Justin Sawyer

Analysis:

This is a simple non-graphical game where the user tries to make it through Shelby Hall (make 20 steps) with the most intelligence, time, and money at the end. The uses would be the user starting the program, then typing in their name when prompted and the program will print a welcome message followed by a main menu, where the user will be given three option, start a new game, run high scores, or quit.

If the user starts a new game, it should generate the user an random intelligence, time, and money. From then on the user will be given 5 options (move forward, read technical papers, searc for loose change, view character, quit game). The user will keep getting these options until the user runs out of intelligence, time, or money in which case the user loses, or the user has taken 20 steps, in which case he wins the game. The game will then check to see if the user's score is eligible for the high scores and will record if it is.

Then the user is returned back to the main menu screen with the options to start a new game, view high scores, or quit. If the user chooses view high scores the top 5 high scores will be displayed to the user along the with the name of the players. If the user chooses to quit, then the program will terminate.

Design:

- Player
  - The Player class stores the name, intelligence, time, spaces, money, and score. The intelligence will This class depends on the File class for file file I/o functions and highScore class. Almost all the other classes depend on this one.
  - Variables
    - string userName
    - int intelligence
    - int time
    - int money
    - int score
    - int steps
  - Functions
    - setName
      - will change name to what is entered in parameter
    - calculateScore
      - Calculates score.
    - highScoreString
      - Prints the name folowed by a space then the current score
- Menu
  - The purpose of this class is to generate the main menu for the user to interact with, and handle those interactions properly. This class depends on the Player class and HighScores class.
  - Variables
    - Player currentUser
    - int userInput (input for selection on menu screen
  - Funtions
    - runMainMenu

        - Takes in paramteter for Player. will prompt user for name(sets name in Player object), then prints welcome message, then prints menu main menu options for user (1. To start new game, 2. to open high scores, 3. quit).
    - runMenu
        - Will give user option to 1. move forward, 2. read technical, 3. search for loose change, 4. view character, 5. quit game
- System
    - The purpose of this class is to store the main functinon and run all the other classes properly. This class depends on all other classes.
    - Functions
        - main
            - Instantiate a Player object and runMainMenu.
            - Set name of Player object
            - Use a while loop to keep everything running in the main menu
            - If user starts a new game, use a while loop to keep everything in game until game is finished, and runMenu
            - If user views high scores, then runHighScores
            - If user quits then terminate the program.
- Encounter
    - The purpose of this class is to manage the encounters the players occurs after taking every step. This class depends on the Player class, and Puzzle class.
    - Variables
        - int randEncounter
    - Functions
        - void runEncounter()
            - 25% nothing happens
            - 25% user encounter a puzzle
            - 10% encounter a professor. Loses random amount of time, but may slightly increase intelligence
            - 10% encounter another graduate student. Loses random amount of time
            - 15% attacked by grunt work, lose time and intelligence
            - 10% grade papers. Lose time, but gain money
- Puzzle
    - The purpose of this class is to instantiate and supply puzzles for the game. The Puzzle object consists of a question and 4 answers.
    - Variables
        - vecor<Puzzle> puzzles
    - Functions
        - createPuzzles
            - will instantiate pre-made puzzles and store them into the puzzles vector
        - getPuzzles
            - will pick and return a random Puzzle from puzzles
- HighScore
    - The purpose of this class is to display, record, and store the high scores.
    - Functions
        - runHighScore
            - Will display top 5 high scores.
        - checkIfHighScore

        - compares current user’s score to list of high scores and see if it’s eligible. Returns true if eligible and false otherwise.
    - recordScore
        - Will check to see if score is eligible in high scores and will record it.
- File
    - The purpose of this class is two provide the other classes with functions to read and write files. This class depends on the Player class.
    - Functions
        - readFile
            - Takes in the name of file and returns the numbers as an int vector. Also an overridden version to read Players
        - wrieFile
            - Takes in the name of file and will overwrite (write a new file if doesn’t exist) with the vector of ints. Also an overidden version to handle Players

Testing:

- Check if user gets same score as someone on the high scores and will add user to high score. If this situation occurs, the user’s new high score will take the place of the old one, with newer scores having higher priority in the high scores list.
- Check to see if the first, fifth, and sixth high scores will be recorded properly.
- Check to see if program will handle the user input properly, if their input is out of bounds from the options displayed to them in the menu, trying not to crash or quit the program, but instead asking the user to re-type their input correctly.
- Run test a few test cases to make sure readFile is working properly
- Run test a few test cases to make sure the HighScores functions are working properly
- Run test a few test cases to make sure the Puzzle functions are working properly
- Run test a few test cases to make sure the Encounters functions are working properly
- Run test a few test cases to make sure the Player functions are working properly