Team 3 WSS project
Tavina Chen, Aryan Parwal, Chi Hao Nguyen, Toni Liang, Viet Ha, Andres Vargas

Contributions: https://github.com/hcnguyen1/CS3560Project/tree/main
Tavina worked on vision and brain diagrams and implemented vision, brain, path, and cost classes.
Viet worked on brain diagrams and implemented some brain strategy code.
Aryan worked on trader, bonus diagrams and implemented trader and bonus related classes.
Andres worked on trader, bonus diagrams and structured code for trader and bonus related classes.
Toni worked on difficulty, terrain, and map diagrams and implemented code for related classes.
Chi worked on difficulty, terrain, and map diagrams and implemented code for related classes.

Project Status:

Most features of the WSS are implemented. A few features are not yet implemented. One is repeating bonuses. The other is proper resource depletion when the player stays. Currently, due to using integers, the ceiling of ½ the normal rate is used, so if the cost was 1, instead of ½, the depletion is still 1. If the cost was 2, the depletion should be 1.

Project Specifications:

Our project has 5 types of terrain: deserts, forests, lakes, mountains, and plains. Our traders are split into 3 types, one that specializes in food, one that specializes in water, and one that specializes in gold. Each trader decides with a 50/50 coin flip whether or not to accept or counter an offer. The player has 3 chances to trade with the trader before that interaction is locked. We also have 4 different vision and brain personality types which are randomized upon spawning the player for a total of 16 different combinations. Each brain personality type is determined by their thresholds, in other words, how much of a resource they believe is acceptable before they are considered low on a certain resource. The hoarder wants to keep as many resources as possible, so its threshold is at ⅔ the maximum allowed amount. The adventurer's threshold is at ½ the max amount except for gold; their gold threshold is at 1 because they don't care about it. The risk taker's threshold is 1 for every resource so they have a high chance of dying when resources are low, and default is ⅓ the max amount.

OOP Design Principles and Applications:

A place where we use hierarchy can be seen with the relationship between the Vision classes (Vision.java, CautiousVision.java, FarSight.java, FocusedVision.java, KeenEyedVision.java). The latter 4 are all types of visions. The vision classes also demonstrate inheritance because they each extend the vision class above it in the hierarchy –using and overwriting some of its parent methods. Polymorphism is seen in the Player.java class' private variables where the player has a vision object. That vision object can become any one of the 4 vision classes once the player is spawned. Encapsulation can be seen in Brain.java's makeMove() method; how the brain determines what move to take is completely hidden away and it only needs to be called for the player to move. Abstraction can be seen in the Trader.java class and vision subclasses; they have only the essentials for completing their tasks. Lastly, a place where we have really good comments can be seen in App.java.

The principles of object oriented design helped us coordinate and be more efficient in implementing our code. By preplanning the relationships outlined above, many lines of repetitive code were no longer needed and a clear separation between parts was made. This separation helped with debugging and splitting the work among teammates. The diagrams helped us coordinate our method names so that multiple members can be working on their parts simultaneously for higher efficiency.