CPSC3175 Final Project

The Adventure/Role Playing Game

Due Date: The day and time of the Final Exam. Due to our current situation we will do the presentations at different times throughout the finals period. *This is also the presentation day.*

The Game

Your task is to create and implement a text-based adventure/role playing game using the C# programming language. You have been given a simple framework that lets you walk through a few rooms. **You must use this as a starting point**. **You must implement at least 5 software design patterns in the project (Simple patterns such as Enumerators, Designated Constructors, etc. are NOT included)**. **You should use loosely coupled components and cohesiveness in your design**.

Please, read this

Implementing at least 5 design patterns is not optional, in fact, that affects your overall grade. The points for the project are assigned according to Table 1 and Table 2 (see page 5.) The maximum number of points is 1000. If you implement 5 software design patterns (or more) properly, whatever points you earn according to the features of the game will be your grade. If you earn 950, your grade is 950 points. If you only implement 4 software design patterns then your 950 points will only be 760. If you only implement 3 software design patterns then your 950 points will only be 570, so on and so forth. In other words, the percentage of the number of software design patterns will be the percentage applied to the points earned in Table 1 and Table 2.

1 Read The Code

Reading code is an important skill that you need to practice. Your first task is to read some of the existing code and try to understand what it does. By the end of the assignment, you will need to understand most of it if not all of it.

2 Make small extensions

As a little exercise to get warmed up, make some changes to the code. For example:

- change the name of a location to something different.

- change the exits — pick a room that currently is to the west of another room and put it to the north
- add a room (or two, or three, ...)

These and similar exercises should get you familiar with the game.

3 Design Your Game

First, you should decide what the goal of your game is. It should be something along the lines of: You have to find some items and take them to a certain room (or a certain person?). The items might be keys to open locked rooms. Then you can get another item. If you take that to another room, you win.

For example: *You are at Columbus State University, Main Campus. You have to find out where your lab class is. To find this, you have to find the front office and ask. At the end, you need to find the exam room. If you get there on time, and you have found your textbook somewhere along the way, and you have also been to the tutorial class, then you win. And if you've been to the Cafeteria more than five times during the game, your exam grade gets reduced by 50%.*

Or: *You are lost in a dungeon. You meet a dwarf. If you find something to eat that you can give to the dwarf, then the dwarf tells you where to find a magic wand. If you use the magic wand in the big cave, the exit opens, you get out and win.*

It can be anything, really. Think about the scenery you want to use (a dungeon, a city, a building, etc.) and decide what your locations (rooms) are. Make it interesting, but do not make it too complicated. (I would suggest no more than 30 rooms.) Put objects in the scenery, maybe people, monsters, etc. Decide what task the player has to master.

Please, create a One Page Document describing your game concept. You should identify all members in the team if you are to work in a group (groups cannot have more than 3 members.)

4 Implement the Game

Decide what classes you need to implement the game, then implement and test them.

5 Levels

You may choose to create the ***single-player*** version (default) or the ***multi-player*** version (client/server.) The latter is more challenging since it requires

thread synchronization in the program as well as networking. This networking will require a substantial change on the base system.

Regardless of the version you choose, the **base functionality** that you have to implement is:

- The game has several locations/rooms.
- The player can walk through the locations. (This was already implemented in the code you were given.)
- There are items in some rooms. Every room can hold any number of items. Some items can be picked up by the player, others cannot.
- The player can carry some items with him/her. Every item has a weight. The player can carry items only up to a certain total weight.
- The player can win. There has to be some situation that is recognized as the end of the game where the player is informed that he/she has won.
- Implement a command 'back' that takes you back to the last room you have been in. If you had visited 10 rooms in a row since the beginning of the game and you used the command 'back' 10 times you should go back to where you started. Therefore, the system should keep track of the rooms you have visited using the command 'go' but not the command 'back'.
- Add at least five new commands (in addition to those that were already present in the code and the back command).

Challenge tasks:

- Add characters to your game. Characters are people or animals or monsters — anything that moves, really. Characters are also in rooms (like the player and the items). Unlike items, characters can move around by themselves.
- Make the characters able to interact with the player similar to using a Command pattern.
- Extend the parser to recognize three-word commands. You could, for example, have a command

  ```
  give bread dwarf
  ```

  to give some bread (which you are carrying) to the dwarf.
- Add a magic transporter room — every time you enter it you are transported to a random room in your game.

- Each item may also have volume besides weight and the player can carry items only up to a certain volume.
- Add the multiplayer/threading capabilities to the game.
- In a multiplayer game, have the player drop all items if either dies or leaves the game.
- Create locked doors and use keys to open them.
- Generate your rooms, their connections, and all items randomly every time you start your game or the server if it is a multiplayer game.
- Create a game reader to have the game read a file where the layout of the rooms, items and other characters start. This basically substitutes the createRooms: method with a readLevel: method.
- Create a role-playing game where the player may grow in a series of areas (strength, spells, mastery, etc).
- Provide a way to have the players wear armor or use weapons.
- Make weapons/armor wear out.
- Give selling/buying value to all items.
- Create a trading room where you can buy or sell items.
- Create a battle system where your character may engage in fights with other characters.
- Create a quest system which can be used to have your character complete specific tasks to obtain items or information that otherwise would be unaccessible.
- Create a way to leave/save the game and come back to it later (single-player).
- Create a way to leave and come back to the game (multiplayer).
- Others. You can come up with additional challenge tasks yourself. You have to discuss those with me and get my approval before you implement them. I will advise you if you have picked something that is too difficult or too much work.

Note: If you implement a challenge task you should use it in the game.

Presentation	Report	Log	Game	Total
100	100	100	700	1000

Table 1: Final Project Deliverables.

Score/Team	**One member**	**Two members**	**Three members**
490-553	Base functionality	+2 Challenge tasks	+5 Challenge tasks
560-623	+2 Challenge tasks	+5 Challenge tasks	+7 Challenge tasks
630-700	+5 Challenge tasks	+7 Challenge tasks	+10 Challenge tasks

Table 2: Game by number of members and tasks.

6 Submission and Assessment

You have to submit the **C#** project in CougarView/D2L. All code must be professionally written (comments and indentation!) and will be graded for

- correctness
- appropriate use of language constructs
- style (commenting, indentation, etc.)
- difficulty (extra credit for difficult extensions)

You also have to submit a report that includes

- the name and a short description of your game
- the description should include at least a user level description (what does the game do?) and a brief implementation description (what are important implementation features?) and which design patterns and where they are used.
- special features of your game
- known bugs or problems (Note: for a bug in your code that you document yourself, you may not lose much credit — maybe none, if it is in a challenge task. For bugs that we find that you did not document you will probably lose credit. Test your system thoroughly.)
- a log of all the time allocated to this project. The log should include date, time of day, length of time, team members and task for each entry. You may use a spreadsheet to keep this log.

The assignment must be handed in and presented to a panel of judges (faculty members or just me). **Late submissions will not be accepted!** If you, for any reason, cannot hand in or post the project on the due date, you have to hand your assignment in earlier!

The presentation

Your work will be assessed after a presentation of your game. You will not be graded without this presentation. You are expected to have written all the new code yourself (everything else is plagiarism!) and to be able to explain *in detail* all of the code you have written. Your grade for this assignment will reflect your understanding of the code that you demonstrate in the presentation. You are expected to create an object-oriented designed application.

7 Groups

You may decide to work in groups with no more than 3 people. Each person will be responsible for understanding all of the code. See Table 2 for grading according to the number of members in the team.

8 Consulting

I will serve as a consultant for you. I will keep track of who consults, what they consult about and what the major issues are. There is no excuse for a group not to be able to solve whatever technical difficulties may show up. You should keep track of all open issues, problems and bugs in your system. Keep track of all the time you invest in this project by keeping a log of all the hours, meetings dedicated to your project.

Each entry in your log should contain at least date, start time, end time, total time, brief description of work. You may use a spreadsheet to keep these entries and have a total number of hours invested in the project.