Object-Oriented Programming

Summer Semester 2023

Homework Assignment 4.

21/08/2023

Contents

Introduction

1. Try to provide clear and careful solutions.
2. You should provide comments for your code so it will be completely clear what you are trying to achieve. WARNING! Lack of comments might lead to points reduction.
3. Please note the following few points which may lead to points reduction during the submission check:
   (a) Avoid using *magic numbers*. For example: “if (i>17)”, 17 is a magic number. If 17 is representing, for example, the number of shoes, then instead you should write: “if (i>shoesNumber)”.
   (b) Try to avoid code duplication as much as possible.
   (c) You should not globally enable *std* namespace usage or any other namespaces, e.g. *using namespace std;*.

Learning from past experience, please note that some (small) updates to the definitions and requirement of the assigment might be published in next few days and that following said updates and the HW forum is required.Posting your questions in the HW forum helps your fellow students with similar questions.

Definitions

For the fourth homework, you will be challenged with the following (Relevant tutuorials are 9-12) :

1. Casting.
2. STL.
3. Exceptions.
4. Design Patterns
5. Multiple Inheritance

In this assignment you will implement requested functionality given to you from KSF's management.

KSF's CEO wants you to add certain functionality he thinks is missing from the previous software you submitted to him and add a few features he thinks will be good.

You can and should use the code you wrote for HW3.

This HW requires using c++17.

Unless it is explicitly written, HW3 definitions are still valid.

Like in the last assignment, The methods signature is up to you, but :

1. Use the **const** keyword where possible and needed.
2. Create **getters** and **setters** for all private fields
3. Use the **friend** functions where needed.
4. Use **reference** variables where possible and needed.
5. Use **static** variables and functions where possible and needed.
6. Use **virtual functions** where it is right to do so.
7. Use **Abstract base classes** where it is right to do so.
8. Use c++ style **Casting** where it is needed.
9. Be aware of what **exceptions** your functions might throw!
10. Use **virtual inheritance** where it is right to do so.

You need to understand where to use what, but if you're unsure, feel free to ask me for help.

Your goals are:

- Having the main files, you should provide declaration and implementation while keeping in mind the basic *OOP* concept of *encapsulation*.

  Keep in mind that the included main file is very basic and do not cover all possible scenarios and end cases.

Branch class

Changes to branch class:

- catalog - A STL vector of item (replacing the item pointer array from HW3)
- capacity - Branch catalog capacity should be a field, initiated by the constructor and doesn't change afterwards.
  - Represented by an int
- Add new Item: Inventory management has changed from HW3,STORE_SIZE is no longer relevant.
  - An exceptions should be thrown when the same item is being added for the 2nd time
  - An exception should be thrown when an item is being added but the branch is at full capacity
- Remove item by id:
  - Given an id, the item with said id should be removed from the catalog
  - Should return a pointer to the item
  - An exception should be thrown when trying to remove non existing item
- "Give me your finest" function
  - Given a pointer to subclass of item, return a pointer of the same type to the Item from the same class with the highest price in the branch catalog. Hint: typeid supports polymorphism
  - An exception should be thrown when there is no item with the same type
- Defualt construction : "~" for location, 0 for capacity
- Copy construction – copying the location and capacity only

Item class

- Change to destructor – destructor should print only:
  ```
  Throwing away an item
  ```
  - This chage is made so you can hold the Items in the catalog in any order you like
- Note : There is a function in Main.cpp file that prints a branch's catalog, the function requires a comparator function from Item class.

Computer class

Each computer should also have the following field:

- numOfPorts - Usb ports number
  - Represented by an int, will not change.
- Print connected deviced
  - Should print the string of each peripheral device connect to the computer, from first to be connected to last.
  - Beware of circular dependencies
- A computer should disconnect from every PeriphalDevice connected to him on destruction.

```
First Line : There are x connections to std::string(computer)
New line : string(connected#1)
New line : string(connected#2)
Newline:.
Newline:. string(connected#x)
```

```
There are 2 connections to id 7: Maple 120$, Laptop, AMD
id 5: Sasio 20$, Wireless, Gold, Keyboard with 24 keys
id 6: Goldline 10$, Wired, White, Mouse with dpi : 1000
```

PeripheralDevice class

- Connect function changes :
  It is up to you how to implement the following required functionality :
  1. A peripheral cannot be connected to 2 computers at the same time
  2. A computer cannot have 2 peripheral devices of the same type connected to him
  3. Amount of peripheral devices connected to a computer cannot be higher then his usb port's
  - An exception should be raised when either of this conditions is not met
    - The action that caused the exception should be disregarded.
  - If connecting to the computer the device is already connected to, do nothing.
  - Note – there should be certain prints even when the connection will fail, like in the output txt file.
- Disconnect – disconnect from the computer the device is connected to
  - If not connected to a computer, do nothing
  - Should disconnect from the connected computer on destruction
  - The previously connected computer should then be able to connect to a peripheral device of same type

Keyboard class

No changes

Mouse class

No changes

Tablet class

Each Tablet Is-A PeripheralDevice and is-A computer. also has the following attributes:

- screenSize – the tablet's screen size
  - Represented by an int
- Tablet can connect to computers and get connection from peripheral devices like any peripheral device / computer.
- to string conversion: refer to output.txt for details.
- Note –
  - Tablet is always wireless
  - Tablet is always not a laptop (but also not a desktop!)
  - A tablet can be connected to a computer (but no more then one) and peripheral devices simultaneously

For int to string convertion, I recommend you use std::to_string fuction that comes with <string>.
There is no need in this assignment for default constructors unless stated.

Main Office

Main Office is the "holder" of all branches, a branch should exist only in the Main Office object.

Main Office is a Singleton class, the has the following attributes:

- branches - STL Map of branches:
    - Key to each branch should be his location (will not change).
- Add branch:
    - Given required attributes to create a new branch, add branch to branches map.
    - When branch is added with existing location, an exception should be thrown.
    - I recommend using try_emplace for branch creation in the map.
- Remove branch by location:
    - Given the branch's location, should remove the given branch from the map.
    - When trying to remove a branch by a non existing location, an exception should be thrown.
- Get branch of location:
    - This method should return a reference to the branch with the given location.
    - Should throw an exception if no such branch exists.
- Print branches methods:
    - By location – alphabetically
    - By value of the branch (sum of all items' prices)
        - If 2 or more branches has the same value, then print them ordered alphabetically
    - Note that the methods should receive a function pointer to a function that prints the branches' catalog (such as the function defined in main.cpp).

Methods

Same from HW3

BUT – you need to implement a copy constructor for the Branch class.

There is (also) no need to implement :

- getter/setter for branches in MainOffice
- setter for location in Branch (as it should not change)
- setter for number of ports in Computer (as it should not change)

Required exceptions

Name of Exception class	**Thrown by**	**What**
ExistingBranchInsertError	MainOffice::addBranch	Trying to add a branch with an already existing location
NonExistingBranchRemoveError	MainOffice::removeBranch	Trying to remove a branch with a non existing location
NonExistingBranchRetrieveError	MainOffice::getBranch	Trying to retrieve a branch with a non existing location
ExistingItemError	Branch::addItem	Trying to add an item with an id already in the catalog
FullCatalogError	Branch::addItem	Trying to add an item to a full catalog
NonExistingItemError	Branch::RemoveItem	Trying to remove an item with a non existing id
NoneExistingItemTypeError	Branch::giveMeFinest	Trying to get an item with a non existing type
ConnectError	PeripheralDevice::connect	Failed connection attempt

All exceptions should inherite from std::exception.

The exceptions should be declared and implemented in a single file, "HWExceptions.h" .

Evaluation

Homework exercise provided with the following example program files and corresponding outputs:

1. main.cpp
2. output.txt

You should be able to compile your code with “main.cpp” and receive the correct output in “output.txt”.

Submission should only include the following files :

MainOffice.h/.cpp, Branch.h/.cpp, Item.h/.cpp, Computer.h/.cpp, PeripheralDevice.h/.cpp, Mouse.h/.cpp, Keyboard.h/.cpp, Tablet.h/.cpp, HWExceptions.h

Bonus- Competition

The top 5 fastest code will be rewarded with a bonus to the HWs grade.
The bonus will be given in the HWs grade calculation :

$$Final\ Hw\ grade = \frac{bonus + \sum_{i=1}^{4} hw_i.grade}{4}\ .(maximal\ grade : 100)$$

#1 wil be given 15 pts bonus
#2 wil be given 10 pts bonus
#3,4,5 will be given 5 pts bonus
10 pts will be given for every submission the passes the TA's (Omer) benchmark results.
Benchmarking method will be published later on this week .
Important : If you want to participate, please #define ID in "Item.h" one of the submitter's id number. If such define will be missing, your code won't be part of the competition.

In Item.h:
#define ID 123456789 (replace with your ID number)

Make sure you keep the C++ syntax convention and submit the files exactly as described in the "Oop – Cpp – Conventions and Requirements" file in Moodle.

GOOD LUCK! ☺