Assignment #2: Stubbed class

Background:

This assignment consists of two parts. They should combine to help you think through the methods your object will need to define, and the data it will need to receive and return. The data flow table should help you think through the scenarios your object is involved in, and translate those into interfaces between your object and other objects in the game. Then you will use the table to build a skeleton of your object that defines the interface your object exposes to the outside world. You will also expand the test application that you created in Assignment #1, to help you build and test your object.

Assignment Task Part 1: Data Flow Table

Think about the exercises you have been going through in class, walking through game scenarios and talking about what method calls and data passing is involved. Try to brainstorm all of the game scenarios that involve your object. Think about what data you will need to have in order to perform your assigned tasks, and what data you will need to return for others to use. As you brainstorm, fill out the data flow tables that describe all data that your object uses or returns.

For example, some data flow table entries which might apply to the GameLocations object:

Task	**Called by or Calls This Object**	**Callee Receives this Data**	**Callee Returns this Data**
Move Player	Called by Game Control	New Room Number (or direction to move)	True if move is valid False if move is invalid
Move Player	Calls Cave	Current room	Array of connected rooms

(Your GameLocations implementation might be different based on the decisions your team has made during in-class discussions.)

This table is intended to help you take note of everything you can think of that applies to your object. The number of tasks to be performed varies for each object in the game. You do not need to fill out all of the rows in the table – but make an effort to write down a list that's as complete as possible for your object!

You may be uncertain or have questions – that's expected! Do your best, write down question marks where you're unsure, and use your notes to discuss your questions with your mentor and teammates in class later.

Data flow tables for Object _____

Task	Called by or Calls This Object	Callee Receives this Data	Callee Returns this Data

Assignment Task Part 2: Stubbed class

Now that you have thought through the scenarios that apply to your class, it is time to turn that into a set of properties and methods. Create a stubbed class for your assigned object(s) based on the data flow table you created in Assignment Task Part 1. The stubbed class is ONLY a definition of properties and methods – not an implementation. Your methods will be empty. The point is to create something your teammates can start working with to call your code, not to create something that works yet. Requirements for the stubbed class (or classes):

- Is in correct code syntax
- Contains complete member function signatures, including:
    - Function name
    - Argument name(s) and data type(s)
    - Return value data type
- Includes a dummy return line – for functions that have a return value
    - Example "return 0;" for a function that returns an integer
- Contains a comment inside the function body with:
    - The purpose of the function
    - The names of any other functions that it will call
- For all functions with return values, a comment indicating what the return value is

The stubbed class should also contain declarations for any member variables that are needed (that belong to the class – not the local member functions). The class must build without errors (even though the function implementations don't yet exist).

When you're done, push your code to your team's repo. **You will not receive credit for any assignment unless it is committed to the repo, or otherwise in your project manager's hands!**

Game Control specific instructions:
Without everyone's stubs, it will be difficult for you to make much progress yet. Write stubs you see a real need for in Game Control, especially focusing on how you'll work together with the User Interface object to start the game and display the current room. Beyond those stubs, you should start integrating everyone's objects together into one project. Get a copy of the "empty" object code everyone just turned in last week. Combine it all into one project, make changes as necessary to get it building, and push those changes back to the students who own those objects.

For projects in C#/Visual Studio, here are some tips:

- The objects either need to share the same "namespace" or declare that they are "using" each other's namespaces.
- Adding a file to a workspace usually causes Visual Studio to make a copy. To leave a file in another location, add the file with "add existing item" and choose "add as link" off the Add button sub-menu.