Use Cases

Menu Controller

- Start Game
- View high scores
- Quit

Game Loop

- Query player
- Decide encounter
- Win or lose

Player

- Take a step
- View character
- Search for money
- Read technical papers
- Quit

Puzzle

- Decide Puzzle
- Display Puzzle

File Accessor

- Read high scores
- Modify high scores - I want to keep track of whether or not a player has finished for each score
- Get puzzle from file

Potential Classes

MainMenu

Vars:

boolean continuePlaying

Functions:

void startGame()

void viewHighScores()

void quit()

Uses:

Player

FileAccessor

Player

Vars:

istream instruction

int stepsRemaining

int time

int intelligence

int money

Functions:

void step()

void searchMoney()

void readPaper()

void viewCharacter()

void quitToMainMenu()

int tallyScore()

boolean isAlive()

Uses:

FileAccessor

EncounterDecider

Puzzle

ValidResponse

EncounterDecider

Vars:

double oddsNothing

double oddsProfessor

double oddsGradStudent

double oddsGruntWork

double oddsGradePapers

Functions:

int decideEncounter() - returns int that represents the type of encounter

Uses:

none

Puzzle

Vars:

char[] puzzle

Functions:

int decidePuzzle()

Uses:

FileAccessor

FileAccessor

Vars:

ifstream inputFile

ofstream outputFile

Functions:

void getHighScore()

void changeHighScore(int score)

void getPuzzle(int puzzle)

Uses:

none

Test Cases

Player

step()

input 1:

set stepsRemaining to 20

output 1:

stepsRemaining == 19

input 2:

set stepsRemaining to -1

output 2:

stepsRemaining == -1

searchMoney()

input 1:

set money to 5

output 1:

money > 5

input 2:

set money to -1

output 2:

money == -1

isAlive()

input 1:

time = 1

money = 1

intelligence = 1

output 1:

isAlive() == true

input 2:

time = 0

money = 1

intelligence = 1

output 2:

isAlive() == false

input 3:

time = 1

money = 0

intelligence = 1

output 3:

isAlive() == false

input 4:

time = 1

money = 1

intelligence = 0

output 4:

isAlive() == false

input 5:

time = 0

money = 0

intelligence = 0

output 5:

isAlive() == false

EncounterDecider

decideEncounter()

input 1:

none

output 1:

execute decideEncounter() 5000 times and keep track of how many times each int from 0-4 is returned. Test passes if the results are in a margin of error compared to the variables in the class that determine the chances of each encounter occurring.