**Settled Ground**

*Lucas Lehman CS121 Final Project Documentation*

***Summary***

Settled Ground is a simple text-based resource-collecting dice game. It will be played through a command line with 2 players. The players’ goal is to get to 25 survival points by collecting resources, building structures, and avoiding the pesky bandit.

The purpose of this project is to demonstrate concepts learned in CS-121, and to create something of my own.

My biggest goal is to get the base game working. After that I will definitely need to tweak the point system and bandit probabilities to make the game more balanced and fun. One I get that working I would like to add some more features/actions to the game to make it a bit more intricate. I would also love to try and program a computer player and run simulations to see which strategies work best.

***Game Rules***

* **Players:** 2
* **Goal:** Be the first player to reach 25 survival points
* **On each turn**
  + Roll the die and collect your resource
  + 1 action is allowed per turn:
    - *Continue*– nothing happens
    - *Barter* – use barter token to add one of any other resource to your inventory
    - *Stash* – stash 1 resource to earn 1 point (any item except barter tokens can be stashed. Stashing a resource removes it from inventory)
    - *Build* - use resources to build a structure for points
* **Die Values:**
  + 1: Bandit (if rolled, a coin is flipped for a *50% chance to lose all resources)*
  + 2: Food
  + 3: Water
  + 4: Wood
  + 5: Stone
  + 6: Barter token
* Structure build costs/points:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Wood | Stone | Food | Water | Barter Token |
| **Fire Pit** (*3pts*) | 1 | 1 |  |  |  |
| **Shelter** (*5pts*) | 1 |  | 1 | 1 |  |
| **Cabin** (*10pts)* | 2 | 2 | 1 | 1 |  |
| **Outpost** (*10pts*)  (*Barter tokens now worth 2 resources when used for bartering)* | 1 | 1 | 1 | 1 | 1 |

***Sample Run***

. . . Lucas rolls a 6

Lucas collects 1 Barter Token

Lucas’s Inventory

- 1 Points

- 1 Barter Token

Actions

1.) Continue

2.) Stash

3.) Barter

4.) Build

5.) Rules/Instructions

6.) Exit Game

Please Enter (1-6): 3

Barters for one resource of your choice

1.) Food

2.) Water

3.) Wood

4.) Stone

Please Enter (1-4): 4

Lucas collects 1 Stone

. . . Levi rolls a 5

Levi collects 1 Stone

Levi’s Inventory (Now jumping to late game)

- 5 points

- 1 Food

- 1 Water

- 1 Wood

- 2 Stone

- 1 Barter Token

Actions

1.) Continue

2.) Stash

3.) Barter

4.) Build

5.) Rules/Instructions

6.) Exit Game

Please Enter (1-6): 4

Structures

1.) Fire Pit (3pts) (1 Wood, 1 Stone)

2.) Shelter (5pts) (1 Wood, 1 Food, 1 Water)

3.) Cabin (10pts) (2 Wood, 2 Stone, 1 Food, 1 Water)

4.) Outpost (10pts) (1 of each item)

Please Enter (1-4): 4

Levi Builds an Outpost for ten points. Barter Tokens can now be traded in for 2 resources

Welcome to Settled ground!

…blah blah blah

Game Menu

1.) Start Game

2.) Rules/Instructions

3.) Exit Program

Please Enter (1-3): 1

Player Selection

Name of Player 1: Lucas

Name of Player 2: Levi

A coin was flipped, and Lucas will roll first

. . . Lucas rolls a 3

Lucas collects 1 Water

Lucas’s Inventory

- 0 Points

- 1 Water

Actions

1.) Continue

2.) Stash

3.) Barter

4.) Build

5.) Rules/Instructions

6.) Exit Game

Please Enter (1-6): 2

What item would you like to stash?

1.) Water

Please Enter the item number: 1

You stashed 1 Water and earned one Point!

. . . Levi rolls a 1 and encounters a Bandit...

He gets lucky and escapes with his inventory

Levi’s Inventory

- 0 points

Actions

1.) Continue

2.) Stash

3.) Barter

4.) Build

5.) Rules/Instructions

6.) Exit Game

Please Enter (1-6): 1

***Classes***

* **Player**
  + Holds the inventory array, the main data structure of this program
    - Each index in the inventory array represents a resource, with the last few indices for structures like Outpost or Shelter and Points
    - Resource types will be defined as constant integers in a different file for easy array access (player1.addResource(WOOD))
  + Each player has a name property, set by the user at the beginning of the program
  + printInventory()
    - prints out the players inventory by iterating through inventory[]
    - only prints if the value is not zero
  + addResource(resource)
    - adds 1 to the specific resource in inventory[]
    - called right after each dice roll
  + stashResource()
    - prompts for which resource to stash – checks to make sure it’s not zero
    - subtract one from that resource and add one to points
    - called from the action menu
  + useBarter()
    - checks for barter token, if not zero than subtracts 1 from the barter tokens and prompts for which resource to add
    - also checks for an Outpost. If there is one or more outpost a prompt for two resources will be given and added to inventory
  + clearInventory()
    - changes all resources to zero except points
    - called if the bandit is encountered
  + build()
    - stores an array of int equal in length to inventory for each structures specific build cost
    - prompts user for which structure they’d like to build
    - calls canBuild to check if possible
    - subtracts the build cost from the inventory, adds the amount of points earned and adds 1 to the structure built
  + canBuild(structureCost[])
    - Boolean method checking to see if a structure can be built
    - Loops through arrays making sure each value in inventory is greater than or equal to the structure cost
    - Returns true or false
* **Game**
  + Contains two instances of Player
  + main()
    - Instantiates game
    - Calls start()
  + start()
    - displays gameMenu() and exits, prints rules, or starts game
    - once the game starts, call namePlayers()
    - flip a coin (flipCoin()) to see who goes first
    - Then it loops alternating takeTurn() until one player has 25 or more points
  + gameMenu()
    - shows options and returns a string with player response
  + selectPlayers()
    - asks users for names and stores them into their name attribute
  + takeTurn()
    - rolls a die (rollDie())
    - if bandit, flip coin (flipCoin()) and see if inventory gets removed
    - for everything else it adds the resource to players inventory
    - print out their inventory
    - display actionMenu() and follow the users choice calling methods like add/stash resource, build, etc.
  + actionMenu()
    - shows options and returns string with player input
  + printRules()
    - prints detailed rules of the game with structure costs and items all explained
  + rollDie()
    - returns random int 1-6
  + flipCoin()
    - returns random int 0-1
    - \*\*actually probability might not be 50-50 depending on how annoying the bandit ends up being
* **Inventory Constants**
  + declares an integer value for each resource/points/structures so that both game and player can import them and make working with resources clearer than trying to remember array indices
  + const int FOOD = 0
  + const int WATER = 1
  + const int OUTPOST = 7 etc.

***A screenshot of a diagram

Description automatically generated***

***Milestones***

* get UML approved
* Write InventoryConstants
* Build basic Player class
* Write every Player method and test individually
* Write basic Game class
* Write the Menus
* Write rollCoin and rollDie
* Write selectPlayers
* get one takeTurn working
* write top level start() control
* test game thoroughly

***Stretch Goals***

* tweak structure costs, coin probability, and win condition to balance game and try to make it fun
* add the ability to save to a file (serialization, or simply printing inventory and name?)