Design:

Design: Goal is to survive until finding chest with map in it

Note: each space has flag that when turned on will show the space on printed board after user explores that space

- Space class (abstract Base class)
    - Circular 2d Linked list structure to represent board spaces
    - 4 space pointers, (top, right, left, bottom)
- Wood space
    - Adds random amount of wood to backpack between 2 and 8
        - Used for shelter or fire, or spear
- Grass space
    - Build Shelter – gets added to map, automatically return to closest shelter when gets dark/end of day
    - Build Fire – 50% chance hypothermia without fire at night
        - Goes out when leaving space
    - GetShelter – returns true if space has shelter
- Pond space
    - Water is not clean, 50% chance of sickness
        - Sickness = 2 days without moving
    - Drink water
    - getSickness return true if water made person sick
- Creek space
    - Water is cleaner, 5% chance of sickness
        - Sickness = 2 days without moving
    - Drink water
    - getSickness return true if water made person sick
- Bush space
    - Contains berries to eat
    - eatBerry
        - updates food counter random value from 10 to 50%
- Cabin space – creepy old cabin in woods
    - Contains map that leads you out of woods and to safety
- Character class
    - Creates character
    - Updatewater
    - updateFood
- Game class
    - Creates character with 100% water and 100% food
        - Start with hatchet to cut wood
    - Creates backpack container to hold wood
    - printBoard
    - makeBoard – with random spaces, but cabin must be one of them

- updateWater()
- decreaseTime (12 hours per day daylight, 5 days to find map = 60 moves)
- returnShelter- returns character to nearest shelter
- printHourlyResults – print each move, next option
- printDailyResults
- endOfDay
  - updateWater
  - updateFood
  - decDay

<u>Test Cases:</u>

Move menu test cases

Test case	Input Values	Driver Function	Expected Outcome	Observed Outcome
Function a not an unsiged integer	A, 1.5, -4	ValidInt()	Display error, repeat options to user	Display error, repeat options to user
Integer not a valid option	5	ValidInt()	Display error, repeat options to user	Display error, repeat options to user
Integer a valid option	1	validInt(), Board class moveCharacter()	Print moves character pointer up	Print moves character pointer up
Integer second option	2	validInt(), Board class moveCharacter()	Print moves character pointer down	Print moves character pointer down
Integer third option	3	validInt(), Board class moveCharacter()	Print moves character pointer left	Print moves character pointer left
Integer fourth option	4	validInt(), Board class moveCharacter()	Print moves character pointer right	Print moves character pointer right

Yes/No interaction prompts

Test case	Input Values	Driver Function	Expected Outcome	Observed Outcome
Function a not a valid option	A, 1.5, -4	ValidStr()	Display error, repeat options to user	Display error, repeat options to user
Enters a char ‘Y’ or ‘y’	Y, y	ValidStr(), enterSpace()	Validates and enters space to interact	Validates and enters space to interact
Enters a char ‘N’ or ‘n’	N, n	validStr(), enterSpace()	Validates and enters space to interact	Validates and enters space to interact

Reflection:

This was definitely the most enjoyable project in terms of freedom and being able to do pretty much anything I wanted. Of course, that led to some problems early on. My original design did not contain a board class, which I soon realized I was making circular declarations and needed a class to contain the spaces, and that making a space that contained other spaces was overly complicating things and was near impossible to keep straight. This was the biggest problems with errors and to fix those I just added the board class to control my space objects and their pointers.

I also decided to simplify from the original design to make fewer spaces and possibilities to help the grader figure out the game and not take forever trying to grade my project. For instance, I removed having a chance of the person getting sick from drinking water, causing me to need less functions and making it less complicated to win. This caused me to remove the pond class, since it was something that was not needed after removing the chance of the player getting sick, since they could just go to a creek space to drink more water. This also including having to keep the players food levels up, which caused me to delete the bush class completely.

Another design change was instead of having a character class, I made a space pointer representing the character to keep track of its spacing and had a backpack and its water level held in the game class. This allowed me to write functions that were easier to keep track of and allowed me to not have nested functions calls using the board, space, and character functions in a single declaration, which I thought would make the code cleaner.

Lastly, the final major change that was made was to add keys and a car to find in order to leave the woods. This allows the player to search for a key while surviving and adds an objective other than just surviving for a dedicated amount of time. I think this made sure I was following the project guidelines and that the player had to interact with different spaces in order to win. I also added matches to start fires with in the same location as the car key.

Overall, these changes were to simplify the overall game and make it easier to win, but the final project is completely different than the original plan. I think this mostly has to do with the amount of freedom in design. Before we had set classes and files to get a very good idea on what we would need, but in this one my plan did not cover everything I would need and needed serious adjustments while writing the code, which will improve as I create more projects with a lot of freedom and room for creativity.