Driver Class Algorithm

\*\*Imports java.util[Scanner]\*\*

Public static void main(String args[]){

Scanner scan = new Scanner();

String resp = “y”;

Boolean won = false;

**Do{**

Print(“Welcome to Adventure Time…!”);

Print(“Do you want to play? (Y/N)”);

|Try/catch for user response|

resp = scan.next().toLowerCase();

if(resp.equals(“n”))

break;

GameMap game = new GameMap();

**While**(!getCurrrentRoom.equals(getFinalRoom()) && chancesLeft())//loop doors

Room[] rooms = game.getConnectingRooms();

Print(game.printRooms()));

How door will appear to user when printed. –locked/+unlocked🡪 instructions

[1-][2+][3+][4-]

Print(“Select a room number. Enter 0 to quit”);

~~|Try/catch for user response|~~

Int choosenRoom = scan.nextInt();

If(choosenRoom ==0)

Resp = “n”;

Break;

If(getCurrentRoom().getDoor(choosenRoom).isLocked())

Print(getLockedMsg());

Key[] keys = getCurrentRoom().getRoomKeys();

String choice = “n”;//selected key

**While**(choice.equals(“n”) && chancesLeft()) //choosing keys

Print(“You have ”+ getChances()+” to guess or you lose!”);

Print(game.printKeys());

\*\*\*\*User must spell word correctly\*\*\*\*

[apple][dog][cat][elephant][pen][notebook][cabbage][snowman][mat][pin]

Print(“Type the name of the key you want to view. Type Q to quit”);

~~|Try/catch for user response|~~

String choosenKeyStr = scan.next().toLowerCase();

If(choosenKeyStr.equals(“q”))

Resp = “n”;

Break;

Key choosenKey = getKey(choosenKeyStr);

Print(“The key tells you “ +choosenKey.getActiveMsg());

Print(“Is this the key you choose to unlock the door?(Y/N). Enter Q to Quit);

~~|Try/catch for user response|~~

choice = scan.next().toLowerCase();

If(choice.equals(“q”))

Resp = “n”;

Break;

If(choice.equals(“y”))

if(rightKey(choosenKey))

Print(getUnlockedMsg()); //DOOR

unlockDoor(Door lockedDoor); //Ref. a 0 door instead

else //Key was incorrect

Print(“The key tells you “+ choosenKey.getInactiveMsg());

Print(“Oh no! That’s not the key!”);

wrongAnswer(); //max\_chance

choice = “n”;

Else//Not a room that is locked

Print(unlockedDoorMsg());//for door zero. Even if you unlock door, you still have a choice not to enter it.

Print(“Do you wish to enter room “+ rooms[choosenRoom]? (Y/N). Enter Q to quit.)

~~|Try/catch for user response|~~

String enteringDoorResp = scan.next().toLowerCase();

If(enteringDoorResp.equals(“q”))

Resp = “n”;

Break;

If(“y”)

setCurrentRoom(rooms[choosenRoom]);

//Outside of door loop

If(getCurrrentRoom.equals(getFinalRoom())

Hangman minigame = new Hangman();

..omitted must be fixed🡪 Jaguilar

If(minigame.status())

Won = true;

If(won)

Print(“Yay you won! You saved the Don in distress!!” );

Print(“Would you like to play again?(y/n)”);

~~|Try/catch for user response|~~

String newGame = scan.next().toLowerCase();

If(newGame.equals(“n”));

Resp = newGame;

Else{//game was lost

Print(“Sorry you lost the Don! He got taken to a new location!”);

Print(“Do you want to try to find him again?(y/n)”);

~~|Try/catch for user response|~~

String newGame = scan.next().toLowerCase();

If(newGame.equals(“n”));

Resp = newGame;

}

**While**( resp.equals(“y”) *//The User still wants to play*);

Print(“Oh no, sorry to see you go! Good Bye!”);

}

List of needed Methods:

1. getConnectingRooms(); 🡪 GameMap [done]

/\*\* It returns the rooms that the current room connects to[the successors]. Could be array||linkedList, whatever works. If linkedList, must alter the calling methods of the array to be that of a linkedList .

@param:-- @return: LinkedList/Array of the connecting rooms \*/

1. getDoor(Room selectedRoom); 🡪 GameMap [done]

/\*\* We want it to return the door object that connects to the current door to said room.

@param: Room selectedRoom @return: Door connectingDoor \*/

1. getRoomKeys(); 🡪 Room Class [done]

/\*\* Returns an array of the keys being held in the current room.

@param:-- @return: Key[] roomKeys \*/

1. validKey(String choosenKeyString): Room Class [done]

/\*\*NOT IN CODE BUT WE NEED A METHOD THAT CHECKS IF THE TEXT THE USER ENTERED IS A VALID KEY.WE WON’T USE THIS BESIDES IN THE TRY/CATCH BLOCK FOR THE RESPONSE WHICH WAS OMITTED BUT WE STILL NEED IT

@param: String userResponse @return: Boolean isValid \*/

1. getKey(String choosenKeyString); 🡪Room Class [done]

/\*\* Once the key has been checked to be valid[validKey(-)], this method returns the key object for the corresponding Key name[String] by looping through and comparing the keys.

@param: String nameOfSelectedKey @return: Key key \*/

1. chancesLeft(); 🡪 GameMap [done]

/\*\* Returns whether there are chances left for the user to guess.

@param: -- @return: Boolean chancesLeft\*/

1. getChances(); 🡪 GameMap [done]

/\*\* Returns number of chances left for the user to select wrong keys.

@param: -- @return: int chancesLeft \*/

1. unlockDoor(Door lockedDoor) 🡪 Room Class [done – I had to put this method in GameMap]

/\*\* Changes, in the adjMatGraph, the weight of the current room and the

connecting room’s edges to zero

NOTE: May need to take in room, IDK

@param: Door doorToUnlock @return: void \*/

1. wrongAnswer() 🡪 GameMap [done – chancesLeft counts down from MAX\_CHANCES]

/\*\* Increments the count when the user enters a wrong key.

@param:-- @return: void \*/

1. setCurrentRoom(Room selectedRoom) 🡪 GameMap [done]

/\*\* Replaces the currentRoom instance variable to point at the entered room.

@param: Room roomToReplaceCurrrent @return: void \*/

1. printRooms() 🡪 GameMap //Does it from the current Room

/\*\* Returns a string of the rooms with the room number as well as a +||- to indicate whether the doors between the current room and connecting room is locked or unlocked[weight 0 or not].

@param: -- @return: String stringRepresentationOfTheRooms\*/

1. printKeys() 🡪 GameMap //Does it from the current room

/\*\* Returns a string of the all the keys in the room where the key names are represented.

@param: -- @return: String stringRepresentationOfCurrentRoomKeys\*/

//\*\*\*\*\*\*\*\*\*\*\*Fix the chances MAX\_CHANCES to represent correctly in algorithm

Already fixed the algorithm and included the necessary methods needed\*\*\*\*\*\*\*\*\*\*\*\*\\