Andrew Kallmeyer

Professor Bolotin

Object-Oriented Design

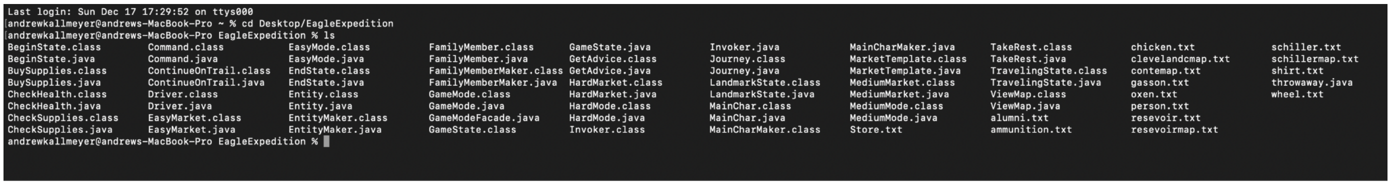
17 December 2023

Eagle Expedition

Throughout the course of this class, when thinking about an idea for my final project, I oftentimes tried to imagine myself in a scenario where object-oriented design concepts would carry the most utility. Of course, the purpose of these design elements is to be able to create concise, efficient code that limits the hindrance of thousands of variables and lines of code. When thinking of programs that would present a lot of moving parts for a programmer, I thought of old-school video games. Their constant state progressions, health meters, and inventory items had to be a challenge for any engineer coding in an object-oriented language. The first game that came to mind was *Oregon Trail*, the desktop game born out of the 1970s and eventually developed into gaming console software, which to me epitomized the template for old-school games, and allowed a lot of room for creative influence. All this being said, I eventually came to the conclusion that I would build a Boston College version of *Oregon Trail*.

Basic Game Element Demonstration

I. <u>List of classes and other files</u>

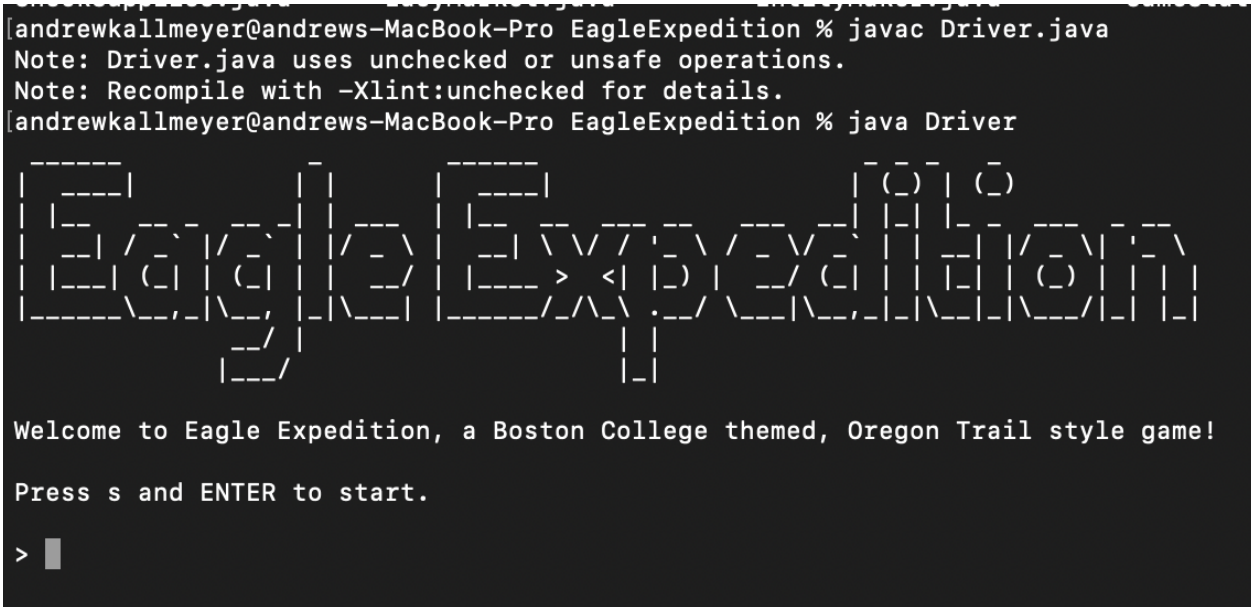


```
Last login: Sun Dec 17 17:29:52 on ttys000
[andrewkallmeyer@andrews-MacBook-Pro ~ % cd Desktop/EagleExpedition
[andrewkallmeyer@andrews-MacBook-Pro EagleExpedition % ls
BeginState.class        Command.class           EasyMode.class          FamilyMember.class      GameState.java          Invoker.java            MainCharMaker.java      TakeRest.class          chicken.txt             schiller.txt
BeginState.java         Command.java            EasyMode.java           FamilyMember.java       GetAdvice.class         Journey.class           MarketTemplate.class    TakeRest.java           clevelandcmap.txt       schillermap.txt
BuySupplies.class       ContinueOnTrail.class   EndState.class          FamilyMemberMaker.class GetAdvice.java          Journey.java            MarketTemplate.java     TravelingState.class    contemap.txt            shirt.txt
BuySupplies.java        ContinueOnTrail.java    EndState.java           FamilyMemberMaker.java  HardMarket.class        LandmarkState.class     MediumMarket.class      TravelingState.java     gasson.txt              throwaway.java
CheckHealth.class       Driver.class            Entity.class            GameMode.class          HardMarket.java         LandmarkState.java      MediumMarket.java       ViewMap.class           oxen.txt                wheel.txt
CheckHealth.java        Driver.java             Entity.java             GameMode.java           HardMode.class          MainChar.class          MediumMode.class        ViewMap.java            person.txt
CheckSupplies.class     EasyMarket.class        EntityMaker.class       GameModeFacade.java     HardMode.java           MainChar.java           MediumMode.java         alumni.txt              resevoir.txt
CheckSupplies.java      EasyMarket.java         EntityMaker.java        GameState.class         Invoker.class           MainCharMaker.class     Store.txt               ammunition.txt          resevoirmap.txt
andrewkallmeyer@andrews-MacBook-Pro EagleExpedition % █
```

- File entitled “Driver.java” refers to main class, will be used to compile and run rode.
- Files including “State” or “Journey” (GameState.java, Journey.java, BeginState.java, TravelingState.java, LandmarkState.java, EndState.java) refer to classes implementing State pattern.
- Files describing activity statements (BuySupplies.java, CheckHealth.java, CheckSupplies.java, ContinueOnTrail.java, GetAdvice.java, TakeRest.java, ViewMap.java) as well as Command.java and Invoker.java refer to classes implementing Command pattern.
- Files Entity.java, EntityMaker.java, MainChar.java, MainCharMaker.java, FamilyMember.java, and FamilyMemberMaker.java refer to classes implementing Factory Method pattern.
- Files including “Market” (EasyMarket.Java, MediumMarket.java, HardMarket.java, MarketTemplate.java) refer to classes implementing Template pattern.
- Files including “Mode” (GameMode.java, EasyMode.java, MediumMode.java, HardMode.java) refer to classes implementing Facade Pattern.

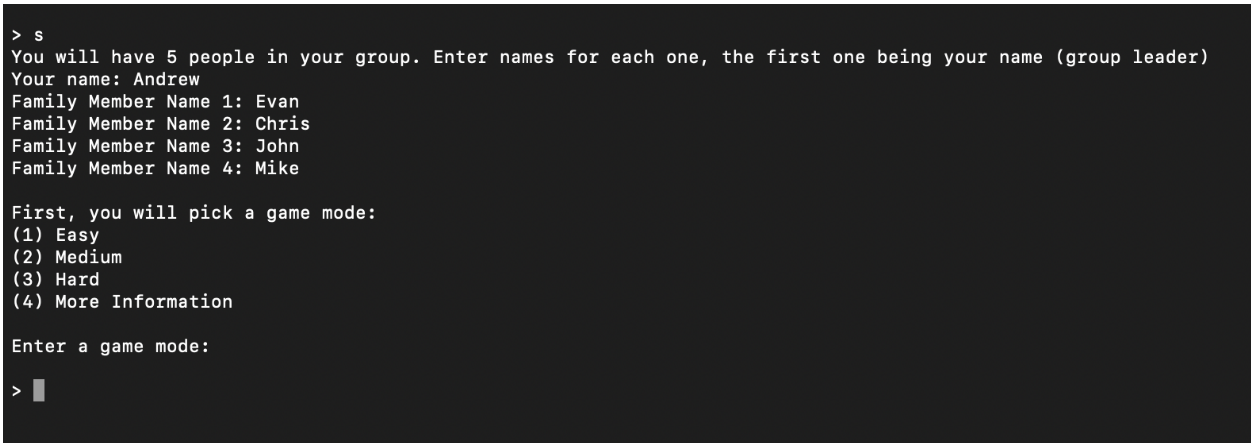
- Files ending in ".txt" refer to text files containing ASCII art used to portray various elements of the game.

II. Beginning Game



```
[andrewkallmeyer@andrews-MacBook-Pro EagleExpedition % javac Driver.java
Note: Driver.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
[andrewkallmeyer@andrews-MacBook-Pro EagleExpedition % java Driver
 ______             _        ______                     _ _ _   _
|  ____|           | |      |  ____|                   | (_) | (_)
| |__   __ _  __ _| | ___  | |__  __  ___ __   ___  __| |_| |_ _  ___  _ __
|  __| / _` |/ _` | |/ _ \ |  __| \ \/ / '_ \ / _ \/ _` | | __| |/ _ \| '_ \
| |___| (_| | (_| | |  __/ | |____ >  <| |_) |  __/ (_| | | |_| | (_) | | | |
|______\__,_|\__, |_|\___| |______/_/\_\ .__/ \___|\__,_|_|\__|_|\___/|_| |_|
              __/ |                    | |
             |___/                     |_|

Welcome to Eagle Expedition, a Boston College themed, Oregon Trail style game!

Press s and ENTER to start.

> 
```

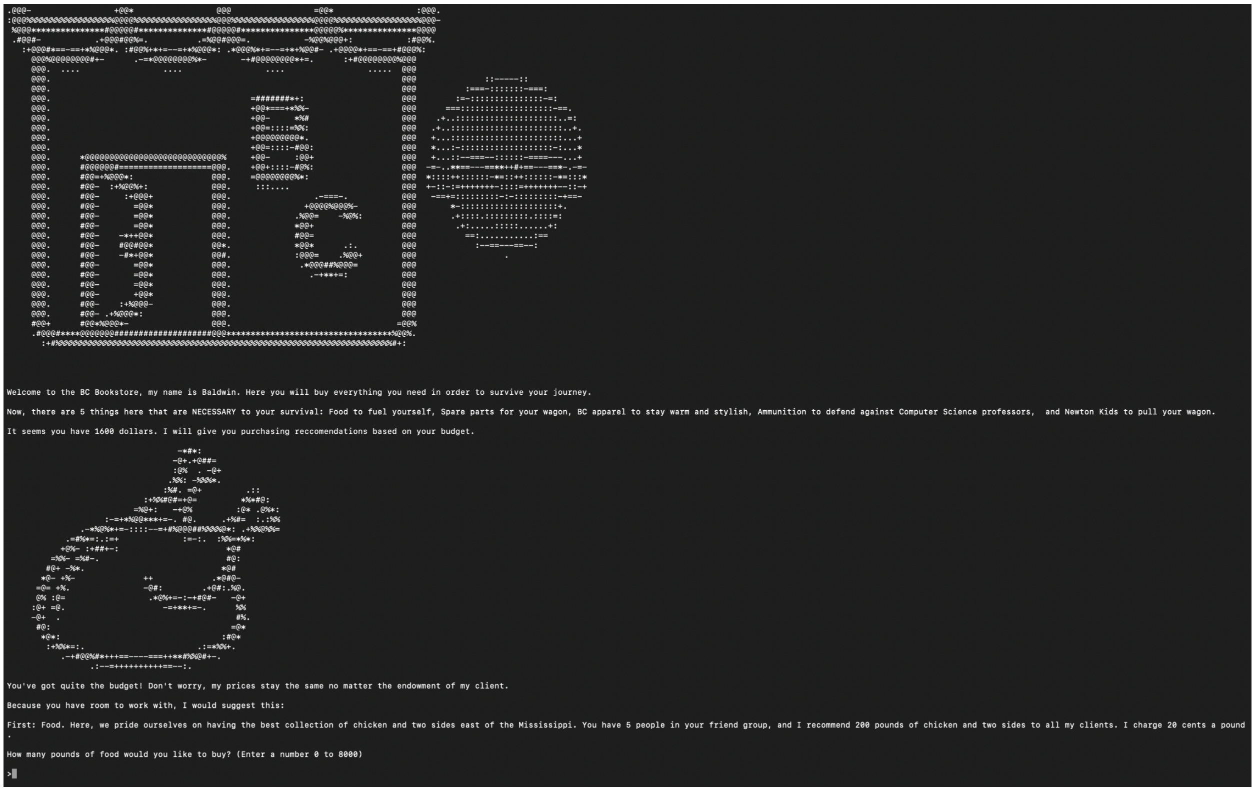
- Enter "javac Driver.java" and then "java Driver" and the program should begin to run.
- Title ASCII art will appear, prompting user to input "s" to begin. It will prompt you again if you input anything other than "s" or "S".



```
> s
You will have 5 people in your group. Enter names for each one, the first one being your name (group leader)
Your name: Andrew
Family Member Name 1: Evan
Family Member Name 2: Chris
Family Member Name 3: John
Family Member Name 4: Mike

First, you will pick a game mode:
(1) Easy
(2) Medium
(3) Hard
(4) More Information

Enter a game mode:

> 
```

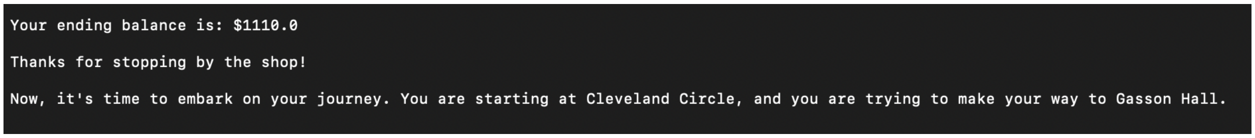
- Program will prompt user for names for the group leader (user name) and then names of 4 other people. These are required fields, and program will not let you pass unless you enter all Strings.
- Program will then prompt user to pick a difficulty, or get more information. Game modes differ only on amount of money allotted to the user at the start of the game (this is what More Information will tell you).

III. Market



```
Welcome to the BC Bookstore, my name is Baldwin. Here you will buy everything you need in order to survive your journey.

Now, there are 5 things here that are NECESSARY to your survival: Food to fuel yourself, Spare parts for your wagon, BC apparel to stay warm and stylish, Ammunition to defend against Computer Science professors,  and Newton Kids to pull your wagon.

It seems you have 1600 dollars. I will give you purchasing reccomendations based on your budget.

You've got quite the budget! Don't worry, my prices stay the same no matter the endowment of my client.

Because you have room to work with, I would suggest this:

First: Food. Here, we pride ourselves on having the best collection of chicken and two sides east of the Mississippi. You have 5 people in your friend group, and I recommend 200 pounds of chicken and two sides to all my clients. I charge 20 cents a pound
.

How many pounds of food would you like to buy? (Enter a number 0 to 8000)

>
```

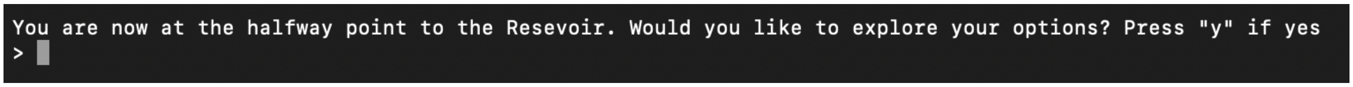
- Program will then take user to the supplies market, which will prompt user for inputs on how many units of each supply type (Food, Spare Parts, Apparel, Ammunition, and Oxen) they would like to purchase.



```
Your ending balance is: $1110.0

Thanks for stopping by the shop!

Now, it's time to embark on your journey. You are starting at Cleveland Circle, and you are trying to make your way to Gasson Hall.
```

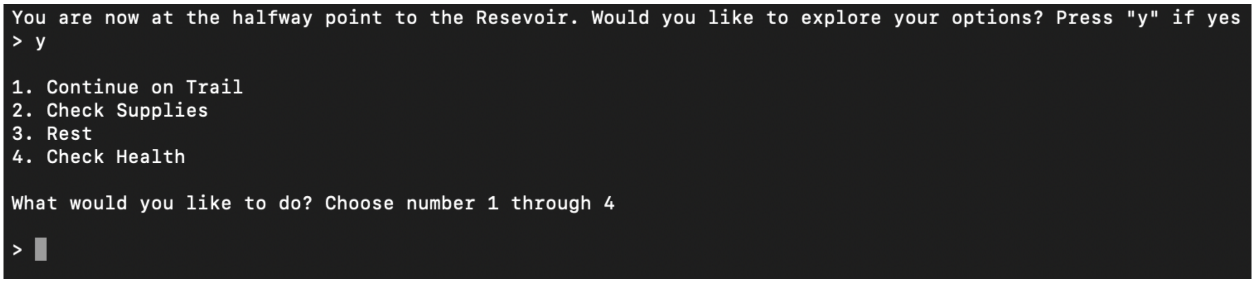
- After purchases, program will tell user final balance and then user will embark on the journey immediately.

IV. State Progression: Traveling States



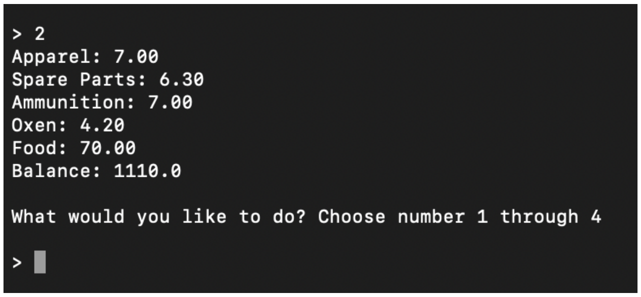
```
You are now at the halfway point to the Resevoir. Would you like to explore your options? Press "y" if yes
>
```

- At all states, users will be prompted to input "y" if they want to explore their options. If input is not "y", program will go to the next state.



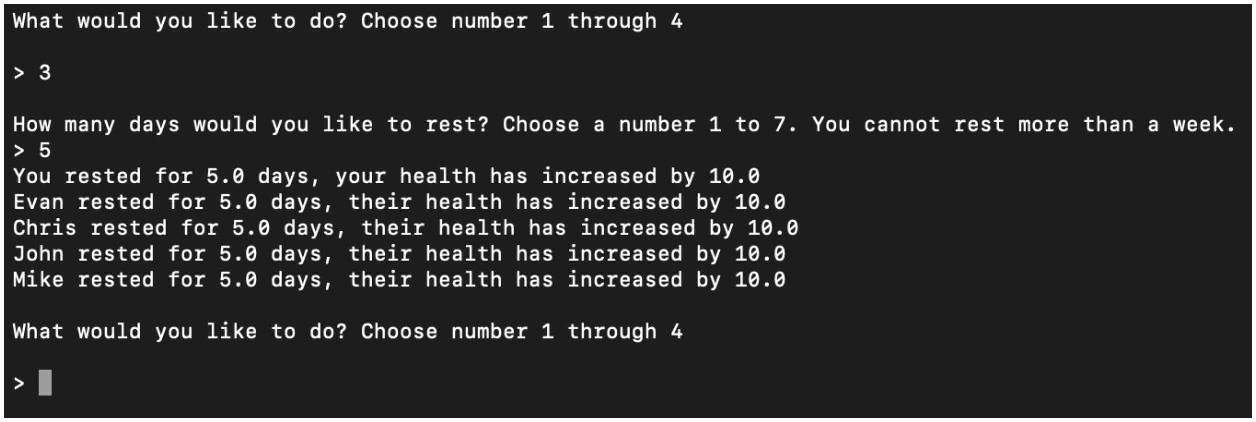
```
You are now at the halfway point to the Resevoir. Would you like to explore your options? Press "y" if yes
> y

1. Continue on Trail
2. Check Supplies
3. Rest
4. Check Health

What would you like to do? Choose number 1 through 4

>
```

- Upon pressing "y", for Traveling States users will be prompted with the 4 options listed above. If input is not one of the listed integers, program will prompt again.



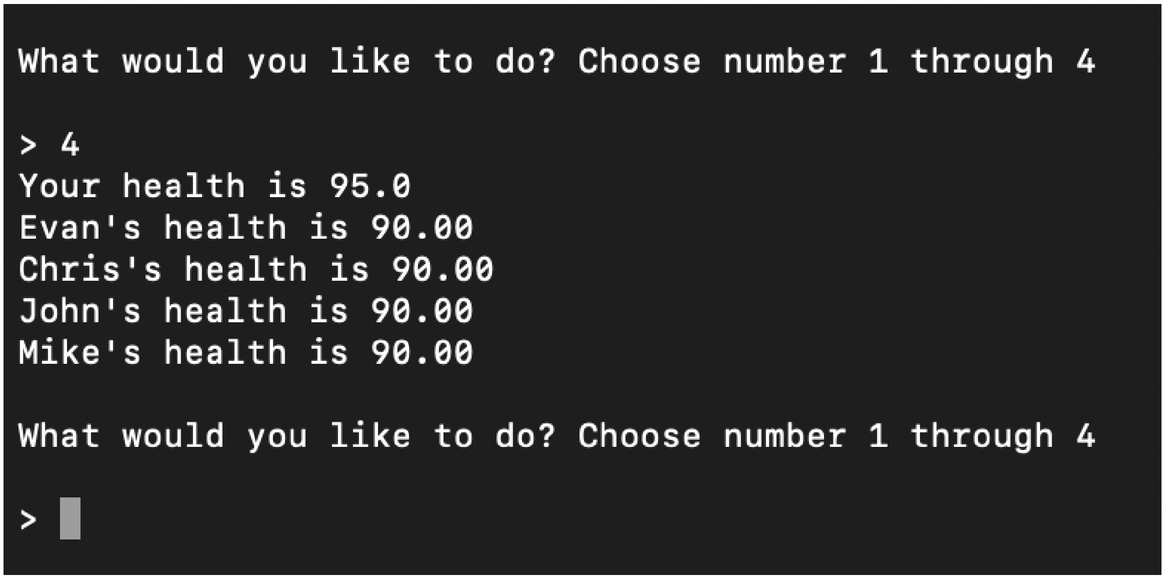
```
> 2
Apparel: 7.00
Spare Parts: 6.30
Ammunition: 7.00
Oxen: 4.20
Food: 70.00
Balance: 1110.0

What would you like to do? Choose number 1 through 4

>
```

- Upon pressing “2”, program will output the current state of each supply type, as well as the money balance. The supply types deplete at a rate of 30% until reaching 3, when they will deplete by 1. Once a supply type reaches 0, the user dies and the game ends.



```
What would you like to do? Choose number 1 through 4

> 3

How many days would you like to rest? Choose a number 1 to 7. You cannot rest more than a week.
> 5
You rested for 5.0 days, your health has increased by 10.0
Evan rested for 5.0 days, their health has increased by 10.0
Chris rested for 5.0 days, their health has increased by 10.0
John rested for 5.0 days, their health has increased by 10.0
Mike rested for 5.0 days, their health has increased by 10.0

What would you like to do? Choose number 1 through 4

> 
```

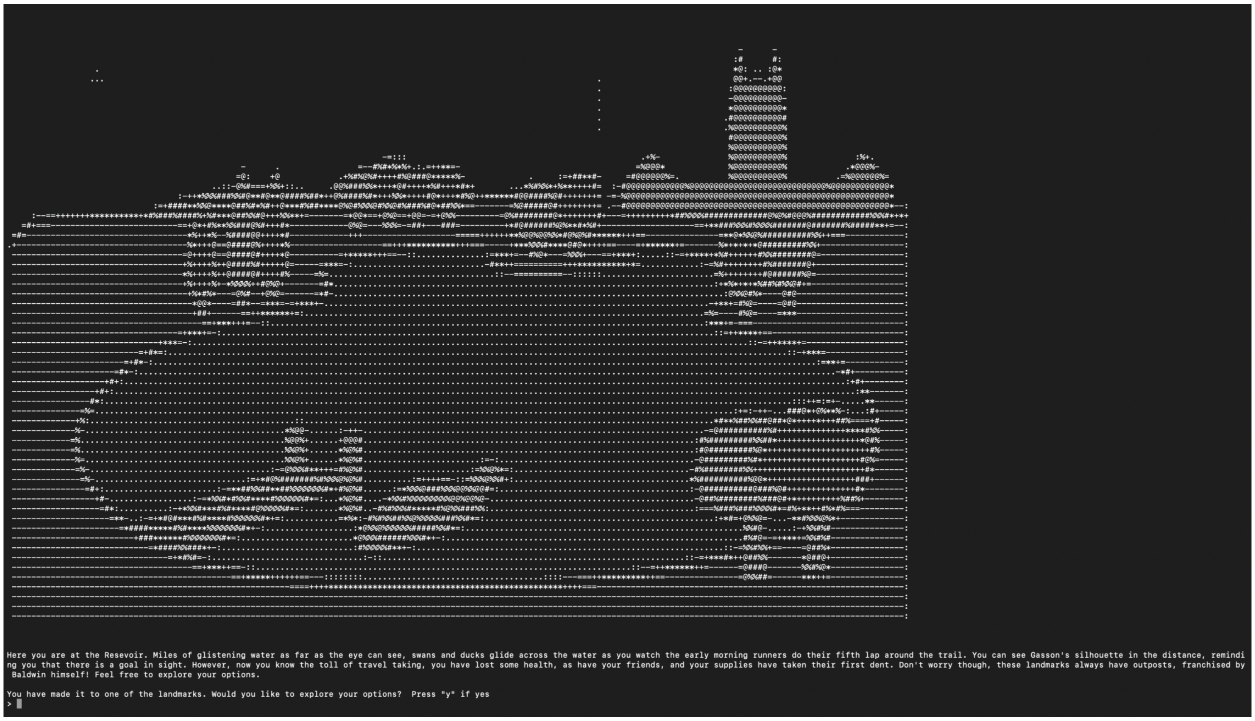
- Upon pressing “3”, program will prompt user to input how many days they would like to rest, although they cannot rest more than 1 week and cannot rest more than once in a given state. Resting increases all users’ health at twice the amount of days.



```
What would you like to do? Choose number 1 through 4

> 4
Your health is 95.0
Evan's health is 90.00
Chris's health is 90.00
John's health is 90.00
Mike's health is 90.00

What would you like to do? Choose number 1 through 4

> 
```

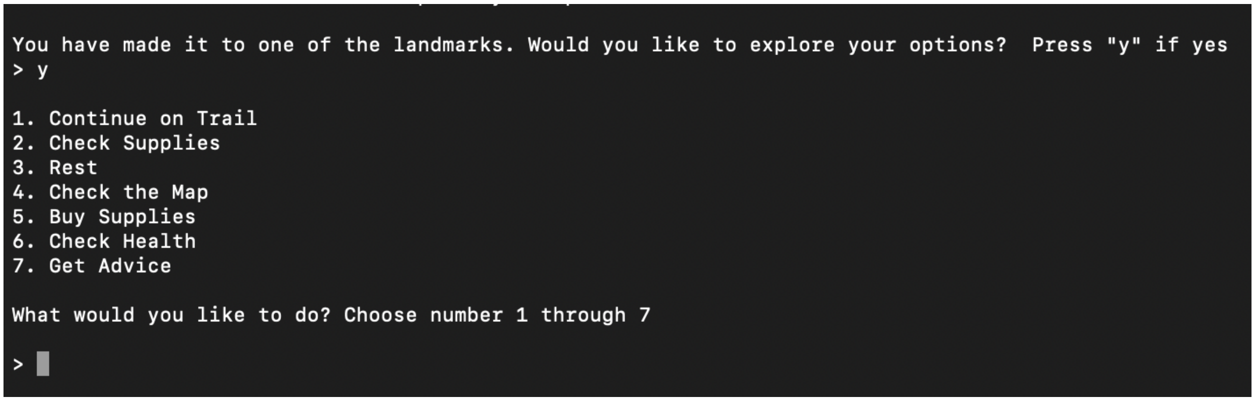
- Upon pressing “4”, program will output the current health of each entity.
- And, when user eventually presses “1”, next state will be presented.

V. <u>State Progression - Landmark States</u>



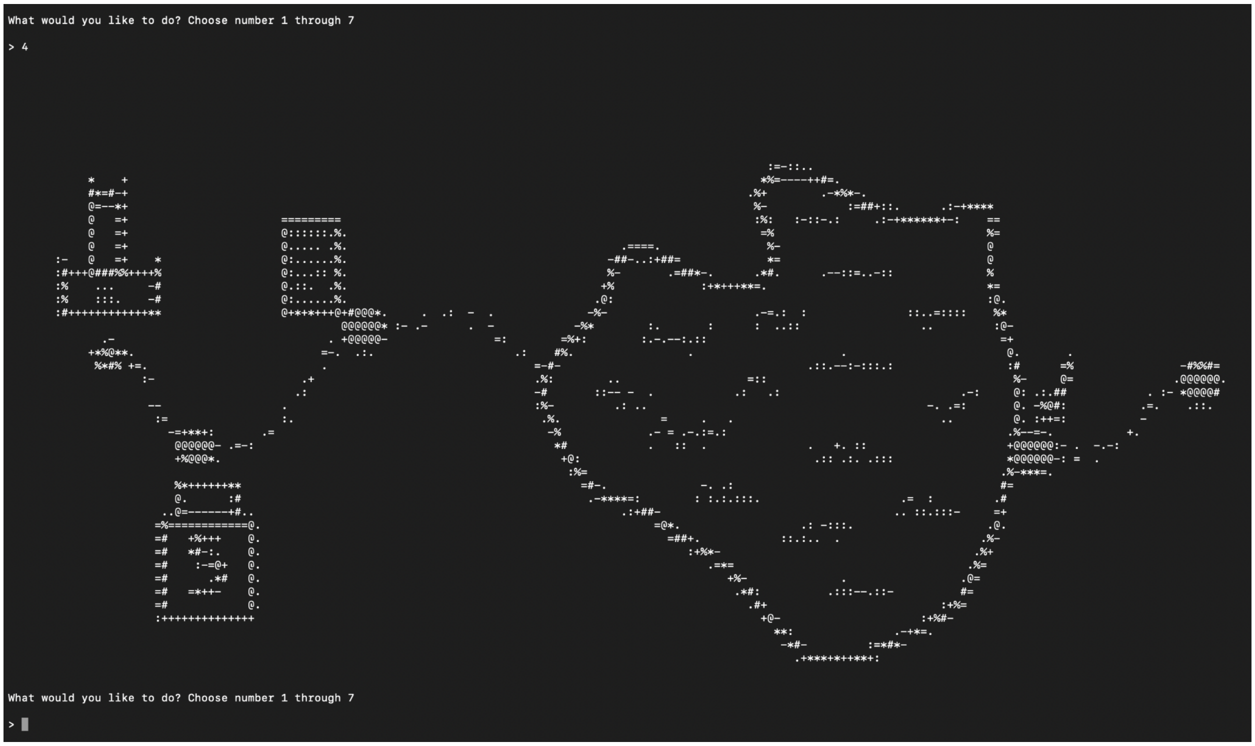
```
Here you are at the Resevoir. Miles of glistening water as far as the eye can see, swans and ducks glide across the water as you watch the early morning runners do their fifth lap around the trail. You can see Gasson's silhouette in the distance, remindi
ng you that there is a goal in sight. However, now you know the toll of travel taking, you have lost some health, as have your friends, and your supplies have taken their first dent. Don't worry though, these landmarks always have outposts, franchised by
 Baldwin himself! Feel free to explore your options.

You have made it to one of the landmarks. Would you like to explore your options?  Press "y" if yes
>
```

- All landmark states are accompanied with ASCII art corresponding to their location.



```
You have made it to one of the landmarks. Would you like to explore your options?  Press "y" if yes
> y

1. Continue on Trail
2. Check Supplies
3. Rest
4. Check the Map
5. Buy Supplies
6. Check Health
7. Get Advice

What would you like to do? Choose number 1 through 7

>
```

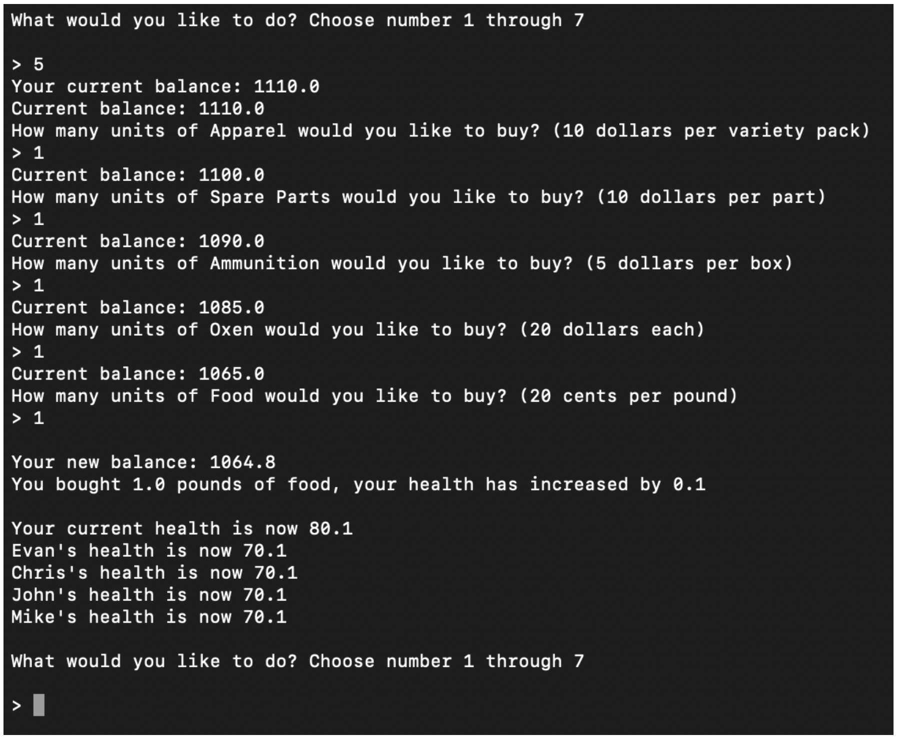
- Landmark states contain same functionality as traveling states with added options.



```
What would you like to do? Choose number 1 through 7

> 4
```

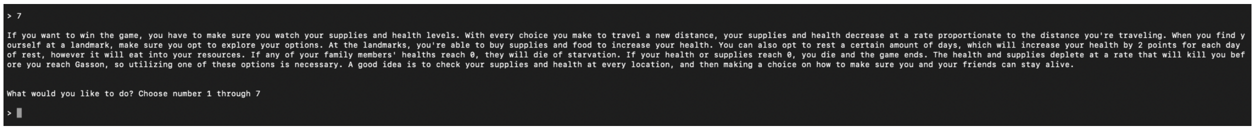
```
What would you like to do? Choose number 1 through 7

>
```

- Upon pressing “4”, user can view a map, which changes every landmark state with the arrow pointing at the current location.



```
What would you like to do? Choose number 1 through 7

> 5
Your current balance: 1110.0
Current balance: 1110.0
How many units of Apparel would you like to buy? (10 dollars per variety pack)
> 1
Current balance: 1100.0
How many units of Spare Parts would you like to buy? (10 dollars per part)
> 1
Current balance: 1090.0
How many units of Ammunition would you like to buy? (5 dollars per box)
> 1
Current balance: 1085.0
How many units of Oxen would you like to buy? (20 dollars each)
> 1
Current balance: 1065.0
How many units of Food would you like to buy? (20 cents per pound)
> 1

Your new balance: 1064.8
You bought 1.0 pounds of food, your health has increased by 0.1

Your current health is now 80.1
Evan's health is now 70.1
Chris's health is now 70.1
John's health is now 70.1
Mike's health is now 70.1

What would you like to do? Choose number 1 through 7

>
```

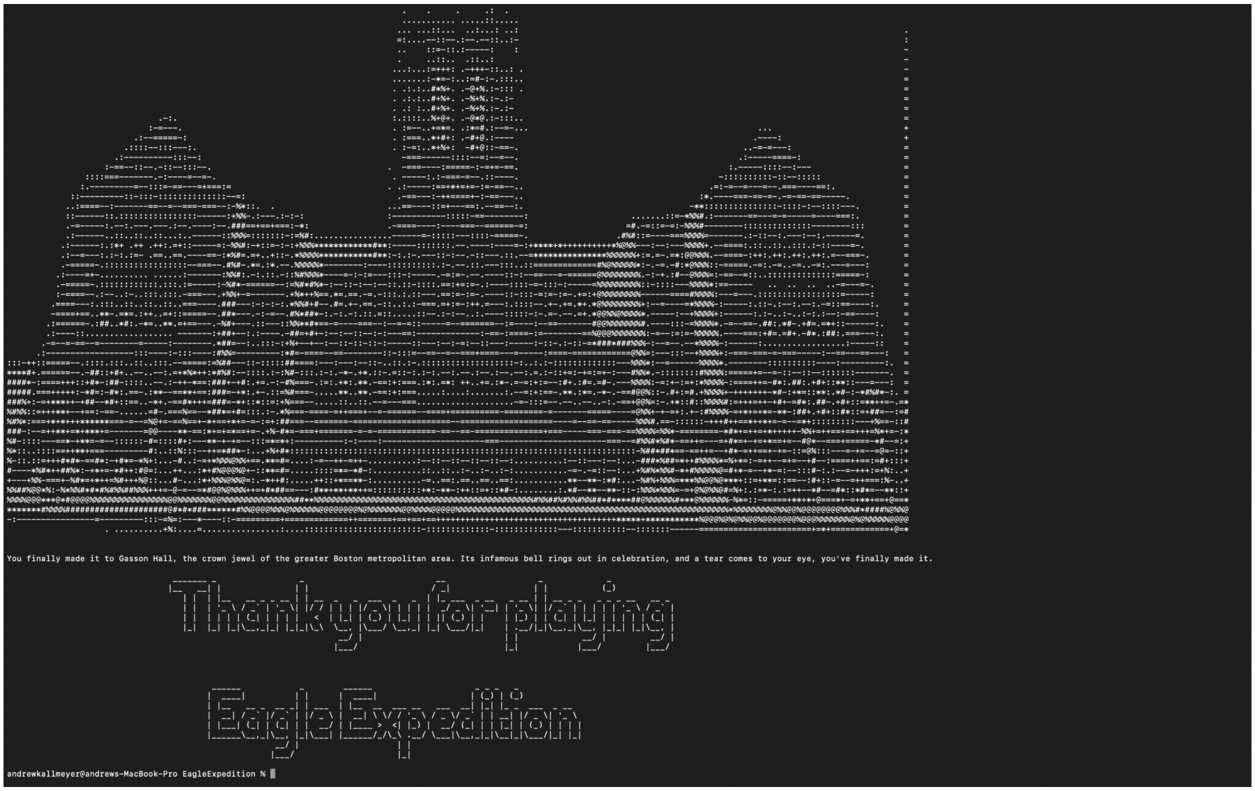
- Upon pressing “5”, user can buy more supplies. Same as the market, user can only buy as much product as they have money. However, at the landmark outposts, you are able to increase your health by buying food. (10% of pounds of food bought).



```
> 7

If you want to win the game, you have to make sure you watch your supplies and health levels. With every choice you make to travel a new distance, your supplies and health decrease at a rate proportionate to the distance you're traveling. When you find y
ourself at a landmark, make sure you opt to explore your options. At the landmarks, you're able to buy supplies and food to increase your health. You can also opt to rest a certain amount of days, which will increase your health by 2 points for each day
of rest, however it will eat into your resources. If any of your family members' healths reach 0, they will die of starvation. If your health or supplies reach 0, you die and the game ends. The health and supplies deplete at a rate that will kill you bef
ore you reach Gasson, so utilizing one of these options is necessary. A good idea is to check your supplies and health at every location, and then making a choice on how to make sure you and your friends can stay alive.


What would you like to do? Choose number 1 through 7

> 
```

- Upon pressing “7”, user can get an advice message on how to complete the game.

VI. <u>End</u>



```
You finally made it to Gasson Hall, the crown jewel of the greater Boston metropolitan area. Its infamous bell rings out in celebration, and a tear comes to your eye, you've finally made it.
```

```
andrewkallmeyer@andrews-MacBook-Pro EagleExpedition % 
```

- If user is able to perform the correct steps to not let their health reach 0 before reaching the end, this success state will be presented.

Rationale for Chosen Design Patterns

- State: This was my first choice of design pattern to use, because I knew it would provide me with the most time and storage saving code. Depending on what state the user is in, different code needs to be output, different options need to be presented, and different functions need to be available to the user. The state pattern allowed me to trim the 10 states I had to only 4, and the ability to outsource them to different files as to not crowd my Driver code.
- Command: Another easy choice for me, command was necessary if I was going to give the user so many options at each state in the game. Command allowed me to organize the code needed to perform all possible command options into their own respective files, as well as interact them with the MainChar.java file which proved useful for getting and setting different aspect quantities and qualities.
- Factory Method: Another distinction I needed to make was that between the main character (the user) and the family or friend group members. Most functions they each contained were common, but some needed to be differentiated, for example the deplete Health function, which imparts more severe health depletion on family members than the main user (as is true in the real game as well). Classifying all the family members as Entity object also allowed me to perform different functions on them as a group in an array or list than separately.
- Template: Template was important because the first real feature of the game is the market, and depending on the game mode the user chose, the market had to present different options for the user based on their budget. Template proved to alleviate a mess of if-else statements, and simple organize different program outputs into files based on

difficulty. Using the MarketRunThrough function in the Driver class also made it much easier to run an instance of a market template with one line in the main function in Driver.

- Facade: One of the integral aspects of the game is also the difficulty level, and as the program explains the only distinction between game modes is the amount of money allotted to the player. Therefore, the facade pattern worked closely with the template pattern to discern the amount of money the user would get based off their game mode input.

UML Diagram

UML Diagram will be in separate file due to size