SUBMISSION INSTRUCTIONS

Send ONE *.zip compressed file containing all your *.java files.
Submit your assignment through blackboard (submit assignment link).
Name your assignment file (the zip file) Assignment_Final_<student name>.
For example; Assignment_Final_Neilsen_Janeil.zip

GRADING MATRIX

The following provides you with feedback on each area you are graded on for the assignments in this course.

Trait	90% –- 100%	70% –- 89%	50% - 69%	0% - 49%
Specifications	The program works and meets all of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
Readability	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Reusability	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability.
Documentation	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Delivery	The program was delivered on time.	The program was delivered within a week of the due date.	The code was within 2 weeks of the due date.	The code was more than 2 weeks overdue.
Efficiency	The code is extremely efficient without sacrificing readability and understanding.	The code is fairly efficient without sacrificing readability and understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.

ASSIGNMENT BACKGROUND

This assignment builds upon all the material learned in this course.

ASSIGNMENT PREPARATION

In order to prepare for this final assignment question you should have attempted:

- complete 5 exercises from all chapters (odd questions)
- completed assignments 1 to 4

ASSIGNMENT QUESTION – Game Zone

Create a game called Sheep Herder. The idea of the game is to herd the sheep (find) before the sheep are eaten. Simply put, the user chooses spots in a grid and if it is a sheep, the sheep was herded.

In the game there will also be a dog and a wolf. If found, the dog will help in two ways:

1. Give the user an extra turn.
2. Fight the wolf if the wolf attacks you.

If found, the wolf will attack you and you will lose unless you already found the dog.

All animals have a random strength value (str). This will come in to play when the dog defends you from the wolf or the wolf bumps into the dog. Say the Dog str = 10 and the wolf's str = 8. Well your dog would win and survive with only 2 left over and the poor wolf dies. But what if it was vise versa? Your dog would have died and the wolf survives with str = 2. But happily you still survive in either scenario.

Now the game starts and the computer creates a 5x5 grid and randomly chooses a coordinate to put the sheep, dog and wolf. Normally the user cannot see where they are but for this assignment I want to be able to see all of them for testing purposes. Make it similar to the example provided below.

In each turn the user gets to choose a coordinate and then the wolf moves. You get to decide how smart that wolf is. Does it move by smell or random? But if that wolf hits the sheep, the sheep is no more and the game ends.

The game has to be similar to what I said but you can change it up to make it better if you wish. If I see a smart wolf (AI) you would get better marks… Lastly add levels.

Be sure to use JOptionPane, Switches, Loops, StringBuilder, and arrays. There should be at least three classes created. I created six classes to create this game (SheepHerder.java, Player.java, Grid.java, Coordinate.java, Call.java and Animal.java). Make it user friendly with a smart AI and OOP (meaning almost no code in the main method)!

You will use this program again in the Intermediate Java course if you take it, so don't delete it.