Project 1: Gilded Rose Store Inventory

CPSC 131 Spring 2019

Introduction

Hi and welcome to team Gilded Rose. As you know, we are a small inn with a prime location in a prominent city ran by a friendly innkeeper named Allison. We also buy and sell only the finest goods.

You are provided a listing of items from the latest shipment to the Gilded Rose with each items name, sell in value, and quality. The sell in value is based on gold coins (there is no fraction of a coin). The store wants you to build a program that is able to store each item in an internal list and later retrieve the information in a number of ways.

Objective

You are given a partial implementation of one header file, GildedRose.h. `Item` is a class that holds the information for each item for the inn. `GildedRose` is a class that holds an internal listing of many `Item` objects. This inventory should hold at least 10 items. For this you can use arrays, the std::array class, or even the vector class.

Complete the implementation of these classes, adding public/private member variables and functions as needed. You should choose an appropriate data structure to maintain this inventory with an unknown size known only at runtime. Your code is tested in the provided `main.cpp.`

You will need to implement the following functions:

- **`Constructors/Destructors`** - Initialize your data. Allocate memory if using a native array. The destructor should deallocate memory if using a native array.
- **`size()`** - This should return the number of items currently for sale (this is different from the max).
- **`get(size_t)`** - This should return the item with the matching index. For example if given an index of 3, you should return the item at index 3 in the list.
- **`add(Item)`** - This should add another item for sale in the Gilded Rose by adding it to your inventory.
- **`operator[](size_t)`** - This should perform identical to the `get(size_t)` function.

Initially the given code will not compile. As you complete the code, the tests should start to pass in `main.cpp.`

Source Code Files

You are given “skeleton” code files with declarations that may be incomplete and without any implementation. Implement the code and ensure that all the tests in `main.cpp` pass successfully.

- `GildedRose.h`: This is to be completed
- `main.cpp`: The main function tests the output of your functions. You may wish to add additional tests. During grading your main.cpp file will be replaced with the one you were provided with.
- `README.md`: You must edit this file to include your name and CSUF email. This information will be used so that we can enter your grades into Titanium.

Hints

This is an introductory assignment meant to test your knowledge of constructing a class with storing the inventory and some member functions that change the state of that inventory. Decide which data structure you wish to use for your underlying inventory, array, std::array or vector.

Make sure your code compiles, and then try and solve the logic. Focus on solving one test at a time.

Obtaining and submitting code

We will be using GitHub Classroom to distribute the skeleton code and collect your submissions. This requires you to have an account on github.com. If you are new to GitHub, do the following to get started:

1. Create an account at github.com. You may want to use this account to show a portfolio of your work to prospective employers in the future, so choose something professional.
2. Read Understanding the GitHub Flow and Hello World at GitHub Guides.
3. Read the instructions below for instructions on how to test.

Once you understand the basic operation of git, click the assignment link to fork your own copy of the skeleton code to your PC.

Do not fork your repository to your personal github account (instructors have admin access to private repositories under https://github.com/CSUF-CPSC-131-Spring2019/). Your code should have a URL like https://github.com/CSUF-CPSC-131-Spring2019/project1-brians, NOT https://github.com/brian/project1-brians.

https://classroom.github.com/a/ouXvMuPI

Then edit your code locally as you develop it. As you make progress, commit and push your changes to your repository regularly. This ensures that your work is backed up, and that you will receive credit for making a submission. Don't wait until the deadline to learn how to push code!

Testing

Unless otherwise directed, use the following command to compile your program:

```
clang++ -g -std=c++14 main.cpp -o test
```

To attempt to run the compiled test program, use the following command:

```
./test
```

Grading rubric

Your grade will be comprised of two parts, *Form* and *Function*.
Function refers to whether your code works properly as tested by the main function (80%).
Form refers to the design, organization, and presentation of your code. An instructor will read your code and evaluate these aspects of your submission (20%).

Deadline

The project deadline is Monday, February 18th before midnight.

You will be graded based on what you have pushed to the main branch of your GitHub repository as of the deadline. Commits made after the deadline will not be considered. Late submissions will not be accepted.

Your code must compile/build for it to be tested and graded. If you only complete part of the project, make sure that it compiles before submitting.

Credits

This project was inspired in part from the Gilded Rose Refactoring Kata. The actual coding kata is very different but the setting and item data were reused for this assignment.