PROJECT/ASSIGNMENT: Console Game Application

Sandra H. Jones
Course: CSE 1301
Kennesaw State University

IMPLEMENTATION RECOMMENDATIONS

Notes to assist students with their efforts:

Following are some tips to help students with this assignment. I typically cover these during lecture and include with assignment instructions. Because this assignment takes into account all aspects of the first programming course, the goal is to get the students thinking about solving the problem before they start working on the code.

- Make a plan before you start writing! Use meaningful array and variable names to make creating this program easier.
- COMMENT YOUR CODE!
- It will be helpful to create a numMonstersCaught variable and pass by ref to the method that evaluates a player guess.
- Use a do loop to run the game until all of the monsters are caught.
- You are REQUIRED to write at least two methods. You may write more if you'd like.
- Make sure you give your player good feedback regarding their status - guesses made, monsters caught, etc.
- Feel free to make enhancements to your game!

Suggestions for increasing complexity of the assignment

The assignment above is rather simple, but it includes all of the foundational topics typically seen in an introductory course. As written, it is ideal for a lab or homework assignment. Should you choose to use this as a semester project, it is suggested that the complexity of the program be ramped up a bit to really challenge the students. Following are some suggestions on how this might be done. Note that these suggestions build on the base described above. The concepts tested are the same as those describe above, but the overall problem is more challenging.

1. If the class has covered 2D arrays, consider implementing the game with a 2D array vs. 1D.
2. Add a Player class, with a name field and an array field of Weapons objects equal to or slightly more than the number of Monsters that the game will contain.
3. Add a Weapon class that has a name field and a field for the ability of the weapon. e.g., Sword defeats strength <= 10. Write a method in Main that automatically creates an array of weapons of your choosing from which the player can select.
4. Add strength field to the Monster class.
5. Write a method of the Monster class to increase strength during each player turn that it is not found. The initial value of strength and the amounts increased is up to the student, but each Monster should be different.

6. Write a Method of the Monster class to doBattle() when he is found. During the battle, the player should have to think about the choice of Weapons he will use to fight and defeat the Monster. Display the current list of Player weapons to help him decide. Write the method so that it determines the winner, based on which is stronger – the weapon or the monster. Either the Monster will be defeated and captured, or the player loses the battle.

   Have the student use some creativity here with game play and progression. For instance, one approach might be that if the player loses the battle, the Monster is placed in a new random position in the array, and the user can choose a new Weapon. Another might be if all of the Monsters are not captured when the Player is out of weapons, the game is over and the Player loses. Having the student build in their own dynamics and mechanics will give him ownership of his own game and encourage originality.
7. When the game starts, create the player instance based on user input. Write a method that presents the player with a collection of weapons from which they can chose and a schedule of what strengths it can defeat. e.g. Sword – defeats Monsters with strength of 10 or less.
   Here you might also list the potential Monsters in the array and their strength. e.g., "Orcs start with a strength of 8 and earn 3 points for each turn they remain hidden" so that the player might strategize.

Notes for the instructor:

As students progress through an introductory programming course, I have found that while they may go to lab assignments and successfully create topical programs, they sometimes they have trouble "putting it all together" in a more robust, fully functioning way. I find that it helps to discuss this assignment from an algorithmic perspective before they get started.

In my own class, I have found that additional explanation is usually needed for:

- Creating and working with collections of objects
- Understanding arrays as reference types
- The use of parallel arrays in the game

If the more complex problem is used, challenges often occur with:

- The notion of aggregation
- How instances of different classes can interact with one another
- Using creativity in our programming assignments, but focusing on how to implement those ideas most efficiently and effectively.

One final note. This game was created as a reinforcement of fundamental programming topics before moving on to creating games using a game engine. The "theme" of the game might be easily changed. For instance, it might be miners looking for precious gems; it might be centered around pirates looking for treasure; or one might even search for some piece of hidden data and be required to solve logic puzzles when found. You might even ask your students to gear the game to a particular audience, or develop their own idea of how the game may function.