Homework 1 (Classes, Inheritance)

**Instructions**

Every assignment needs to provide two main artifacts – a complete working program, along with its full source code, and your solution documentation. The documentation should be in a single text file (e.g. doc), placed inside your Solution folder.

**For this assignment, the documentation will make 10% of the grade, and code will be 90%.**

In the documentation you need to describe the data types you chose for implementing the classes member variables, the return types and parameters for the public functions (including the constructors), and the reasons for choosing them.

The software fraction of the homework will be graded according to the following rules:

* existence of compile-time errors not more than 20 %
* existence of run-time errors not more than 50 %
* existence of logical-errors not more than 70 %
* correct program not less than 70 %

The last 30 points for the software part will be given for **program comments**, efficiency, class hierarchies, textual representation, structure, layout, etc.

**NOTE:** If the program is not correct, credits for these properties will not be given!

**NOTE:** An incomplete implementation of the required functionality (e.g. missing class property) will be considered a logical error.

**Deliverables**

A zipped top-level folder of the Visual Studio solution, containing all of the code, plus the document describing the program.

**NOTE:** You may be asked to demonstrate your solution in person.

**Deadline**

The complete homework (code and documentation) must be uploaded on Canvas no later than Sunday, 4th February, 2024, 23:59.

**NOTE: Late work will be accepted but penalized.**

**Task Description**

Implement a program in C++ to represent a Factory that sells its production to store suppliers. The factory has different **divisions** which produce different **artifacts**. Artifacts are sold for a certain price and can have a discount. The factory also offers **Services** to its clients – e.g. custom development.

For this first assignment, you need to develop the following **classes**:

1. **Division**

Data Slots:

* GUID (string) – *e.g. ABCD-EFGH-1234*
* Name (string) – *e.g. Electronics*
* PhoneNumber (string) – *e.g. 0888 123456*
* Description (long string)
* Parent (Division) – *a parent division, can be NULL*

Methods:

* Constructor(s)

Axioms:

* GUID must be unique across other divisions’ GUIDs

2. **Artifact**

Data Slots:

* GUID (string)
* Name (string) – *e.g. Sony Vaio*
* Description (long string)
* Category (string) – *the artifact category – e.g. Laptop*
* Division (Division) – *division to which the artifact belongs;* *cannot be empty*
* Price (double) – *e.g. 185.99*
* Discount (double) – *e.g. 30*
* DiscountType (enum) – *amount* or *percentage*
* Quantity (number)

Methods:

* Constructor(s)
* double GetEffectivePrice {
* If DiscountType is amount:
* Return Price – Discount
* Else:
* Return Price – Price \* Discount / 100
* }
* double GetTotalPrice { Return Quantity \* GetEffectivePrice() }

Axioms:

* Price, Discount and Quantity cannot be negative
* GetEffectivePrice() and GetTotalPrice() should not return a negative amount

3. **Service** (inherits **Artifact**)

Own Data Slots:

* Duration (double) – *e.g. 1.5 (service execution duration in hours)*
* Rate (double) – *e.g. 8.50*
* RateDiscount (double) – *e.g. 30*
* RateDiscountType – *amount* or *percentage*

Methods:

* Constructor(s)
* double GetEffectiveRate {
* If RateDiscountType is amount:
* Return Rate – RateDiscount
* Else:
* Return Rate – Rate \* RateDiscount / 100
* }
* double GetTotalPrice { Artifact::GetTotalPrice() + GetEffectiveRate() \* Duration }

Axioms:

* Duration, Rate and RateDiscount cannot be negative
* GetEffectiveRate() and GetTotalPrice() should not return negative amount

**Driver Program:**

1. Define three vectors for storing the data for the Market Store:
2. Divisions – pointers of object instances of class Divisions available for your store
3. Artifacts – pointers of instances of Artifact
4. Services – pointers of instances of Service
5. Add code to implement the following functionality in the rest of the driver program:
6. Add at least three new divisions to the Divisions collection
7. Add at least three artifacts to the Artifact list, which have a correct Division from the list of available and instantiated divisions in the previous step
8. Add at least three services to the Services list, having a valid Division
9. Use two loops to show the Name and Total Price of each artifact and service in the store

**NOTE:** The purpose of the driver program is to test the classes you have implemented. There is no need to create any additional functionality such as user menus, reading user input, or reading information from files. All of the driver program requirements can be implemented by adding statements and static data in the **main()** function of your program.