Object-Oriented Programming

Spring Semester 2022

Homework Assignment 4.

23/05/2022
Last updated–
05/06/2022

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.

In this assignment you are very much encouraged to use STL algorithms for fast and easy written code. This will also help you get a better grasp of the STL library.

And remember – google is your best friend!

(but don't hesitate asking me questions)

Definitions

Read carefully!

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

1. Casting.
2. STL.
3. Exceptions.
4. **Extremely difficult -** Understaing the same code you wrote just days/weeks before.

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

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

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

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!

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 files are very basic and do not cover all possible scenarios and end cases.

Branch class

Changes to branch class:

- A STL vector of item pointers (instead of array), known as "catalog"
- 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
  - Note – the name of the function is deleteItem, but do not delete the item pointer in the method.
- "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. **Hint: typeid supports polymorphism**
  - An exception should be thrown when there is no item with the same type
- Print catalog
  - By id
  - By price (use stable sort)
- Note : a value of a branch is : $value = \sum_{item \in branc} \quad item.price$
- Defualt construction : "~" for location, 0 for capacity
- Copy construction – copying the location and capacity only

Item class

No changes.

Computer class

Each computer should also have the following field:

- Usb ports number
  - Represented by an int, will not change.
- (Beware of the includes you use) Print connected deviced
  - Should print the string of each peripheral device connect to the computer, from first to be connected to last.

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

```
There are 2 connection 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

- (Beware of the includes you use) 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 already connected to, do nothing.
  - Note – there should be certain prints even when the connection will fail, like at the output txt file.
- Disconnect – disconnect from the computer the device is connected to
  - If not connected to a computer, do nothing
  - Should disconnect on destruction

Keyboard class

No chagnes

Mouse class

No changes

Webcam class

Each webcam Is-A PeripheralDevice and also has the following attributes:

- resolution
  - Represented by a string

For int to string convertion, I recommend you use std::to_string fuction that comes with <string>.There is a known issue with mingw and std::to_string, if you encounter problem with this function, contact me.
There is no need in this assignment for default constructors unless stated.

Bonus Class – Main Office

This is a bonus class, not mandatory – submission with appropriate implementation will be given 10 pts bonus

I highly recommend trying, to get a better hold of STL's map.

Note: compilation errors will result in point deduction, running time errors or test mismatch won't.

The class is not in the main given to you, so you need to test it yourself.

main office should have :

- STL Map of branches:
  - Key to each branch should be his location (will not change)
- Add branch:
  - Add branch to branches map

- When branch is added with existing location, an exception should be thrown

- Delete branch by location:
  - Given the branch's location, should delete the given branch
  - When trying to delete a branch by a non existing location, an exception should be thrown
- Branch printing functions:
  - Print by location
- There should be only one main office in existence (Singleton)

Required methods signature :

addBranch(string location,int capacity) – should initialize a branch(location,capacity) in the map
deleteBranch(string location) – should delete and destroy the branch with location = location
printByLocation() – should print the branches with printById method , sorted by location
getInstance() – returns the mainOffice instance (singleton)
getBranches() – returns the mainOffice's branches map, return by reference.

Required getter/setter method signiture

Marked with yellow – bonus related

Class	Method
Branch	getCapacity()
Branch	getCatalog(void)
Computer	getPorts()
Webcam	setResolution(..)
Webcam	getResolution ()

The required getter/setter methods from HW3 are still required.
Without the above methods **exact** signature, your code might not compile with the test code and this will result in point deduction,
Note that you are required for a getter/setter for **every** private field (with exclusion) , but only the above will be tested in the test code, the rest will be checked manually.

Exculded methods

Same from HW3
BUT – you need to implement a copy constructor for the Branch class.

Required exceptions

Marked with yellow – bonus related

Name of Exception class	Thrown by	What
ExistingBranchError Note – should be declared and implemented in MainOffice.h	MainOffice::addBranch	Trying to add a branch with an already existing location
NonExistingBranchError Note – should be declared and implemented in MainOffice.h	MainOffice::deleteBranch	Trying to delete a branch with an 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::DeleteItem	Trying to delete an item with a non existing id
NoneExistingItemTypeError	Branch::retrieveFinest	Trying to get an item with a non existing type
ConnectError	PeriphrelDevice::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. main _output.txt

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

“main_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

Webcam.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
Benchmarking method will be published later.
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! ☺