Zachary Parker
3/19/17
COMP 2710

Part 1: Analysis

Use Case 1: Name Entry

Precondition: The program has begun execution.

Description:

- The user will first be prompted to enter their name: "What's your name?"
- The user may enter whatever they wish.

Postcondition: User will be given a welcome message and will be brought to an options menu.

Use Case 2: Options Menu

Precondition: User had entered their name.

Description:

- The user will then be brought to an options menu.
- The options are (1) start a new game, (2) view a tutorial, (3) view top 10 high scores, and (4) quit.
- The user inputs which of the four options they would like to choose.
- If they choose (1), then they begin a new game.
- If they choose (2), then a tutorial will appear.
- If they choose (3), then a list of 10 high scores and the corresponding usernames will appear.
- If they choose (4), then they will successfully quit and the program will terminate.
- If they choose any other value, then they will be told that their input is invalid and will be told to choose a valid input.

Postcondition:

- If (1) was chosen, the user will have begun a new game.
- If (2) or (3) were chosen or if the input was invalid, the user will have been brought back to the same options menu.
- If (4) was chosen, the program will have been terminated.

Use Case 3: In-Game Options

Precondition: The user entered (1) in the Options Menu.

Description:

- If the user chooses to begin a game, then they will frequently encounter a new in-game options menu.
- At the beginning of every turn, the user will be given several options.
-

- The options are (1) move, (2) read technical papers, (3) pickpocket, (4) view character, and (5) quit.
- If they choose (1), then their character will move one step closer to the goal while risking an encounter.
- If they choose (2), they will gain intelligence and lose time.
- If they choose (3), they will gain money and lose time.
- If they choose (4), they will view the attributes of their character and how many more steps they need to complete the game.
- If they choose (5), they will be given a "You Lose" message and the program will terminate.
- If they choose any other value, then they will be told that their input is invalid and will be told to choose a valid input.

Postcondition:

- If (1) was chosen, the user will have moved one space and possibly faced an encounter.
- If (2) or (3) were chosen, the user will have been brought back to the same menu if all attributes remain above zero, and will have been given a “You Lose” message otherwise, where the program will have been terminated.
- If (4) was chosen, the user will always have been returned to the same menu after being shown their character’s statistics.
- If (5) was chosen, the user will have been given a message and the program will have been terminated.
- If any other value was chosen, the user will always have been returned to the same menu.

Use Case 4: Puzzles

Precondition: The user chose (1) at the In-Game Options Menu and a puzzle was randomly chosen as an encounter.

Description:

- If the player chooses to move, there is a chance that they will encounter a puzzle.
- A question will be randomly selected and presented to the user, and they will freely type their response.
- If their response is correct, the user will be rewarded with a random attribute.
- If their response is incorrect, the user will lose a random attribute.
- In either case, after their response, the program will tell the user whether they were correct or incorrect, and it will also tell them the amount of the attribute that they gained or lost.

Postcondition:

- The user will have either gained or lost an attribute.
- The user will have either returned to the In-Game Options Menu if an attribute remained above zero, and will have been given a “You Lose” message otherwise, where the program will have been terminated.

Zachary Parker
3/19/17
COMP 2710

Part 2: Design

Class 1: Character

The Character class keeps track of all attributes related to the player's character. Any attributes or functions related to the character, such as their name, remaining steps, or point total, are stored here. The Character class does not use any other classes in its functions.

Class 2: Encounter

The Encounter class deals with encounters that occur during a game. It randomly decides which encounter occurs, chooses randomly from a list of puzzles if a puzzle is set to occur, and changes values accordingly if any other encounter is set to occur. The Encounter class uses elements of the Character class in some of its functions.

Class 3: Highscore

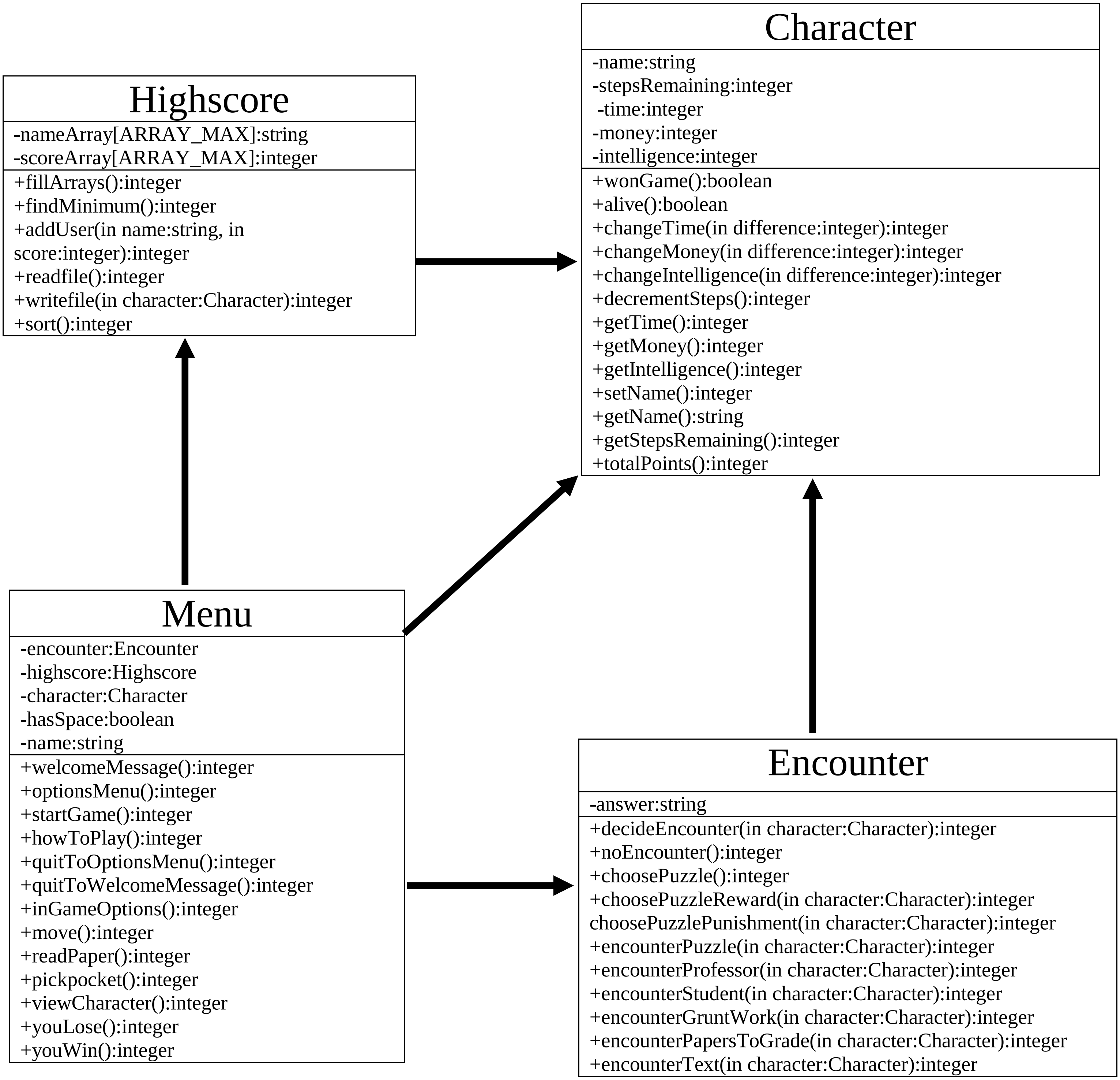
The Highscore class deals with everything related to high scores. It can change the list of high scores, print the list, and write the list to a file. The Highscore class uses elements of the Character class in some of its functions.

Class 4: Menu

The Menu class deals with all of the menus and other associated functions of the game itself. It contains functions for the welcome message, the options menu, the in-game options, and all of the associated choices the player can make from these menus. It also had functions that deal with the outcome of either winning or losing the game. Because so many other functions must be called from the Menu class, this class uses elements of the Encounter, Highscore, and Character classes in many of its functions.

Zachary Parker
3/19/17
COMP 2710

Part 2: UML Class Diagram



Highscore

-nameArray[ARRAY_MAX]:string
-scoreArray[ARRAY_MAX]:integer

+fillArrays():integer
+findMinimum():integer
+addUser(in name:string, in score:integer):integer
+readfile():integer
+writefile(in character:Character):integer
+sort():integer

Character

-name:string
-stepsRemaining:integer
-time:integer
-money:integer
-intelligence:integer

+wonGame():boolean
+alive():boolean
+changeTime(in difference:integer):integer
+changeMoney(in difference:integer):integer
+changeIntelligence(in difference:integer):integer
+decrementSteps():integer
+getTime():integer
+getMoney():integer
+getIntelligence():integer
+setName():integer
+getName():string
+getStepsRemaining():integer
+totalPoints():integer

Menu

-encounter:Encounter
-highscore:Highscore
-character:Character
-hasSpace:boolean
-name:string

+welcomeMessage():integer
+optionsMenu():integer
+startGame():integer
+howToPlay():integer
+quitToOptionsMenu():integer
+quitToWelcomeMessage():integer
+inGameOptions():integer
+move():integer
+readPaper():integer
+pickpocket():integer
+viewCharacter():integer
+youLose():integer
+youWin():integer

Encounter

-answer:string

+decideEncounter(in character:Character):integer
+noEncounter():integer
+choosePuzzle():integer
+choosePuzzleReward(in character:Character):integer
choosePuzzlePunishment(in character:Character):integer
+encounterPuzzle(in character:Character):integer
+encounterProfessor(in character:Character):integer
+encounterStudent(in character:Character):integer
+encounterGruntWork(in character:Character):integer
+encounterPapersToGrade(in character:Character):integer
+encounterText(in character:Character):integer

Zachary Parker
3/19/17
COMP 2710

Part 3: Testing

Test Case 1: Name Entry Test

This is a test for the user's name input when the first run the program. It has a test for a name with white space and a test for a name without white space.

Test Case 2: Main Menu Test

This is a test for the main menu. It has a test for starting a new game, a test for viewing the tutorial, a test for viewing scores, a test for quitting, and a test for an invalid input.

Test Case 3: In-Game Options Menu Test

This is a test for the in-game menu. It has a test for moving the character, a test for reading technical papers, a test for pickpocketing, a test for viewing the character, a test for quitting, and a test for an invalid input.

Test Case 4: In-Game Puzzle Test

This is a test for the puzzle encounters that sometimes occur after moving the character. There is a test for a correct answer and a test for an incorrect answer.

Test Case 5: End-of-game Test

This is a test for the scoreboard containing the high scores. There is a test for when the user fails the game, a test for when the user wins the game but fails to get a high score, and a test for when the user wins the game and the high score must be updated.