CS1073 - Assignment #12 – Fall 2020

Submission Deadline: THURSDAY, December 10th before 12:00 MIDNIGHT (by 11:59 pm), Atlantic Daylight Time Zone, in the Assignment 12 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is to:

- Give you some practice with two-dimensional arrays.
- Give you some practice with partially filled arrays of objects.
- Introduce class (static) variables.
- Review some of the material from earlier in the course.

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.

1. Code Decryption[1]

Note the following requirement: You must use a two-dimensional array in your solution.

You are working as a codebreaker for an intelligence agency. Your informers have uncovered a new method used by some adversaries (and allies) to transmit encrypted messages. You've been tasked with writing an application that performs the *decryption* and outputs the unscrambled messages. The *encryption* is performed as follows:

The number of columns is selected first. Afterwards, the message (letters only) is written across the rows alternating left-to-right and right-to-left, and padding with extra random letters so as to make a rectangular array of letters. For example, if the message is "Find the secret de-coder ring!" and there are five columns, the following would be written down:

```
f i n d t
c e s e h
r e t d e
r e d o c
r i n g x
```

[1] Based on the 2004/2005 ACM Regional Programming Competition in the "North America – East Central" region

Note that only letters are included and written all in lower case. In this example, the character `'x'` was used to pad the message to make a rectangle; however, any letters could be used for padding. Afterwards, the message is sent by writing the letters in each column (left to right), alternating bottom-to-top and top-to-bottom. So the above would be encrypted as:

```
rrrcfieeeindtsndedogxceht
```

Your task is to recover the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain a non-negative integer indicating the number of columns to use when creating the array. The next line is a string of lower-case letters; this is the encrypted message. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Sample Input

```
5
rrrcfieeeindtsndedogxceht
3
kcnanaehrduowahtrelilngaaxyttnipsivofct
0
```

Sample Output

```
findthesecretdecoderringx
watchoutfordrevilheisplanninganattackxy
```

Testing

Once you have tested your solution with input that you type in by hand, please use the input file that we have prepared, named **`cypherText.in`** (available in Desire2Learn).

Recall: To feed in the input data from a file, simply use the **<** symbol followed by the name of the file that contains the input, e.g.:

```
java Decryptor < cypherText.in
```

Save the output that is produced when you run your program with **`cypherText.in`** as the input.

2. Lending Library Classes

A friend of yours is part of the management team of a university residence building. To encourage students to read more in their leisure time, a new lending library has recently been added to the building; they plan to soon begin lending out books, magazines, etc. to residents. On your friend's recommendation, the university hired you to develop software to support this new lending library. You have decided to begin by writing three classes, named **LendingItem**, **ResidentMember**, and **ShortTermResidentMember**.

The **ResidentMember** class represents a resident of the building who has signed up for the building's lending library. You must record the full name of each resident member (e.g. "Maria Melani"), along with their room number (e.g. 152), and their phone number (e.g. "555-1234"). Each person is also automatically assigned a membership number when they are entered into the system. These membership numbers are assigned in ascending (increasing) order, starting at the number 10000. Once assigned, each resident will retain the same membership number (it will not change from year to year).

For each resident, we need to keep track of what items they currently have signed out. A **LendingItem** is any item that is available at this facility. We keep a brief description of each item (e.g. "The Fellowship of the Ring – J.R.R Tolkien – BOOK"). We also record the original price of the item (e.g. 24.95), and whether or not it is a book that has been recommended by one of the book clubs on campus. Once these values are initialized, they cannot be changed. Three accessor methods must be included in the **LendingItem** class to retrieve the information when needed.

Mutator methods must be provided in the **ResidentMember** class to allow the resident to sign out items that they wish to read and to return items that they have finished reading. The managers of the residence have decided that return dates will not be assigned to items when they are signed out; a resident may take as long as they need to read the book or magazine. However, each resident can have at most 8 items signed out at any given time. The methods for signing out and returning items both return a boolean value to indicate whether or not the transaction was successful. When an item is successfully returned, the array of currently signed out items must be adjusted so the signed out items are always stored in contiguous elements at the beginning of the array. It is not necessary to maintain the signed out items in any particular order.

The methods for signing out and returning an item both receive a **LendingItem** object as their only parameter. When returning an item, search through the array to find the item provided, and if found, remove it from the array. Two **LendingItem** objects are considered to be equal if their description, price, and book club recommendation status are all the same.

Four accessor methods should be provided in the **ResidentMember** class to retrieve the person's name, room number, phone number, and membership number. An accessor method should also be provided to retrieve a ***copy*** of the list of items that a resident currently has in their possession. This list is to be returned as an array, and this array will always be full. Since the list of items for a given resident can vary between 0 and 8 at any point in time, the length of the array returned by this accessor method will vary between 0 and 8.

The residence managers have noted that they wish to handle short-term residents (i.e. people who stay in the residence for less than a term) a bit differently than other residents. They note that the general information (name, room number, and phone number) is still important, and each **ShortTermResidentMember** still receives a membership number. However, for short-term residents, we must also record their departure date (e.g. "December 15, 2018"), and be able to access this information when required. Furthermore, short-term residents are not allowed to sign out any items that have been recommended by a university book club. (Items that have been recommended by a book club tend to be the most popular, and are therefore reserved for "regular" (longer term) residents.)

The items signed out by a resident (including short-term ones) may change over time. A person's phone number may also change. However, all other resident information, once set, should remain fixed.

Write the complete **LendingItem**, **ResidentMember**, and **ShortTermResidentMember** classes. For full marks, you must use inheritance. Also note: Data structures other than arrays (e.g. Vectors, ArrayLists, Linked Lists, etc.) are **not** permitted in this assignment (as they are not covered until CS1083). Also, for full marks you must use the techniques that were discussed in our CS 1073 lectures whenever you work with a partially filled array.

Test your classes by using the **LendingLibraryTestDriver** class that is provided. ***Do not change anything in this class***. For full marks you must make your classes compatible with this class, and all test cases must complete successfully.

Submission instructions are on the next 2 pages…

Your electronic assignment submission (submitted via Desire2Learn) will consist of two files:

i. a written report. This should begin with a title page; just as we described in Assignment #1, your title page should include: the course (CS 1073), your section (FR01A, FR02A, FR03A or FR04A), the assignment number, your full name, and your UNB student number. That should be followed by five sections, with each part clearly identified with a section heading. Include:

   a. the source code for the decryption problem (question 1)

   b. the output produced when testing your decryption solution with `cypherText.in`

   c. the source code for the three requested classes for question 2

   d. the output produced when testing your question 2 solution with the `LendingLibraryTestDriver` class that was provided.

   Note: As always, your source code should contain Javadoc comments, but you do not need to include the .html files.

   This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) The **SINGLE pdf file** containing your report will be submitted to the appropriate assignment drop box on Desire2Learn. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

   Note: Please name this report as follows: **YourName_As12_Report.pdf**

Continued on the next page…

ii. an archive file **(.zip)** that contains your Java source code and output for this assignment. Make sure that your archive includes all .java files (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

    Note: Please name this archive file as follows:
    YourName_As12_Archive.zip

Please note: As indicated at the start, this assignment is due on Thursday, Dec. 10th before 12:00 MIDNIGHT (by 11:59 pm).

End of Assignment 12
Maintained by Natalie Webber