Object-Oriented Programming

Spring Semester 2022

Homework Assignment 3.

26/04/2022

Updated
02/05/2022

Contents

Introduction 3
Definitions 4
Branch class 5
Item class 5
Computer class 6
PeripheralDevice class 6
Keyboard class 6
Mouse class 6
Required getter/setter method signiture 7
Exceptions 7
Evaluation 8

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.

Definitions

Read carefully!

For the third homework, you will work with the Inhertiance mechanism in C++, the relevant tutorials are 1 to 8.

In this assignment you will model and implement KSF's store software.

KSF is a new chain store brand, that currently sales computers and computer equipment,

but has serious future plans on expanding its selection of items, so they asked you that take that in account when creating the software.

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.

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.

- Take care of correct dynamic memory management. Be aware of both allocating and releasing resources.

  ☒ In this submission, you are **RESTRICTED** from using *STL* library regardless of prior knowledge you have.

  ☑ You **can** assume that the input for the functions you write will be legal.

Branch class

Each of KSF's branches are going to have:

- An Array of Item pointers, the store's item catalog
  - The array should have the defined size – STORE_SIZE in the Branch.h file , should be 10 as defualt.
- Location
  - Represented by a string
- Add item function
  - Adds a new item to the store
  - If the store is full, throw away the oldest item in the store and insert the new item at its place
    1. Oldest as in the first to be the added to the store

Item class

Each item has the following attributes:

- price
  - Represented by an int
- manufacturer
  - Represented by a string
- id
  - Represented by an int
  - The first id is 1, each item is given the next free id (the last id given +1)
  - The Id should never change once given to an item
- To string convertion function (cast to string operator overload)
  - Should return a string with the item id, manufacturer and price with a comma between them.
  - Please notice, that sub-classes of Item might need to override this conversion, and add extra details of the **Is-A Item** to the string, as you can see in the main_output.txt. Beware of unnecessary code duplication.
  - For example :

id	2
Price	60
Manufacturer	Dell

Should return the string :

```
id 2: Dell, 60$
```

- Destruction
    - When item is destroyed, print the following:

```
Throwing away an item
```

Computer class

Each computer Is-A Item and also has the following attributes:

- cpu
    - Represented by a string
- Is a laptop
    - Represented by a bool

PeripheralDevice class

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

- color
    - Represented by a string
- Is wireless
    - Represented by a bool
- Connect function
    - Takes a computer as argument, and prints :

```
*string(device) * is Connecting to
computer : *string(computer) *
```

Keyboard class

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

- number of keys
    - Represented by an int

Mouse class

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

- dpi
    - Represented by an int

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.

Required getter/setter method signiture

Class	Method
Item	setPrice()
Item	getPrice()
Item	getId()
Item	setManufacturer(…)
Item	getManufacturer()
Branch	getCatalog(…)
Branch	setLocation(…)
Branch	getLocation()
Computer	setCpu(…)
Computer	getCpu()
PeripheralDevice	setIsWireless(…)
PeripheralDevice	getIsWireless()
PeripheralDevice	setColor(…)
PeripheralDevice	getColor()
Keyboard	setNumOfKeys(…)
Keyboard	getNumOfKeys()
Mouse	setDpi(…)
Mouse	getDpi()

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 exceptions) , but only the above will be tested in the test code, the rest will be checked manually.

Exceptions

There is no need to implement

- setCatalog for Branch
- Copy constructor for Branch
- Assigment Operator for Branch
- setId for Item

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 :

Branch.h/.cpp

Item.h/.cpp

Computer.h/.cpp

PeripheralDevice.h/.cpp

Mouse.h/.cpp

Keyboard.h/.cpp

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! ☺