
- deptID of type int
- Array profList of type Professor
- noOfProfessors of type int

Write a Class named **University** having following attributes:

- name of type string
- Array dept of type Department
- numberOfDepartments of type int

Write following functions.

1. Write appropriate getter setter of each data member for each Class.
2. Add/delete/update Department in University class

bool addDepartment(Department D)
bool deleteDepartment(string name)
bool updateDepartment(int id, string name) //Update name of department given department id.

void Display() function to display university information. Also display department information in this function.

3. Add/delete/update Professor in Department class

bool addProfessor(Professor p)
bool deleteProfessor (int id)
bool updateProfessor (int id, string newDesignation) //Update designation of the professor given employee id

void Display() function to display department information. Also display professors information in this function.

For Q2 Test the functionality using testcases.