Isaac Ray Shoebottom

CS 1073 (FR02A)

Assignment 12

3429069

# Section A

## Source Code:

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

/\*\*

\* Decodes encrypted text in a specific format

\* @author Isaac Shoebottom (3429069)

\*/

public class Decoder {

public static void main(String[] args) throws FileNotFoundException {

Scanner scanFile = new Scanner(new File(args[0]));

int cycleCount = 0;

scanFile.useDelimiter("\\A");

String in = scanFile.next();

String[] codes = in.split("\\r?\\n");

for (String i: codes) {

if (i.length() > 1) {

int columns = Integer.parseInt(codes[cycleCount \* 2]);

int rows = codes[cycleCount \* 2 + 1].length() / columns;

char[][] decode = new char[rows][columns];

char[] chars = i.toCharArray();

int charCounter = 0;

for (int k = 0; k < columns; k++) {

if (k % 2 != 0) {

for (int j = 0; j < rows; j++) {

decode[j][k] = chars[charCounter];

charCounter++;

}

}

else {

for (int j = rows - 1; j > -1; j--) {

decode[j][k] = chars[charCounter];

charCounter++;

}

}

}

charCounter = 0;

char[] decodedChar = new char[i.length()];

for (int j = 0; j < rows; j++) {

if (j % 2 == 0) {

for (int k = 0; k < columns; k++) {

decodedChar[charCounter] = decode[j][k];

charCounter++;

}

}

else {

for (int k = columns - 1; k > -1; k--) {

decodedChar[charCounter] = decode[j][k];

charCounter++;

}

}

}

cycleCount++;

String output = String.valueOf(decodedChar);

System.out.println(output);

}

else if (i.equals("0")) {

break;

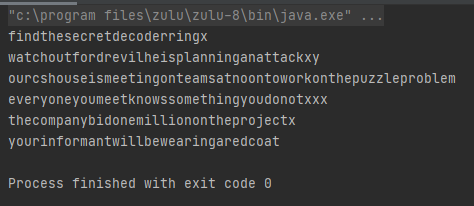
}

}

}

}

## Output:



findthesecretdecoderringx

watchoutfordrevilheisplanninganattackxy

ourcshouseismeetingonteamsatnoontoworkonthepuzzleproblem

everyoneyoumeetknowssomethingyoudonotxxx

thecompanybidonemillionontheprojectx

yourinformantwillbewearingaredcoat

# Section B

## Source Code (ResidentMember):

/\*\*

\* This class holds info for members

\* @author Isaac Shoebottom (3429069)

\*/

public class ResidentMember {

private static int membership = 9999;

private final String name;

private final int room;

private final LendingItem[] list;

private String phone;

private int bookCounter;

/\*\*

\* Constructor for members

\* @param name Name of member

\* @param room Room of member

\* @param phone Phone of member

\*/

public ResidentMember(String name, int room, String phone) {

this.name = name;

this.room = room;

this.phone = phone;

list = new LendingItem[8];

bookCounter = 0;

membership++;

}

/\*\*

\* Get the signed out items of a member

\* @return The updated list item list

\*/

public LendingItem[] getSignedOutItems() {

LendingItem[] updatedItemList = new LendingItem[bookCounter];

for (int i = 0; i < bookCounter; i++) {

updatedItemList[i] = list[i];

}

return updatedItemList;

}

/\*\*

\* Method to sign out items for members

\* @param input The item to be lent

\* @return Boolean for if the item was signed out or not

\*/

public boolean signOut(LendingItem input) {

if (bookCounter < 8) {

list[bookCounter] = input;

bookCounter++;

return true;

}

return false;

}

/\*\*

\* Returns and item for members

\* @param input The item to be returned

\* @return Boolean if the item was returned

\*/

public boolean returnItem(LendingItem input) {

boolean success = false;

for (int i = 0; i < bookCounter; i++) {

if (list[i] == input) {

list[i] = list[bookCounter - 1];

bookCounter--;

success = true;

}

}

return success;

}

/\*\*

\* Gets the name of member

\* @return The name of member

\*/

public String getName() {

return name;

}

/\*\*

\* Gets the room of member

\* @return The room of member

\*/

public int getRoomNumber() {

return room;

}

/\*\*

\* Gets the phone number of member

\* @return The phone number of member

\*/

public String getPhoneNumber() {

return phone;

}

/\*\*

\* Sets the phone of member

\* @param phone

\*/

public void setPhoneNumber(String phone) {

this.phone = phone;

}

/\*\*

\* Gets the membership number of member

\* @return The membership number of member

\*/

public int getMembershipNumber() {

return membership;

}

}

## Source Code (ShortTermResidentMember):

/\*\*

\* This class holds info for short term members

\* @author Isaac Shoebottom (3429069)

\*/

public class ShortTermResidentMember extends ResidentMember {

private final String departureDate;

/\*\*

\* Constructor for short term members

\* @param name Name of short term member

\* @param room Room of short term member

\* @param phone Phone of short term member

\* @param departureDate The departure date of short term members

\*/

public ShortTermResidentMember(String name, int room, String phone, String departureDate) {

super(name, room, phone);

this.departureDate = departureDate;

}

/\*\*

\* Method to sign out items for short term members

\* @param input The item to be signed out

\* @return Boolean for if the item was signed out or not

\*/

public boolean signOut(LendingItem input) {

if (input.isBookClubRecommended()) {

super.signOut(input);

return true;

}

return false;

}

/\*\*

\* Gets the departure date for short term members

\* @return The departure date

\*/

public String getDepartureDate() {

return departureDate;

}

}

## Source Code (LendingItem):

/\*\*

\* This class holds info for the lent item

\* @author Isaac Shoebottom (3429069)

\*/

public class LendingItem {

private final String description;

private final double price;

private final boolean recommended;

/\*\*

\* Constructor for items

\* @param description Description of item

\* @param price The price of the item

\* @param recommended If the item is recommended or not

\*/

public LendingItem(String description, double price, boolean recommended) {

this.description = description;

this.price = price;

this.recommended = recommended;

}

/\*\*

\* Gets the description of item

\* @return The item description

\*/

public String getDescription() {

return description;

}

/\*\*

\* Gets the price of the item

\* @return The price of the item

\*/

public double getPrice() {

return price;

}

/\*\*

\* Gets if the item is recommended by the book club

\* @return Boolean of if the item is recommended by the book club

\*/

public boolean isBookClubRecommended() {

return recommended;

}

}

## Output:

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

Name: Maria Melani

Appt #: 152

Phone: 555-1234

Member #: 10000

Correct result: Maria has zero lending items.

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

Name: Tommy Black

Appt #: 302

Phone: 555-4321

Member #: 10001

Departs: Dec. 15, 2020

Correct result: Tommy has zero lending items.

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

Correct result: 10002 is the correct member number.

\*\*\* Test case #4: Create a LendingItem object & test accessors

Description: Lean In - Sheryl Sandberg - BOOK

Orig. Price: $10.00

Book Club Recommended: 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 LendingItem

Correct result: Maria signed out an item successfully.

Correct result: Maria has one lending item.

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

Correct result: Maria signed out two more items successfully.

Correct result: Maria has three lending items.

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

Correct result: Maria was prevented from signing out more than 8 lending items.

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

>> ERROR: Tommy was able to sign out a book club recommended item.

>> ERROR: Tommy was prevented from signing out a non recommended 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: Lean In should be absent from the following list:

Yoga Journal - October 2020 - MAGAZINE

Maclean's - 23/11/2020 - MAGAZINE

Headstrong: 52 Women Who Changed Science and the World - Rachel Swaby - BOOK

The Time Machine - H.G. Wells - BOOK

The Confidence Code - Katty Kay & Claire Shipman - BOOK

The Immortal Life of Henrietta Lacks - Rebecca Skloot - BOOK

Grit - Angela Duckworth - BOOK

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

Correct result: The Time Machine was successfully returned.

Correct result: Maria's list length changed appropriately.

Confirm return: The Time Machine should be absent from the following list:

Yoga Journal - October 2020 - MAGAZINE

Maclean's - 23/11/2020 - MAGAZINE

Headstrong: 52 Women Who Changed Science and the World - Rachel Swaby - BOOK

Grit - Angela Duckworth - BOOK

The Confidence Code - Katty Kay & Claire Shipman - BOOK

The Immortal Life of Henrietta Lacks - Rebecca Skloot - BOOK

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