COP 3331: Object Oriented Design
Project 2
Due July 8th at 11:59 P.M.

You are not allowed to use the Internet. You may only consult approved references[*].
This is an individual project.
This policy is strictly enforced.

Project Objectives

To learn more about

- Object modeling based on project requirements
- Class creation
- Using objects
- Basic user interfacing

Submission Guidelines

- A zipped Visual Studio solution submitted through Canvas. We will be using VS 2013 for this course.
- **Points will be deducted for not submitting a proper VS solution.**
- **No late submissions will be accepted.**

For full credit, the code that is submitted must:

- Have a comment block including your name.
- Be implemented in a file using the specified file name, if applicable.
- Be readable and easy to understand. You should include comments to explain when needed, but you should not include excessive comments that makes the code difficult to read.
  - Every class definition should have an accompanying comment that describes what is for and how it should be used.
  - Every function should have declarative comments which describe the purpose, preconditions, and postconditions for the function.
  - In your implementation, you should have comments in tricky, non-obvious, interesting, or important parts of your code.
  - Pay attention to punctuation, spelling, and grammar.
- Have no memory leaks.
- Use no magic numbers. Use constants in place of hard-coded numbers. Names of constants must be descriptive.

[*]You may use all class material and the C++ reference sites posted at the beginning of the semester. Any additional resources must be pre-approved. You must cite all references used.

Some additional coding guidelines which will help you to create efficient, organized code (optional)

- Using #define header guards to prevent multiple file inclusion. The typical form of the symbol name is `<FILENAME>_H_`
- Limit the amount of code you copy and paste. If you need to reuse a section of code, you should create a function to place it in instead.
- Define functions inline only when they are simple and small, say, 5 lines or less
- Function names, variable names, and filenames must be descriptive. Avoid confusing abbreviations.
- It is best to use spaces instead of tabs. Most IDEs and text editors will give you the option to insert spaces in place of tabs (typically 4).

Project continues. . .

Project Details

The essential idea here is to revise your kiosk code from project one so that it is more

- efficient
- easy to understand
- modular
- scalable

This does not necessarily mean you will need to redesign your program, but you will need to find areas that need improving and implement some changes to make this happen.

Your revised code should **NOT** use any conditional statements in a way that prevents your program from scaling. As an example

```
Event movie1("Transformers", "A movie about robots");
Event movie2("Star Wars", "Space Ballad Extraordinaire");
Event movie3("Hello, world!", "A punny movie every coder should C");

//Some code printing menus...
cin >> menuChoice;

if(menuChoice == 1)
{
    //do something with movie1
}
else if( menuChoice == 2)
{

    //do something with movie2
}
else if(menuChoice == 3)
{
    //do something with movie3
}

....
```

is not scalable! What would happen if I told you that you would have to support up to 100 movies? or 1000? A lot of copying and pasting, that's what. Note that a switch statement is logically equivalent and is just as bad. *This is* ***NOT*** *allowed in your program!*

As we talked about in class, one method for solving this problem is by using an array to store items in and then index into the vector with the user choice. A vector could also be used now that we have talked about templates.

After revising your code, write a small report detailing

- The major changes to your code
- Why these changes were made
- Why these changes are an improvement

In addition to revising your original code, you also need to move all shopping cart code to it's own class (if you have not already done so).

Extra Credit [10pts]

For extra credit, remove the static declaration of your inventory objects from the beginning of your program and add code to read inventory items from a file of comma delimited values. An example file will be posted on Canvas. Include a readme file that tells the user how the file is structured.

Some classes and functions that will be very useful for this task are

- The fstream class
- The stringstream class
- getline() (an overloaded function part of all stream classes)
- ignore() (also overloaded for all string classes)