Exercise on TypeScript OOP

1. Write a TypeScript program

    a. Create a class `BookList`
    b. Create another class called `Book`
    c. **BookLists** should have the following properties:
        i. Number of books marked as read
        ii. Number of books marked not read yet
        iii. A reference to the next book to read (book object)
        iv. A reference to the current book being read (book object)
        v. A reference to the last book read (book object)
        vi. An array of all the Books
    d. Each **Book** should have several properties:
        i. Title
        ii. Genre
        iii. Author
        iv. Read (true or false)
        v. Read date, can be blank, otherwise needs to be a JS Date() object
    e. Every **Booklist** should have a few methods:
        i. .add(book)
            1. should add a **book** to the books list.
        ii. .finishCurrentBook()
            1. should mark the **book** that is currently being read as "read"
            2. Give it a read date of new Date(Date.now())
            3. Change the last **book** read to be the book that just got finished
            4. Change the current **book** to be the next book to be read
            5. Change the next **book** to be read property to be the first unread book you find in the list of books
    f. **Booklists** and **Books** might need more methods than that. Try to think of more that might be useful.

2. Write a TypeScript Snippet for Manager class Extending Employee.

    a. Containing empNo,name,salary and bonus Attributes [Employee]
    b. Define Parameterized Constructor and initialize attributes.
    c. Add getters and setters for all the Attributes.
    d. Override toString() with new syntax of Data Bindings.

    e. Containing noOfReportees Attributes [Manager].
    f. Call Base Class Constructor, initialize the Properties of Base and Additionally initialize Attributes of Manager
    g. Add getters and setters for all the Attributes.
    h. Override toString() with new syntax of Data Bindings.

3. Create a program that calculates the return date of a rental item based on the day it was rented and on the total number of days before it must be returned. You must determine how many days are in the month and whether the year is a leap year.
4. Create a class to construct an array of Customer objects, and then provide a method to find and return a Customer object for a given ID. The Order class needs to be modified to contain an List of order items, requiring a method to add items into the List, and another method to remove the items.
5. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.

Exercise on TypeScript Interfaces Enums and Generics

1. Write a TypeScript program on Enums.
   a. Create a public enum Weekday with constants for MONDAY, TUESDAY,... until SUNDAY.
   b. The enum should have an instance method boolean isWeekDay() and an instance method boolean isHoliday().
   c. The isHoliday() method should return the opposite of isWeekDay().
   d. Write a program which demonstrates how this enum could be used, which has a method which takes a Weekday as the argument and prints a message depending on whether the Weekday is a holiday or not.
   e. method loops over all values in the Weekday enum and sends them as argument to the method.

2. Write a TypeScript Snippet to Design a class that acts as a library for the following kinds of media: book, video, and newspaper. Provide one version of the class that uses generics and one that does not. Feel free to use any additional APIs for storing and retrieving the media.

3. Write a TypeScript function to
    a. Create an interface called Agreeable that declares how to compare two objects. The interface has an abstract method called isSmallerThan(Agreeable other). Then, create a class called Shape that implements the Agreeable interface. Shape has two fields: int width, int height. The class has two constructors, one with empty arguments and one with two arguments, width and height.
    b. The class Shape has two methods. The first method is called calcArea and returns the area i.e. (int) width * height. The second method implements the Agreeable interface. After create two new objects of type shape and compare them between them.

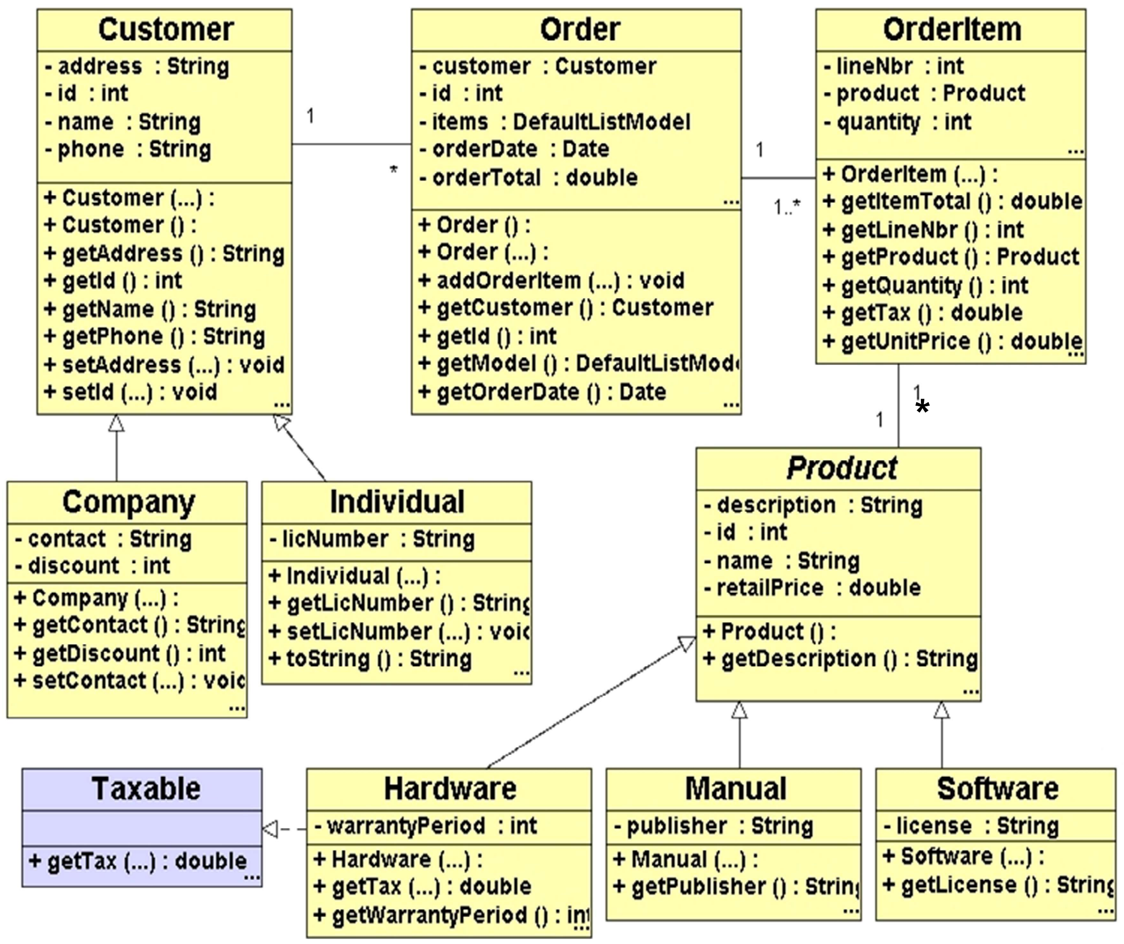
4. Write a TypeScript Snippet
    a. Provide an interface Measurable with a method double getMeasure() that measures an object in some way. Make Employee implement Measurable. Provide a method double average(Measurable[] objects) that computes the average measure. Use it to compute the average salary of an array of employees.
    b. Continue with the preceding exercise and provide a method Measurable largest(Measurable[] objects). Use it to find the name of the employee with the largest salary. Why do you need a cast?

Exercise on TypeScript Modules Compiler Argurments And Model Classes and Build API

1. Write a TypeScript Snippet to Demonstrate Compiler Arguments.
2. Write a TypeScript program
    a. Create a Module for Employee Class and Export it
    b. Using App Entry Class Import the Module and use the Functionality in it.

3. Write a TypeScript Snippet to
    a. Demonstrate Model Classes and Building API for the Following diagram
    b. Modularize the Application

UML Diagram for the Order Entry Application



How to Read the Diagram

The italicized class *Product* is an abstract class, and the lavender-colored class Taxable is an interface. The solid line with an arrow is a generalization and is used for all dependencies between subtypes and supertypes; it generates an `extends` statement in the class.

A generalization is the relationship between a more specific element and a less specific element, and defines the inheritance structure in the model. Generalization relationships can be created between two TypeScript classes, between two TypeScript interfaces, or between two entity objects.

The dotted line with the arrow represents a realization and is used between an interface and a class; it generates an `implements` statement in the class. A realization relationship identifies which TypeScript class (or classes) implements a TypeScript interface. The solid line with no arrow is an association, which signifies a call or a reference from one class to another. The type of relationship between `OrderItem` and `Order` is composition.

The plus symbol (+) next to attributes means that they are public. The "-" symbol means that they are private.

Description of the Order Entry Area of the Business

The Order Entry component of the business needs to be automated. In most respects, the process of ordering products is rather simple. The customers select the items that they want from a list of products. Customers must be included in your system. You keep information about customers such as name, address, and telephone number. You also assign a unique customer ID to each new customer. For customers that are companies, you track a contact person and provide for a discount on company purchases. You identify individual customers by their icense number.

The order itself is not very complicated. Each order has a unique number that is used to keep track of it and has information such as the customer who placed the order, the order date, shipping mode (air or ground), and order status. Each order can have multiple line items. You currently limit your customers to 10 items per order. Each item on an order has the product being purchased, the price, quantity, and the product category. A product category can be a composite category, consisting of additional categories, or a leaf category. You track many aspects about your products, and the key aspects include the name, description, and list price. Additionally, you want to include a warranty period, the supplier who distributes the product, a catalog URL to reference it on the Web, and a weight classification that is used when you calculate shipping costs. It is also important for you to track information about which products are available and where they are located. A number of warehouses are used to store the products.