CS273 ASSIGNMENT #2a: Midsemester Project Part 1, Object-Oriented Design – Inheritance, Polymorphism, and Vectors

Group Members: _____ DUE: Friday, 9/20

Grade:

CATEGORY	POINTS	
PR2_Crawlspace: Part 1 – Design **(Please read the Heads-up Below!)**		50
TOTAL		50

Objectives:

- Apply the concepts of information hiding, inheritance, and polymorphism – and, in general, all the software engineering skills we've been discussing in the first week of class.
- Review using STL containers such as the vector – also previews what we'll study in detail later this semester
- **Above all: Practice building moderately complex software from specification in teams!**

Design Project (this is part 1 of 2 parts to this problem)

Create the following folder on CS1 and place your solution in this folder.
`\\CS1\Students\Your Year\Your Folder\CS273\PR2_Crawlspace`

Do not implement the Crawlspace program until you have also looked at Assignment 2b. Your implementation is not due until the next assignment (assignment #2b).

The goal for this assignment is to practice building moderately complex software from description (specification). This is much harder than you would think, and we can only get better if we practice doing this.

For this exercise, imagine you are in a software company, and you have been given this draft problem specification. The class interface defined needs to be strictly followed because of external dependencies with other software components. Other software groups depend on you to accurately follow the defined classes. Furthermore, much of the code is already written by other members of your team. You job is to understand the design requirements, and complete the implementation.

I would like you to study the problem description and come up with:

1. A refined **requirement specification** of the software that needs to be created in your own words. This should primarily describe **what** you think the software should do, **not how it will implement** its functions. My seed requirements will get you started but…
2. I want detailed **use cases** for ALL the scenarios you imagine the software will be used in. Look at the phone directory example in the text book (**section 1.5**) to guide you with this.
3. I'll also want **UML diagrams** to describe the relationship between the classes described in the problem. You do not need to describe any sequence diagrams unless you wish too.
4. Eventually, I want some high level **pseudo-code** to describe how all methods work.

Please use all the design tools we have discussed in the Phone Directory example (pages 102-121). Imitate what was done there. **You likely will need to talk to Matt and Gwen** (treat us as if we were your clients) **to do this well.**

At the end of this exercise, you should feel confident in implementing the solution to this problem. If you do not feel this, please go back to your design, and address your doubts.

!!! Major "Heads Up" for this assignment!!!

- **It will be helpful to work in groups of two for this part of the assignment.** Describe to each other what you think the software, classes, and methods should do. **However, in the next part of the assignment, you will need to implement your own program individually.**
- This is a large project and you absolutely won't be able to finish it if you start the night before it is due.
- **Plan on at least 6 hours or more for completing both assignments 2a and 2b**.
- **Design and implementation often goes hand-in-hand**. So please look at the starting code now, to understand how your design is going to co-exist with the current implementation.
- **When you are stuck, use the Visual Studio Debugger**. If you've never used it before, Matt is happy to show you in office hours or in class.

A seed requirements specification is given, below, and seed code is provided on the GoogleDrive.

Seed Requirements Specifications

You are to create a demo a backbone for a text adventure game. For ideas about how text adventure games work, check out and take an hour to play Carruther's *Original Adventure* here:

https://www.amc.com/shows/halt-and-catch-fire/exclusives/colossal-cave-adventure

You will not have to come up with the whole game – only a subsystem that manages locations and demos exploring them.

For your subsystem some code has already been provided. There are six classes: **Crawlspace**, **Explorer**, **Location**, **Object**, **Treasure**, and **Exit**. Exits and Treasures are subclasses of Object. Crawlspace is a container class (has-a relation) for Locations, and Location is a container class (has-a relation) for Exits and Treasures. A Crawlspace object organizes and grants access to its contained locations for an Explorer object. Locations have a description and exits to other locations in a three-dimensional, discrete space (i.e. explorers can move up and down as well as in a plane with themselves, and, when explorers move, they move to another distinct location). Optionally, locations have treasures. Exits have descriptions, a name, and an integer representing the location to which they lead. Exits cannot be picked up by the explorer; if the explorer attempts to pick them up, a message gets displayed to the screen that reads "What a concept!" Treasures have descriptions, a name, and an integer representing their point value, and can be picked up by the explorer and added to inventory. If taken by the explorer, the message "You got <N> points" should display to the screen, where <N> is the value of the treasure taken.

Explorers have an inventory and current location in the crawlspace. Gameplay works like so: the Explorer starts at the starting location. Each cycle of the game loop the Explorer is allowed to Examine, Take, or Move. If the Explorer examines, the system should ask them what they want to examine. If the explorer responds "Room" to the prompt, then the description of the location is displayed to the screen. Otherwise, the description of the object selected is displayed. If the explorer Takes then s/he should be asked what s/he wants to take. If the user names a treasure that is not there the message "That's not here" is displayed to the screen. If the Explorer Moves, then a list of all exits is displayed and the Explorer given a chance to pick an exit. If the Explorer tries to move down a non-existent exit, the message "You can't move that way!" is displayed to the screen; otherwise, the Explorer's location is updated.

Some code has already been written for you; you should modify it in appropriate ways. To make use of this code you should remind yourself about C++ vectors (typically covered in CS172).

CS273 ASSIGNMENT #2b: Container Foundations: Object Oriented Design and Vectors / Linked Lists

MY NAME: _____ **DUE: Monday, 9/30**

Grade:

CATEGORY	POINTS	
PR2_Crawlspace: Implementation (**Please start early!**)		50
TOTAL		50

Objectives:

- **Building software from specification.**

Programming Project: implementation (part 2)

PR2_Crawlspace

Place your Visual Studio solution in the following folder:
\\CS1\Your Graduation Year\Your Folder\CS273\PR2_Crawlspace

Please note that every member of your team should be implementing this individually. Base your implementation on the specification that your team came up with in **assignment 2a.** If your team's specifications turned out to be insufficiently complete, you should refine them further and note in your comments what changes you have made.

Please remember to use the Visual Studios Debugger to trouble-shoot any problems you may encounter. **Debugging is a critical problem solving skill** we all need to develop in Computer Science.

Thank you for your hard work!