Interactive Fiction Engine 1.0

Preparation

Start by downloading the starter code here:
[Note: Normally there is a link here to a .zip file stored on Canvas. The starter code is in the **Interactive-fiction-starter-code** directory in the assignment repository.]

After downloading, add **all files** from inside the zipped folder to an otherwise empty project in the IDE.

Overview

In this assignment you will edit the existing methods and add others so that the infrastructure is in place to support the features listed below.

You should begin by reading carefully through **all** the existing code. Understanding how it is organized and what each piece does is essential to building from this point forward. You will need to think ahead about what methods and instance variables it makes sense to include in each class.

Warning! This assignment involves building and working with code in many different classes and methods. My best advice is to:

- Start early enough. Seriously.
- Use the fact that the code is organized into separate classes and methods to help you keep your own thinking organized.
- Make sure you "**work smarter, not harder"** by adding **one feature at a time** and then **testing** it before going on.
- Make sure you carefully **test** after adding or changing small amounts of code. This should happen often!
- Have I mentioned that you should test frequently?
- DO add additional methods to the classes as needed. If you are working on a more complex feature, ask yourself: "what individual steps does this break down into?" and create separate methods for those steps.

Specification

Your code, when combined with an appropriately designed game – as specified by commands in main() – should include each behavior listed below.

1. Respond to direction commands appropriately, by moving the player to the appropriate new location if there is an available exit in that direction.
    - Treat ANY one-word command not on the list below as an attempt to go a certain direction.
    - If there is no exit in the direction entered, print "There is no exit that direction."
    - If there is an exit in the direction entered, change the player's current location to the appropriate location. Then **print the name and description of that location.**

2. Respond to a "**(L)ook**" command. The game should:
    - Print the name and description of the current location.
    - Print **a list of all items present in that room**.

3. Respond to an "**(I)nventory**" command. The game should print "You are carrying" followed by "nothing" or a list of all the items in the player's inventory.

4. Respond to a "**(T)ake *something***" command, by transferring the corresponding item the current location. to player's possessions.
   Additional specifications:
    - This should only succeed if an item with a corresponding name exists in the current location.
    - This should print one of these two responses:
        - If successful, print "Ok, *itemname* taken."
        - If there's no item with a matching name, print "There is no *itemname* here."

5. Respond to a "**(D)rop *something***" command, by transferring the corresponding item from player's possessions to the current location.
   Additional specifications:
    - This should only succeed if an item with a corresponding name exists in the player's inventory.
    - This should print one of these two responses:
        - If successful, print "Ok, *itemname* dropped."
        - If there's no item with a matching name, print "You have no *itemname*."

6. Respond to "**e(X)amine *something***" by printing the description of the corresponding item.
   Additional instructions:
    - This should succeed if an item with a corresponding name exists in the current location OR the player's inventory.
    - If not successful, print "I see no *itemname* here!"

7. Respond to "**(Q)uit**", causing the game to end. (*This is already implemented. Make sure it still works after any changes you make to the rest of the program!)*

8. For **two-word** input that doesn't match anything above, print "I'm sorry, '*command'* did not make sense in that context."

Sample Game

Here is a transcript of a sample game that includes most of the features listed above:

[Note: Again, normally there is a link here to a file stored on Canvas. The example game transcript is the root directory of the assignment repository.]

Submission

To submit your project, upload single .zip file containing the following

- All **.h** and **.cpp** files for the classes involved in the project.
  (You should not include the utils.cpp or utils.h files.)
- An example **main.cpp** of your own that sets up a sample game with all of the functionality you have working.