Deadline: Thu September 1 2022 before 11:59 PM

CSE-271: Object-Oriented Programming

Homework #1: Point Processor

Due: Thu September 1 2022 before 11:59 PM

Maximum Points: 18

Delayed (by no more than 24-hours) submissions earn only 80% credit

Key objectives of this project are:

- Review fundamentals of problem solving from CSE-174
- Recap Java programming using an IDE like Eclipse
- Review and adhere to CSE department's Style guide
- Use Javadoc to document methods and their return values
- Solve programming problems that use `ArrayList`
- Implement method(s) with specified parameter and return types
- Use text file and console I/O in a program

Submission Instructions

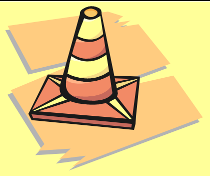
This homework assignment must be turned-in electronically via Canvas CODE plug-in. Ensure your program compiles successfully, without any warnings or style errors. Ensure you have documented the methods. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload just the `SearchHelper.java` source file onto Canvas via the CODE plug-in.

1. The 1 Java source file developed for this part of the homework.

General Note: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Preliminaries: It would be helpful for you to review `ArrayList` concepts, taught in CSE-174 prior to commencing work on this project. It may also be useful to setup the CSE department's style checker in Eclipse to ease checking your style as you develop your program.

Grading Rubric:



The program submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet basic requirements or just skeleton code will be assigned zero score! Programs that do not compile, **have even 1 method longer than 25 lines**, or just some skeleton code will be assigned zero score.

- **NOTE:** Violating CSE programming style guidelines is an error! Your program should not have any style errors.
- **Delayed submission: Only 80% points:** Submission delayed **by no more than 24-hours** will be accepted for a partial credit of maximum 80% of the points.
- **Formatting & Documentation: 5 points** – Reserved for good Javadoc style documentation for each method you implement. If your Javadoc is low quality, then you lose points in this category.

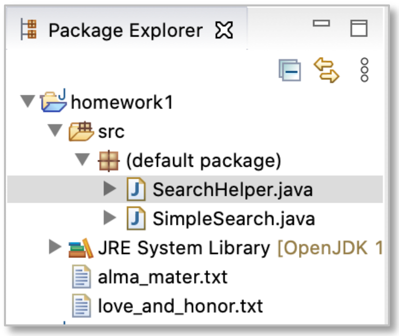
Project Overview

Your assignment is to complete a program that reads a set of lines from a given text file and searches the lines for a given *phrase*. The search phrase can be a simple substring or a substring with wild-card characters (i.e., '?'). A sample run/output of this program appears at the end of this document.

For this project, you have been provided with the following files:

1. **`love_and_honor.txt`**, **`alma_mater.txt`**: These are simple text files from where lines are to be loaded into an `ArrayList<String>` by the `SearchHelper.loadLines` method.
2. **`SimpleSearch.java`**: This file contains the `main` method and a `menu` method that are used for testing. Ideally, you should not modify this file. You will not be submitting it as your solution is required to work with the starter code.
3. **`SearchHelper.java`**: This file contains basic starter code for the methods you will be documenting and implementing as part of this project. This will be the only file you will be submitting for this project.

Directions to setup Eclipse project:



1. Create an Eclipse project called `homework1`.
2. Download the supplied starter code to your project –
   a. `SimpleSearch.java` and `SearchHelper.java` should be dragged to the `src` folder
   b. Place `love_and_honor.txt` and `alma_mater.txt` should be dragged to the project folder as shown in the adjacent figure.
3. Implement the methods in `SearchHelper.java` as described further below. You may write additional helper methods if you wish; but none of the methods should exceed 25-lines of code (excludes comments)
4. Ensure you use Javadoc to add documentation for each method. Ensure you add comments to the body of the method as needed.
5. Ensure you check style periodically to learn style – good style should become a habit!
6. As you write each method, run the program and test your methods. Check values via the debugger!

Descriptions for methods in `SearchHelper.java`

You are required to implement the methods in `SearchHelper.java` to accomplish the following functionality:

Documentation: Each of the methods below (including parameters and return values) should be properly documented using Javadoc style documentation. The method should have comments in the body of the method elucidating pertinent implementation details. [**3 points**]

Minimum required functionality (base cases) to earn any points:

1. **`ArrayList<String> loadLines(final String fileName)`**: This method must read all of the lines points from a given text file into an `ArrayList` and return the list. Lines must be returned in the same order as they appeared in the given text file. This method must use a `Scanner` and associated methods (`hasNextLine`, `nextLine`, etc.) to read lines from the given text file. **Tip:** See `Scanner` examples from prior exercise(s) for examples on using a Scanner to read data from a given text file. [**4 points**].

2. **`void print(final ArrayList<String> lines)`**: This method must print all the lines, one at a time, with the index of each entry before it. If `lines` is empty then this method just prints `"Empty List"`. For example, the first couple lines would be formatted as shown below [**3 points**]:
```
[0]: Love and honor to Miami,
[1]:
[2]: Our college old and grand.
[3]: Proudly we shall ever hail thee,
…
```

The following two features are additional features:

3. **`ArrayList<String> search(final ArrayList<String> srchList, String phrase)`**: Returns a new list of strings that contains all lines that match a given search *phrase*. The returned list of strings must be in the same order as they were in the given `srchList`. If no lines match, then this method returns an empty list (and not `null`).

   The search phrase must be handled in the following manner (both features can be combined into one implementation and that is very good):

   a. Simple phrase: In the simple case, this method can assume the `phrase` is a substring and only return lines that contain the given substring. [**3 points**]

   b. Wild-card phrase: In this case, the `phrase` can contain wild-card characters, i.e., `'?'`s. A wild-card character matches any character at that position. For example, the `phrase` `"t?e"` would match strings "**`the`**", "`assor`**`tme`**`nt`", "`be`**`tte`**`r`", "`coun`**`tle`**`ss`", "`diversi`**`tie`**`s`", etc. [**5 points**]

<u>**Notes for the wild-card phrase feature**</u>:

- This feature may require some thought and problem-solving – *i.e.*, you may need some help from your instructor. So, the key here is to <u>be proactive and start early</u> so that you <u>give yourself time to seek help</u>, if needed. In other words, one of the expected outcomes of this part of the project is to <u>motivate you to start early and be proactive about seeking help</u> as needed.
- For full points, this feature must be implemented without using other libraries or using regular expressions. If you use regular expressions (i.e., methods like `split` that accept a pattern or use the `Pattern` class) then only 2 points (out of the possible 5 points) will be assigned.

Sample outputs:

Note that user inputs are show in green color

Required (base case) operation:

```
Enter text file name to load lines: love_and_honor.txt
Loaded 9 lines from love_and_honor.txt

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 1
[0]: Love and honor to Miami,
[1]:
[2]: Our college old and grand.
[3]: Proudly we shall ever hail thee,
[4]: Over all the land.
[5]: Alma mater now we praise thee,
[6]: Sing joyfully this lay.
[7]: Love and honor to Miami,
[8]: Forever and a day.

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 3
```

Additional Feature example tests:

Simple Phrase Test #1:

```
Enter text file name to load lines: love_and_honor.txt
Loaded 9 lines from love_and_honor.txt

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 2
Enter search phrase (can contain wild-card characters): the
[0]: Proudly we shall ever hail thee,
[1]: Over all the land.
[2]: Alma mater now we praise thee,

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 3
```

Simple Phrase (empty list):

```
Enter text file name to load lines: love_and_honor.txt
Loaded 9 lines from love_and_honor.txt

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 2
Enter search phrase (can contain wild-card characters): blah
Empty list

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 3
```

Wild-card Phrase Test #1:

```
Enter text file name to load lines: love_and_honor.txt
Loaded 9 lines from love_and_honor.txt

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 2
Enter search phrase (can contain wild-card characters): ?h?
[0]: Love and honor to Miami,
[1]: Proudly we shall ever hail thee,
[2]: Over all the land.
[3]: Alma mater now we praise thee,
[4]: Sing joyfully this lay.
[5]: Love and honor to Miami,

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 3
```

Wild-card Phrase Test #2:

```
Enter text file name to load lines: alma_mater.txt
Loaded 22 lines from alma_mater.txt

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 2
Enter search phrase (can contain wild-card characters): ?he?
[0]: Thou hast watched the decades roll.
[1]: While thy sons have quested from thee,
[2]: Sturdy-hearted, pure of soul.
[3]: Weave the story of the glory,
[4]: Our Miami, here's to thee.
[5]: Thou the calm, and they the storm,
[6]: Thou didst give them joy in conquest,
[7]: Strength from thee sustained their arm.
[8]: Crimson tow'rs against the sky;

What to do next:
1. Show all lines
2. Find lines with given phrase
3. Quit
Enter your choice: 3
```

Deadline: Thu September 1 2022 before 11:59 PM

Submission:

This homework assignment must be turned-in electronically via Canvas CODE plug-in. Ensure your program compiles successfully, without any warnings or style errors. Ensure you have documented the methods. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload just the `SearchHelper.java` source file onto Canvas via the CODE plug-in.