Object Oriented Programming
Assignment 2

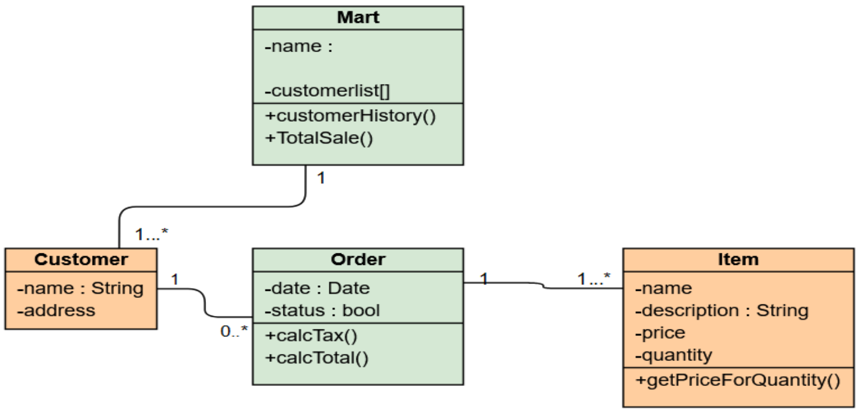
Topic: Association, Inheritance and Polymorphisim

Instructor: Muhammad Aizaz Akmal
Deadline: 20th April 2025

Note: In case of plagiarism, you will get zero marks

Problem 1: [CLO: 4]

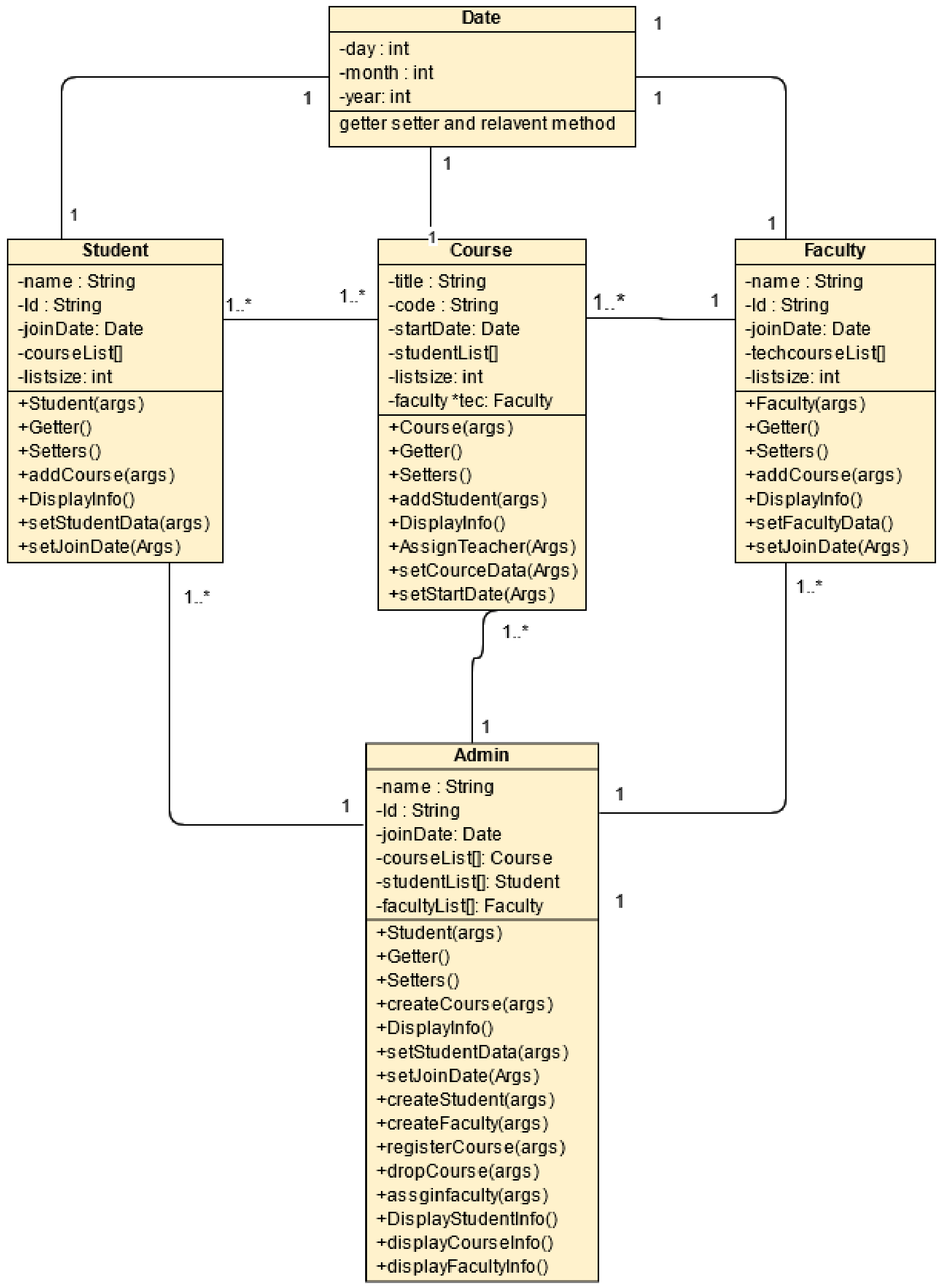
Implement the given UML diagram of customer order details.



UML Diagram of Customer Order

Problem 2: [CLO: 4]

Design a mini–Student Course Registration Portal. The purpose of this portal is tracking the record of the courses in which a student is enrolled and also the detailed information of course that includes the list of students enrolled in a particular course along with teacher information about who is teaching it. The registration of course, students and teacher, is done by an admin officer. The UML Diagram is given below. You may include an additional method or attribute according to your need. Create a menu driven program for all the operations.



UML Diagram of mini student course registration portal

Problem 1: Publishing Company **[CLO: 6]**

a) Imagine a publishing company that markets both book and audiocassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data.

Write a main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata()

b) Start with the **publication**, **book**, and **tape** classes. Add a base class **sale** that holds an array of three **float**s so that it can record the dollar sales of a particular publication for the last three months. Include a **getdata()** function to get three sales amounts from the user, and a **putdata()** function to display the sales figures. Alter the **book** and **tape** classes so they are derived from both **publication** and **sales**. An object of class **book** or **tape** should input and output sales data along with its other data. Write a **main()** function to create a **book** object and a **tape** object and exercise their input/output capabilities.

c) Assume that the publisher in Exercises part a and b decides to add a third way to distribute books: on computer disk, for those who like to do their reading on their laptop. Add a **disk** class that, like **book** and **tape**, is derived from **publication**. The **disk** class should incorporate the same member functions as the other classes. The data item unique to this class is the disk type: either CD or DVD. The user could select the appropriate type by typing **c** or **d**.

Problem 4: Package Inheritance Hierarchy [CLO: 6]

Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping option, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages.

Use Package as the base (abstract) class of the hierarchy, then include classes TwoDayPackage and OvernightPackage that derive from Package. Base class Package should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package.

Package's constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. Package should provide a public member function calculateCost that returns a double indicating the cost associated with shipping the package. Package's calculateCost function should determine the cost by multiplying the weight by the cost per ounce.

Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function calculateCost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class Package's calculateCost function.

Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service.

OvernightPackage should redefine member function calculateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of Package and tests member function calculateCost.

Create a menu driven program in which multiple packages can be placed according to customer requirement either through TwoDayPackage or OverNightPackage.