**TITLE PAGE**

**Course:** CS1073

**Section:** FR03B

**Assignment number:** 7

**Name:** Zohaib Hassan Khan

**UNB student number:** 3740572

MoveAnaylzer.java:

/\*\*

This is a move analyzer program.

@author Zohaib Khan - 3740572

\*/

public class MoveAnalyzer {

/\*\*

This method checks if the hero's attempted move can be completed

or not.

@param maze the 2d array on which the maze is based on.

@param direction the direction in which we want the hero to move.

@return whether the hero’s attempted move can be completed

\*/

public static boolean checkMove (char[][] maze, char direction) {

int row = -1;

int col = -1;

boolean flag = false;

for (int i = 0; i < maze.length && !flag; i++) {

for (int j = 0; j < maze[0].length && !flag; j++) {

if (maze[i][j] == 'H') {

row = i;

col = j;

flag = true;

}

}

}

if (row == -1 || col == -1) {

return false;

}

int newRow = row;

int newCol = col;

if (direction == 'D') {

newRow++;

if (newRow >= maze.length) {

return false;

}

}

else if (direction == 'L') {

newCol--;

if (newCol < 0) {

return false;

}

}

else {

return false;

}

if (maze[newRow][newCol] == 'U') {

return true;

}

else if (direction == 'D' && newRow + 1 < maze.length &&

maze[newRow + 1][newCol] == 'U') {

return true;

}

else if (direction == 'L' && newCol - 1 >= 0 &&

maze[newRow][newCol - 1] == 'U') {

return true;

}

else {

return false;

}

}

}

As7Q2Output.txt:

arr1, Down direction (should be true): true

arr1, Left direction (should be true): true

arr2, Down direction (should be true): true

arr2, Left direction (should be true): true

arr3, Down direction (should be true): true

arr3, Left direction (should be true): true

arr4, Down direction (should be false): false

arr4, Left direction (should be false): false

arr5, Down direction (should be false): false

arr5, Left direction (should be false): false

EntertainmentItem.java:

/\*\*

This class represents an entertainment item.

@author Zohaib Khan - 3740572.

\*/

public class EntertainmentItem {

/\*\*

This is the description of the item.

\*/

private final String description;

/\*\*

This is the price of the item.

\*/

private final double price;

/\*\*

This lets us know if the item was donated or not.

\*/

private final boolean isDonated;

/\*\*

This is the constructor method to initialize instance variables.

@param description the description of the item.

@param price the price of the item.

@param isDonated whether the item is donated or not.

\*/

public EntertainmentItem(String description, double price, boolean

isDonated) {

this.description = description;

this.price = price;

this.isDonated = isDonated;

}

/\*\*

This method gets the description of the item.

@return the description of the item.

\*/

public String getDescription() {

return description;

}

/\*\*

This method gets the price of the item.

@return the price of the item.

\*/

public double getPrice() {

return price;

}

/\*\*

This method checks if the item is donated or not.

@returns whether the item is donated or not.

\*/

public boolean getBenefactorDonated() {

return isDonated;

}

}

ResidentMember.java:

/\*\*

This class represents a resident member.

@author Zohaib Khan - 3740572

\*/

public class ResidentMember {

/\*\*

This is the full name of the member.

\*/

private String fullName;

/\*\*

This is the member's congo unit number.

\*/

private int unitNumber;

/\*\*

This is the member's phone number.

\*/

private String phoneNumber;

/\*\*

This is the membership number assigned to the member.

\*/

private final int membershipNumber;

/\*\*

This is a counter to increment the membership number for each

member.

\*/

private static int nextNumber = 500000;

/\*\*

This is an object of EntertainmentItem class.

\*/

private EntertainmentItem[] items;

/\*\*

This is a counter variable of the items array.

\*/

private int itemsCounter;

/\*\*

This is the constructor method to initialize the instance

variables.

@param fullName the full name of the member.

@param unitNumber the member's congo unit number.

@param phoneNumber the member's phone number.

\*/

public ResidentMember(String fullName, int unitNumber, String

phoneNumber) {

this.fullName = fullName;

this.unitNumber = unitNumber;

this.phoneNumber = phoneNumber;

membershipNumber = nextNumber;

nextNumber++;

items = new EntertainmentItem[7];

itemsCounter = 0;

}

/\*\*

This method gets the member's name.

@return the full name of the member.

\*/

public String getName() {

return fullName;

}

/\*\*

This method returns the member's congo unit number.

@return the member's congo unit number.

\*/

public int getUnitNumber() {

return unitNumber;

}

/\*\*

This method returns the member's phone number.

@return the member's phone number.

\*/

public String getPhoneNumber() {

return phoneNumber;

}

/\*\*

This method returns the member's membership number.

@return the member's membership number.

\*/

public int getMembershipNumber() {

return membershipNumber;

}

/\*\*

This method sets the member's phone number.

@param phoneNumber the member's phone number.

\*/

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

/\*\*

This method returns the list of items signed out by the member.

@return itemList the list of items signed out by the member.

\*/

public EntertainmentItem[] getSignedOutItems() {

EntertainmentItem[] itemList =

new EntertainmentItem[itemsCounter];

for (int i = 0; i < itemList.length; i++) {

itemList[i] = items[i];

}

return itemList;

}

/\*\*

This method checks if signing out an entertainment item is

possible or not.

@param o the entertainment item the member is attempting to sign

out.

@return if signing out the item was successful or not

\*/

public boolean signOut(EntertainmentItem o) {

boolean flag = false;

if (itemsCounter < items.length) {

items[itemsCounter] = o;

itemsCounter++;

flag = true;

}

else {

flag = false;

}

return flag;

}

/\*\*

This method checks if signing out an entertainment item is

possible or not.

@param o the entertainment item the member is attempting to sign

out.

@return if signing out the item was successful or not

\*/

public boolean returnItem(EntertainmentItem o) {

boolean flag = false;

for (int i = 0; i < itemsCounter && !flag; i++) {

if (items[i] == o) {

for (int j = i; j < itemsCounter - 1; j++) {

items[j] = items[j + 1];

}

itemsCounter--;

items[itemsCounter] = null;

flag = true;

return flag;

}

}

return flag;

}

}

ShortTermResidentMember:

\*\*

This is a class for short term resident members.

@author Zohaib Khan - 3740572

\*/

public class ShortTermResidentMember extends ResidentMember {

/\*\*

The departure date of the member.

\*/

private String departureDate;

/\*\*

The constructor method to initialize instance variables.

@param fullName the full name of the member.

@param phoneNumber the phone number of the member.

@param departureDate the departure date of the member.

\*/

public ShortTermResidentMember (String fullName, int unitNumber,

String phoneNumber,

String departureDate) {

super(fullName, unitNumber, phoneNumber);

this.departureDate = departureDate;

}

/\*\*

This method gets the departure date of the member.

@return the departure date of the member.

\*/

public String getDepartureDate() {

return departureDate;

}

/\*\*

This method checks if signing out an entertainment item is

successful or not.

@param o the entertainment item that the member wants to sign

out.

@return if signing out the item was successful or not.

\*/

public boolean signOut(EntertainmentItem o) {

if(!o.getBenefactorDonated()) {

return super.signOut(o);

}

else{

return false;

}

}

/\*\*

This method checks if returning an entertainment item is

successful or not.

@param o the entertainment item that the member wants to return.

@return if returning the item was successful or not.

\*/

public boolean returnItem(EntertainmentItem o) {

return super.returnItem(o);

}

}

As7Q2Output.txt:

\*\*\* Test case #1: Create a ResidentMember object & test accessors

Name: Maria Lopez

Unit #: 163

Phone: 555-1234

Member #: 500000

Correct result: Maria has zero entertainment items.

\*\*\* Test case #2: Create a ShortTermResidentMember object & test accessors

Name: Tommy MacDonald

Unit #: 306

Phone: 555-8642

Member #: 500001

Departs: Apr. 26, 2023

Correct result: Tommy has zero entertainment items.

\*\*\* Test case #3: Automatically generate a member number

Correct result: 500002 is the correct member number.

\*\*\* Test case #4: Create an EntertainmentItem object & test accessors

Description: Uno - Card Game

Original Price: $12.00

Benefactor Donated: true

\*\*\* Test case #5: Change phone number for both resident types

Correct result: Maria's phone number successfully changed.

Correct result: Tommy's phone number successfully changed.

\*\*\* Test case #6: Sign out one EntertainmentItem

Correct result: Maria signed out an item successfully.

Correct result: Maria has one entertainment item.

\*\*\* Test case #7: Sign out multiple EntertainmentItems

Correct result: Maria signed out two more items successfully.

Correct result: Maria has three entertainment items.

\*\*\* Test case #8: Intentionally exceed the sign out limit

Correct result: Maria was prevented from signing out more than 7 entertainment items.

\*\*\* Test case #9: A short-term resident tries to sign out items

Correct result: Tommy was prevented from signing out a benefactor-donated item.

Correct result: Tommy was able to sign out a non-benefactor-donated item.

\*\*\* Test case #10: Returning the only item that was signed out

Correct result: Tommy's item was successfully returned.

Correct result: Tommy's list length changed appropriately.

\*\*\* Test case #11: Returning an item that was not signed out

Correct result: Unsuccessful attempt to return an item that was not signed out.

\*\*\* Test case #12: Returning the first item that was signed out

Correct result: Maria's first item was successfully returned.

Correct result: Maria's list length changed appropriately.

Confirm return: Uno should be absent from the following list:

Connect 4 - Board Game

Skip-Bo - Card Game

Harmonica - Musical Instrument

Scrabble - Board Game

Codenames - Card Game

Ukulele - Musical Instrument

\*\*\* Test case #13: Returning a mid-list item

Correct result: Skip-Bo was successfully returned.

Correct result: Maria's list length changed appropriately.

Confirm return: Skip-Bo should be absent from the following list:

Connect 4 - Board Game

Harmonica - Musical Instrument

Scrabble - Board Game

Codenames - Card Game

Ukulele - Musical Instrument

\*\*\*\*\*\*\*\*\*\*\*\*\* End of Test Cases \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*