Deliverable 2

For this assignment you will focus on encapsulation which is the implementation of abstraction. In the last assignment you designed what your core objects would look like. In this assignment you will write the code to make the classes to define these core objects. You will need to write the classes for the Actor, Item, and Map Cell classes that you designed in Deliverable 1. You will also need to create a WPF form to test your objects and classes.

Some things that might be useful to understand and use for this implementation would be read only properties, and enumerators (enum).

# Actor

An actor can only be instantiated with an overloaded constructor that accepts name, title, HP, attack speed, and starting position.

The name is always capitalized and the titles are title cased. For example, if “Bob The Awesome” is entered, it should be displayed as “Bob the Awesome”. You should be able to get either the name or the name with title from the object. For example, “Bob” or “Bob the Awesome”. There needs to be a way to get either format. You will need to write a method to create the title case for the names. Do not use built in gadgets.

Here are some examples of names and titles title cased:

Mary Mistress of Doom Barry Bane of all Dragons Lisa the Bastion of Society

Davy Appleseed the Planter of Trees Mandy Monroe of Elderon

Actors have a maximum number of HP that they are allowed which is set at construction. The current number of HP is important to track because, when an actor is injured HP are taken away. If the actor ever reaches 0 HP, it is considered dead. The number of HP can never go below 0. However, when an actor is healed, the HP is increased but the value can never go over the maximum.

After initially setting the HP in the constructor, the value can only be modified by the actor taking damage (HP is decreased) or getting healed (HP is increased). The current HP of an actor should be accessible and readable to display at any time.

The X and Y coordinates cannot be modified directly. After they are initially set at construction, they can only be modified through a method to move the actor. An actor can only move up, down, left, or right. Diagonal movement is not supported yet in this game. Using an enum here makes a lot of sense. The position of an actor needs to be able to be read at any time.

Attack speed is how quickly an actor can react. It cannot be changed after set, but must be able to be read.

# Item

Items are things that actors can use. Items have a name and a value that is how much they affect actors.

An item can only be instantiated with an overloaded constructor that accepts name and value of effect.

The Name and value of effect can be changed and read. Possible names of items are: Sword, Healing Potion, Axe, and Extreme Healing Potion.

# MapCell

MapCells are locations that an Actor or Item may be at any point in the game. MapCells start undiscovered or invisible. Whenever a player enters a mapcell, it becomes visible.

MapCells will need to track as to whether or not they have been discovered by the player.

MapCells will need to track of whether or not an enemy actor (called a Monster) is in that cell.

MapCells will need to track of whether or not an item is located in that cell.

MapCells can only contain a monster or an item.

# Test Application

You will need to create a WPF application to test your objects. The test application needs to create an object for each one of the classes. Use TextBoxes, TextBlocks, Buttons, and any other controls to create a test environment for your code. The applications should prove the following:

* The actor object is created and displayed.
* The name and name with title can be retrieved independently.
* HP is accessible and can be modified with damage and heal buttons.
  + When buttons for healing and damaging are pressed, the appropriate text in the display for the actor must update.
* Position is modified by moving the actor up, down, left, or right.
  + When buttons for moving up and down are pressed, the appropriate text in the display for the actor must update.
* The item object is created and displayed.
* The item effect value is available
* The mapcell object is created and displayed.
* Whether or not the cell has been seen.
* Whether or not the cell has a monster or item.

# Grading

Points will be awarded as to how well the classes define objects that match the requirements given in this document. Points can be lost for unprofessional looking code or applications. There are no specific expectations on presentation of your form as long as it is able to show the object behaviors needed and is professional in appearance.

Here is an example of what it might look like:

