**OBJECT-ORIENTED PROGRAMMING**

|  |  |
| --- | --- |
| Lab 10 | |
| **Topic** | Composition and composition |
| **Objective** | Instil skills in modeling relationships, and fostering modular and flexible code design. |

**Solve the following tasks and identify whether they have a composition or aggregation relation.**

**Task 1: Library System with Book and Author Classes**

Create two classes: **Book** and **Author**. The **Book** class should have the following features:

* Private Member Variables:
  + **title** (string): to store the book title.
  + **price** (double): to store the book price.
  + **author** (Author\*): a pointer to the **Author** class representing the book's author.
* Public Member Functions:
  + Default constructor: Initializes the book with default values and a null author pointer.
  + Parameterized constructor: Accepts parameters for the book title, price, and author and initializes the book.
  + **display()**: Displays the book information, including the author's details.

The **Author** class should have the following features:

* Private Member Variables:
  + **name** (string): to store the author's name.
  + **email** (string): to store the author's email address.
  + **birthYear** (int): to store the author's birth year.
* Public Member Functions:
  + Default constructor: Initializes the author with default values.
  + Parameterized constructor: Accepts parameters for the author's name, email, and birth year and initializes the author.
  + **display()**: Displays the author's information.

**Task 2:Employee and Department Classes**

Create two classes: **Employee** and **Department**. The **Employee** class should have the following features:

* Private Member Variables:
  + **id** (int): to store the employee ID.
  + **name** (string): to store the employee name.
  + **salary** (double): to store the employee salary.
* Public Member Functions:
  + Default constructor: Initializes the employee with default values.
  + Parameterized constructor: Accepts parameters for the employee ID, name, and salary and initializes the employee.
  + **display()**: Displays the employee information.

The **Department** class should have the following features:

* Private Member Variables:
  + **name** (string): to store the department name.
  + **Manager** (Employee): an object of the **Employee** class representing the department manager.
* Public Member Functions:
  + Default constructor: Initializes the department with an empty name and a default manager.
  + Parameterized constructor: Accepts parameters for the department name and manager and initializes the department.
  + **display()**: Displays the department information, including the manager's details.

**Task 3: Company System with Employee and Department Classes**

Create two classes: **Employee** and **Department**. The **Employee** class should have the following features:

* Private Member Variables:
  + **id** (int): to store the employee ID.
  + **name** (string): to store the employee name.
  + **salary** (double): to store the employee salary.
* Public Member Functions:
  + Default constructor: Initializes the employee with default values.
  + Parameterized constructor: Accepts parameters for the employee ID, name, and salary and initializes the employee.
  + **display()**: Displays the employee information.

The **Department** class should have the following features:

* Private Member Variables:
  + **name** (string): to store the department name.
  + **employees** (vector<Employee\*>): a vector of pointers to **Employee** objects representing the department's employees.
* Public Member Functions:
  + Default constructor: Initializes the department with an empty name and no employees.
  + Parameterized constructor: Accepts parameters for the department name and initializes the department.
  + **addEmployee(Employee\* emp)**: Adds an employee to the department.
  + **display()**: Displays the department information, including the employees' details.

**Task 4: Computer and Processor Classes**

Create two classes: **Processor** and **Computer**. The **Processor** class should have the following features:

* Private Member Variables:
  + **brand** (string): to store the processor brand.
  + **speed** (double): to store the processor speed in GHz.
* Public Member Functions:
  + Default constructor: Initializes the processor with default values.
  + Parameterized constructor: Accepts parameters for the processor brand and speed and initializes the processor.
  + **display()**: Displays the processor information.

The **Computer** class should have the following features:

* Private Member Variables:
  + **model** (string): to store the computer model.
  + **processor** (Processor): an object of the **Processor** class representing the computer's processor.
* Public Member Functions:
  + Default constructor: Initializes the computer with an empty model and a default processor.
  + Parameterized constructor: Accepts parameters for the computer model and processor and initializes the computer.
  + **display()**: Displays the computer information, including the processor's details.

**Task 5: University System with Student and Department Classes**

Create two classes: **Student** and **Department**. The **Student** class should have the following features:

* Private Member Variables:
  + **id** (int): to store the student ID.
  + **name** (string): to store the student name.
  + **department** (Department\*): a pointer to the **Department** class representing the student's department.
* Public Member Functions:
  + Default constructor: Initializes the student with default values and a null department pointer.
  + Parameterized constructor: Accepts parameters for the student ID, name, and department and initializes the student.
  + **display()**: Displays the student information, including the department's details.

The **Department** class should have the following features:

* Private Member Variables:
  + **name** (string): to store the department name.
* Public Member Functions:
  + Default constructor: Initializes the department with an empty name.
  + Parameterized constructor: Accepts parameters for the department name and initializes the department.
  + **display()**: Displays the department information.

**Task 6: School System with Student, Teacher, and School Classes**

Create three classes: **Student**, **Teacher**, and **School**. The **Student** and **Teacher** classes should have similar features:

* Private Member Variables:
  + **id** (int): to store the student or teacher ID.
  + **name** (string): to store the student or teacher name.
  + Additional member variables as needed.
* Public Member Functions:
  + Default constructor: Initializes the student or teacher with default values.
  + Parameterized constructor: Accepts parameters for the ID, name, and additional details and initializes the student or teacher.
  + **display()**: Displays the student or teacher information.

The **School** class should have the following features:

* Private Member Variables:
  + **name** (string): to store the school name.
  + **students** (Student\*): an array of pointers to **Student** objects representing the school's students.
  + **teachers** (Teacher\*): an array of pointers to **Teacher** objects representing the school's teachers.
* Public Member Functions:
  + Default constructor: Initializes the school with an empty name and no students or teachers.
  + Parameterized constructor: Accepts parameters for the school name and initializes the school.
  + **addStudent(Student\* stu)**: Adds a student to the school.
  + **addTeacher(Teacher\* tea)**: Adds a teacher to the school.
  + **display()**: Displays the school information, including the students' and teachers' details.