Amy Eddins – ale0010
Assignment 1A - "Dunstan and Dragons"
June 14, 2011

1. Program Analysis

When the user starts the program a main menu will appear on the screen asking the user if he or she would like to start a new game, view the top 5 high scores or quit the game. If the user chooses the first option, the program will ask for his or her name and a random value for intelligence, time and money will be given to the user. They can then choose to move, search for loose change, read technical papers, view character or quit the game. For each of the first three options the user can either gain or lose intelligence, time or money. If he or she chooses to move there is a 50/50 chance they will face an encounter. If they face an encounter they have a 10 percent chance they will encounter a professor or graduate student or have to grade papers and a 15 percent chance they will be attacked for grunt work. All these will also give or take away his or her attributes. If any of the player's attributes go to zero then he or she loses and the game is ended. A "you lose" message will appear. If the player reaches the end of the hall then his or her scores are multiplied, the score is saved, and a "you win" message will appear. If it is greater than the lowest high score it is saved as a new high score. The user is then informed that he or she got a new high score.

2. Program Design

A. Player class - This class saves the name, attributes, and score of each player.

1.Variables

a. name - string
b. intelligence - int
c. time - int
d. money - int
e. position - position object

2. Functions

a. chanceIntelligence(int) - change the intelligence amount
b. changeTime(int) - change the time amount
c. changeMoney(int) - change the money amount
d. changeScore(int) - change the score amount
e. getIntelligence() - returns intelligence amount
f. getTime() - returns the time amount
g. getMoney() - returns the money amount
h. calScore() - returns the score amount
i. changePlayerPosition(position object) - changes position
j. getPlayerPosition() - returns position object
k. toString() - displays the player's name, intelligence, time, money, score and position

B. Position class - This is a sub-class of the Hall class. This class saves the player's position in the hall and also contains the move methods and the random number generator to decide how much the player gains and loses when he or she reads technical papers or searches for loose change.

1.Variables

a. hall - double int array of values 0 or not 0 that tell whether the user can go there or not
b. col - int
c. row - int

2. Functions

a. getCol()
b. getRow()
c. moveLeft() - returns if they move left or not
d. moveRight() - returns if they move right or not
e. moveForward() - returns if they move forward or not

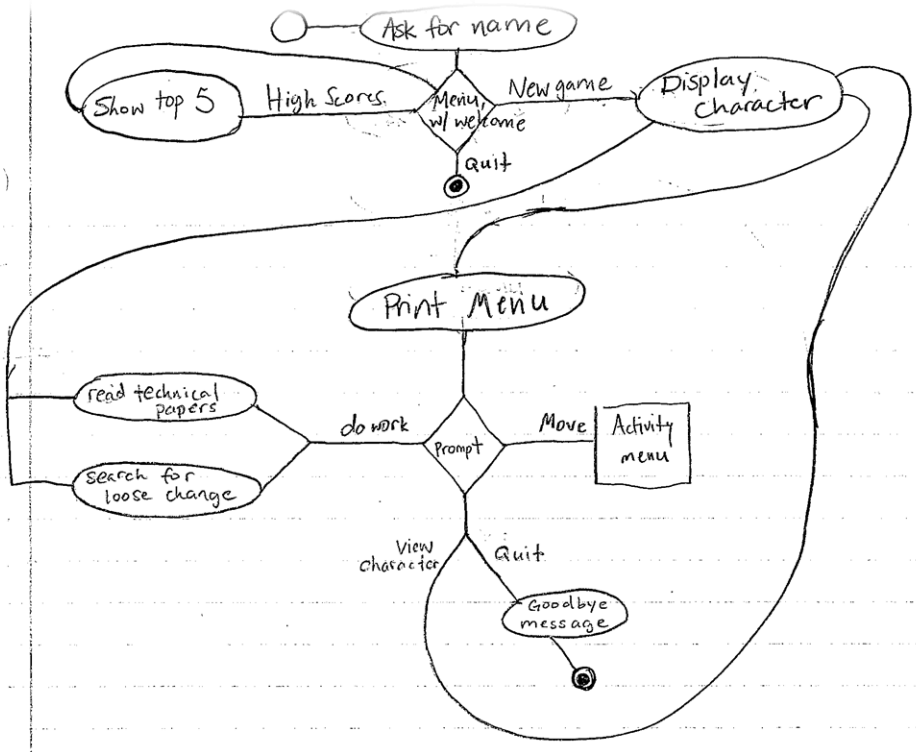
f. moveBackward() - returns if they move backward or not
g. setCol(int)
h. setRow(int)

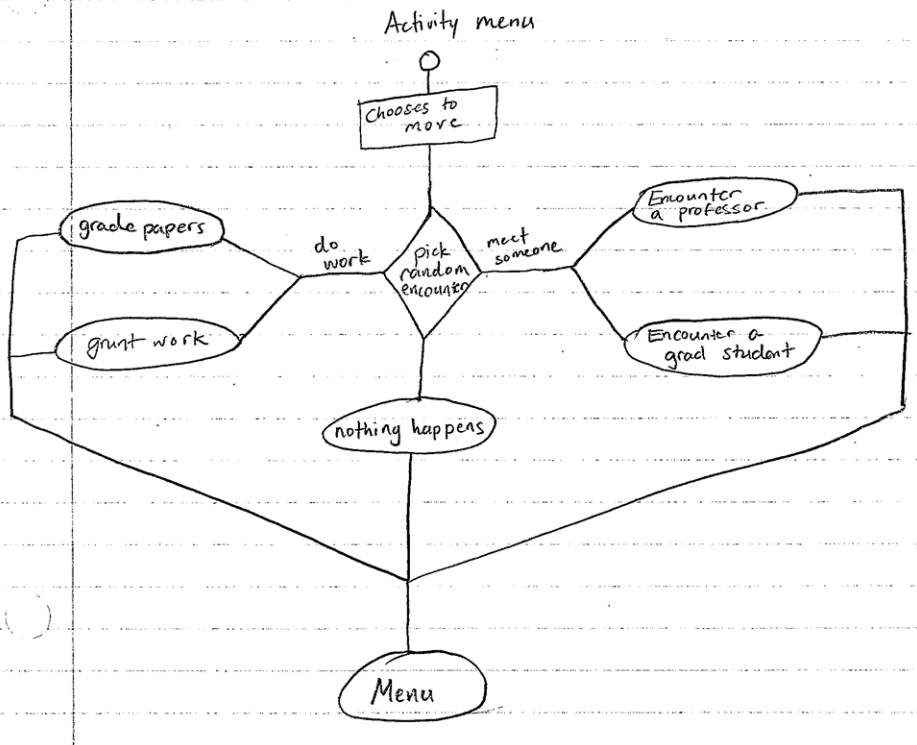
C. Menu class – This class has all the menu options.
1. Variables
a. highScores[5] – int array
2. Functions
a. move()
b. readTechnicalPapers()
c. searchLooseChange()
d. viewCharacter()
e. quit()
f. viewHighScores()
g. newGame()

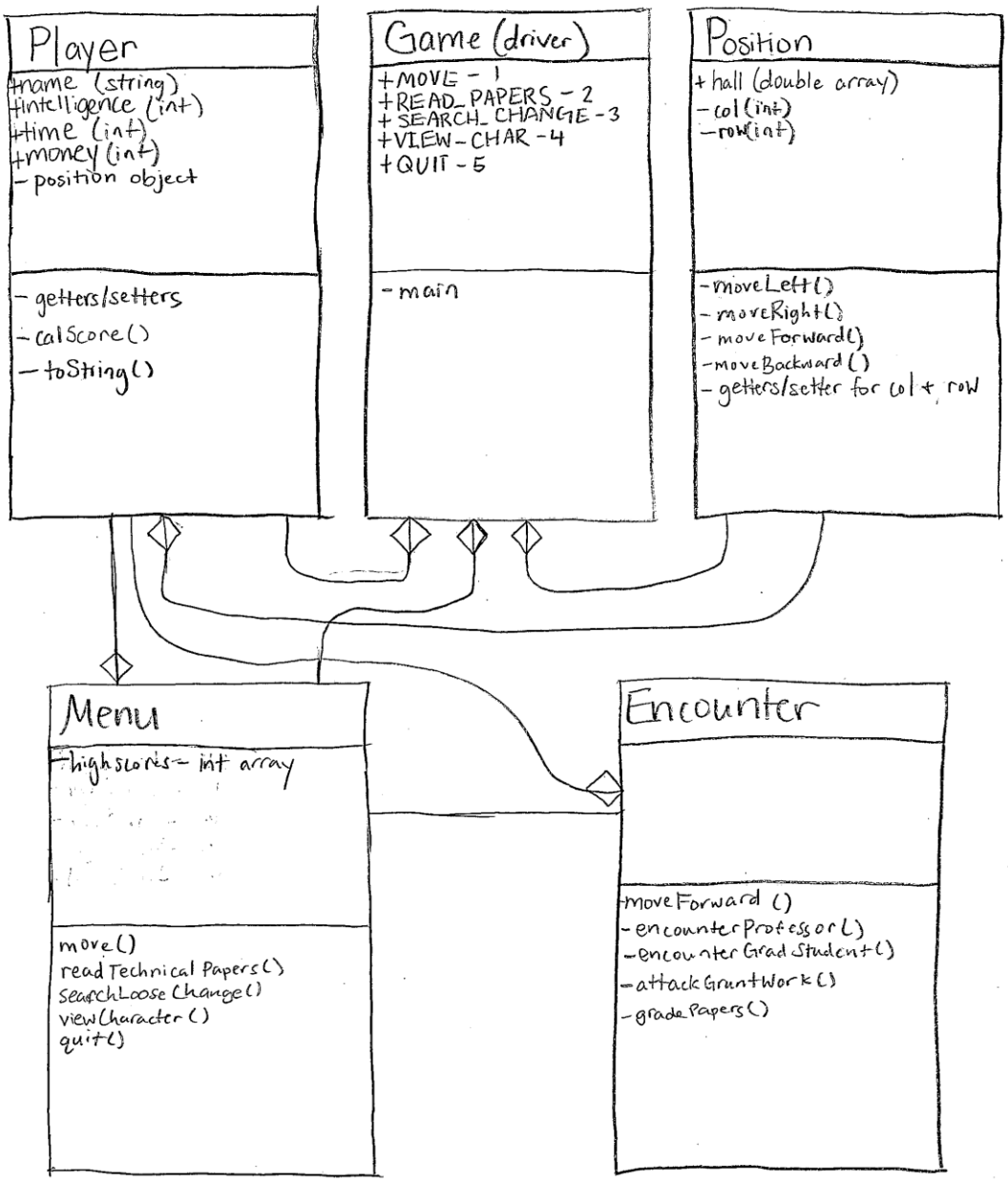
D. Encounter class - This class uses a random number generator to choose what encounter will happen.
1. Functions
a. moveForward() - randomly decides if the player should move forward
b. encounterProfessor() - randomly lose time but gain intelligence
c. encounterGradStudent() - randomly lose time
d. attackGruntWork() - randomly lose time and intelligence
e. gradePapers() - randomly lose time but gain money

E. Game(driver) – brings everything together to make the game.
1. Variables
a. MOVE - 1
b. READ_PAPERS - 2
c. SEARCH_CHANGE - 3
d. VIEW_CHAR - 4
e. QUIT - 5
f. HIGH_SCORES - 6
g. NEW_GAME - 7

3. Test Cases
A. Abnormal Usage
1. If any of the attributes reach zero the player loses. The program will not keep subtracting after one reaches zero. User then loses the game.
2. The program will ask the user to enter the menu choice again if he or she enters a negative number, a number that's not a menu option, a letter or any other character.
3. When choosing where to move they can enter a capital or lower case letter. Anything else the program asks the user to enter the option again.
4. A position cannot go past the bounds of the hall.
B. Normal Usage
1. If the user goes along a wrong path the only option will be to go backward until he or she gets back to the correct path.
2. When a new high score is made it replaces the lowest high score.
3. Always asks for name, displays character, and prints menu at beginning.
4. Does not come up with an incorrect random value


Ask for name
Show top 5
High Scores
Menu, w/ welcome
New game
Display character
Quit
Print Menu
read technical papers
search for loose change
do work
Prompt
Move
Activity menu
View character
Quit
Goodbye message


Activity menu
Chooses to move
grade papers
grunt work
do work
pick random encounter
meet someone
Encounter a professor
Encounter a grad student
nothing happens
Menu


Player
+name (string)
+intelligence (int)
+time (int)
+money (int)
- position object
- getters/setters
- calScore()
- toString()
Game (driver)
+MOVE - 1
+READ_PAPERS - 2
+SEARCH_CHANGE - 3
+VIEW-CHAR - 4
+QUIT - 5
- main
Position
+ hall (double array)
- col (int)
- row (int)
- moveLeft()
- moveRight()
- moveForward()
- moveBackward()
- getters/setter for col + row
Menu
-highscores - int array
move()
readTechnicalPapers()
searchLooseChange()
viewCharacter()
quit()
Encounter
+moveForward()
- encounterProfessor()
- encounterGradStudent()
- attackGruntWork()
- gradePapers()