Due Date: Tuesday, October 9, 2018 @ 11:59 PM

Lab Objectives:

* Practice creating simple Classes.
* Declare and instantiate several objects.
* Improve understanding of Class design and how Java classes interact.
* Gain additional experience working with an API.

Book Material Covered:

* Chapter 2, 3, 7

To complete this assignment:

* Check out the proj01 directory from your SVN repository:  
  https://csprojects.cs.ndsu.nodak.edu/csci160/2018/fall/svn/username/proj01
* Read through all instructions and look over the assignment and make a quick mental estimate of how much time you expect proj01 to take you to complete. You should record this initial estimate in the proj01 Questions document.
* Use the Room, Exit, Item, Creature, and World Class APIs to create the following classes:
  + Room
  + Exit
  + Item
  + Creature
  + World
* Create a TextAdventure class that contains a main method to start the adventure.
  + You will declare an instantiate several objects of the classes you have written in this class.
  + Your adventure should include at least the following:
    - 10 different Rooms
    - 3 different Items
    - 3 different Creatures
    - Exits as necessary to connect the Rooms.
* Complete the Blackboard portion of the assignment by answering the questions listed in the proj01 Questions Word document.
* Compare the actual amount of time it took you to complete the assignment with your original estimate. This may be helpful for developing your estimation abilities going forward.

Grading:

* 40 points (Correctness of solution, following the API)
* 30 points (Having at least the necessary number of Rooms, Items, Creatures, etc.)
* 20 points (Proper formatting, using good identifiers, and following style conventions)
* 10 points (Completing the Blackboard portion of the assignment)

**Assignment Description:**

For this assignment you will be creating several user defined classes in order to create a simple object-oriented text adventure game. You will also be creating several instances of the classes you write to create a short text adventure.

As part of your solution you will be using some provided classes including a Player class and a Game class, both of which are described in further detail below. These classes contain the code necessary to actually drive the game as we have not yet learned enough about coding in order to do this ourselves.

A sample adventure game has been included for you to try. Just unzip the file, go into the TextAdventure folder and double click the Start file. Your adventure game can be about any topic manner you like, so be creative.

Although we have not yet discussed how to write our own classes, you can use the API to begin constructing objects of those classes, even though you have not written them yet.

**Player Class Methods:**

The following methods of the Player will be necessary for you to complete the assignment:

|  |  |
| --- | --- |
| Identifier: | Player(String name, String description) |
| Parameters: | name – A String representing the name of the player character.  description – A String representing a description of the player character. |
| Return Value: |  |
| Other: |  |

**Game Class Methods:**

The following methods of the Game class will be necessary for you to complete the assignment:

|  |  |
| --- | --- |
| Identifier: | Game(World world, Player player) |
| Parameters: | player – A Player object representing the player character for this Game.  world – A World object representing the world in which this Game takes place. |
| Return Value: |  |
| Other: |  |

|  |  |
| --- | --- |
| Identifier: | setStartText(String text) |
| Parameters: | text – A String representing text to be displayed at the start of the Game. |
| Return Value: |  |
| Other: |  |

|  |  |
| --- | --- |
| Identifier: | setVictoryText(String text) |
| Parameters: | text – A String representing text to be displayed at the end of the Game. |
| Return Value: |  |
| Other: |  |

|  |  |
| --- | --- |
| Identifier: | start() |
| Parameters: |  |
| Return Value: |  |
| Other: | Calling this method on a Game object reference will cause the game to start and will accept text input until the game is finished. |

**Class Descriptions:**

Room The Room class represents a single Room in the TextAdventure game. Rooms contain Exits to other Rooms, and optionally an Item that the Player can acquire or a Creature that the Player may need to defeat. Rooms also have a name and a description of the Room. A Room may also have a secret Exit that is not listed among the visible Exits.

Exit The Exit class represents a path to another Room. Each Exit has a name and a description as well as which Room the Exit leads into and should also contain some additional text to display when moving through that Exit. An Exit can optionally require a specific Item in order to use or have passage through it blocked by a Creature in the Room containing the Exit.

Note that an Exit only has one direction. Just because you can travel to a different Room through an Exit does not mean that you can travel back to the previous Room once you have arrived in the new Room.

Item The Item class represents anything that the player can pick up or acquire. It doesn’t actually need to be a physical object. Items are used to defeat Creatures or to use certain Exits.

Creature The Creature class represents other entities or characters in the TextAdventure game. Creatures have a name and a description. Optionally creatures may be defeated when the player uses a specified Item while in the same Room as a Creature which will display a special message when the creature is defeated.

World The World class represents a game world. It contains a Room where the player should start their adventure and a Room where the adventure will end.

Player The Player class represents the player character. It has a name and a description of the player character as well as a description of the character. It also stores the player character’s inventory, a collection of the different Items that the player character has picked up or is currently holding.

Game The Game class contains all of the code necessary to drive the TextAdventure game. You do not need to modify any of the code in this class, but you may want to take a peek at it to see how it works.

**Controls:**

The following commands can be used while playing the game.

look <exit, item, creature> – Provides a description of an exit, item, or creature in a room.

go <exit> – Travels through an exit.

get <item> – Picks up an item in a room and add it to your inventory.

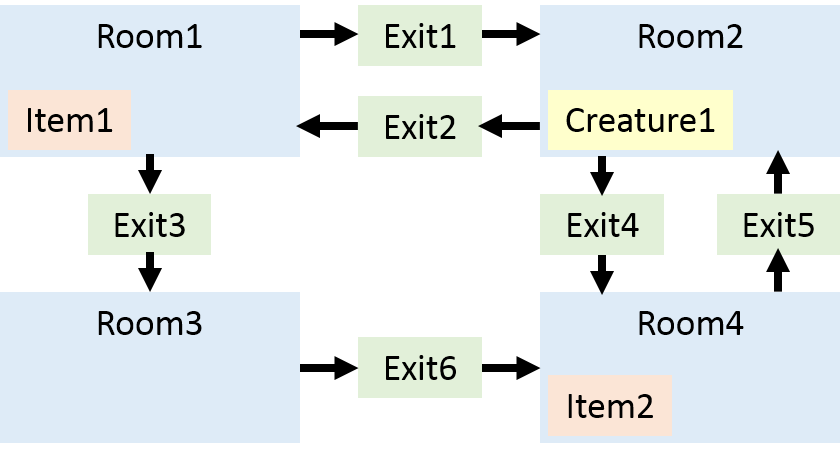
use <item> – Uses an item in your inventory.

inventory – Lists out any items being carried by the player.

help – Prints out a list of game commands.

**Graphical Representation:**

The following is a brief graphical representation of how the different classes interact with each other.



**Challenges:**

If you are experienced with coding, you may wish to extend the game further. Feel free to implement additional features into your game, such as allowing multiple Items or Creatures per Room or including additional ways to interact with those objects. You could add a score system that tracks a player’s progress through the game. Adding the ability to die or otherwise lose the game may also be useful. You could even add RPG elements to the game to add further depth to the game or different solutions based on character attributes.

**Where to Start:**

It is recommended that you start be designing all of the different classes that you will need, based on the provided API documents.

Once you have completed these, you can determine if you have followed the API be checking to see if there are any compiler errors in the Game class. Because the Game class contains code that assumes all of your classes were written according to the API, any compiler errors would indicate a missing or improperly specified method in one or more of your classes.

After resolving those issues, you may want to use a sheet of paper to plan out what your adventure will look like before you start creating instances of your class. That can help clarify your ideas as well as provide a map for how your various Objects will connect together.

Once you have a design in mind, start be instantiating all of the different objects you will need. In some cases, not all of the values will be available. For example, if you make all of your Rooms before you make an Exits, there will not be any value to give to the constructor method for an Exit when you instantiate a Room object. This is okay, as you can always use your mutator methods to set those fields later once you have created all of your objects.

Finally, you should play though your adventure in order to test it and make sure that it works properly. Make sure to test all of the different paths to look for unexpected crashes that can result if you haven't set up some of your objects properly.