Com S 227
Spring 2020
Miniassignment 1
40 points

Due Date: Friday, October 9, 11:59 pm (midnight)

5% bonus for submitting 1 day early (by 11:59 pm Oct 8)

10% penalty for submitting 1 day late (by 11:59 pm Oct 10)

No submissions accepted after October 10, 11:59 pm

General information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus for details.

You will not be able to submit your work unless you have completed the *Academic Dishonesty policy acknowledgement* on the Homework page on Canvas. Please do this right away.

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Note: This is a miniassignment and the grading is automated. If you do not submit it correctly, you will receive at most half credit.

Overview

This is a short set of practice problems involving writing loops. You will write seven methods for the class **`mini1.LoopInMySoup`**. All of the methods are static, so your class will not have any instance variables (or any static variables, for that matter). There is a constructor, but it is declared **`private`** so the class cannot be instantiated.

For details and examples see the online javadoc.

You do *not* need arrays or ArrayLists for this assignment, though you will not be penalized for using them.

Advice

Before you write any code for a method, work through the problem with a pencil and paper on a few concrete examples. Make yourself write everything down; in particular, write down things that you need to remember from one step to the next (such as indices, or values from a previous step). Try to explain what you are doing in words. Write your algorithm in pseudocode.

The other key problem-solving technique is to try solving *part* of the problem, or solving a *related, simpler problem*. For example:

- If you have trouble with **`countMatchingChars`**, can you
    a. iterate over a string and print out the character at each position i?
    b. do the same with *two* strings that are the same length, printing out the two characters at the same position i
    c. do the same, but also print "boo" if the two characters are the same?
    d. count how many times you printed "boo"?
    e. ignore extra characters if one string is longer than the other?

- If you have trouble with **`newtonCount`**, can you
    a. write (and test) a sequence of statements that perform the given steps for the algorithm just *once*? (That is, can you get 5.5 from the initial value x = 10?) How about *twice*?
    b. write a loop that always just repeats the steps 5 times?
    c. write a condition to check whether a number squared is within .0000001 of another number (Tip: it could be above or below, so use **`Math.abs()`**.)

- If you have trouble with **`isFibonacciTypeSequence`**, can you
    a. parse the string and just print the numbers one at a time?
    b. parse the string and print each number, and also the previous number?
    c. parse the string and print each number, and also the previous *two* numbers?
    d. do the same, but print "boo" if the previous two numbers add up to the current one?

- If you have trouble with **`removeRuns`**, can you
    a. iterate over a string and just print the characters one at a time?
    b. iterate over a string and append each character to a new string?
    c. iterate over a string and append each character to a new string only if it is not the letter 'p'?
    d. iterate over a string, and append each character to a new string only if it *isn't* the character you just appended?

- If you have trouble with **`getRun`**, can you
    a. return just the character at index **`start`**?
    b. return the character at index **`start`**, plus all the characters after it?
    c. return the character at index **`start`**, plus the ones after it that match it?

- If you have trouble with **`countSubstrings`**, can you
    a. determine whether t occurs *once* as a substring of s? (*Tip*: use the String method **`indexOf`**)
    b. determine whether t occurs *twice* as a substring of s? (As an example: if you find "bc" as a substring of "abcdbcde", what string do you want to check for the second occurrence? You just want to look in "dbcde" now. The String methods **`indexOf`** and **`substring`** will help you here.)
    c. What's different if you allow overlap? If you have "aa" and "aaaaa", try step (b) with overlap allowed, and with overlap not allowed.

- If you have trouble with **`mergeWithRuns`**, can you
    a. do a problem like **`removeRuns`**, where you have to create a new string by appending a character in each iteration?
    b. merge two strings into one string when all characters are distinct, and both strings are the same length (like "abc" and "xyz" become "axbycz")
    c. do the same, but handle one string being longer ("abcde" and "xyz")?
    d. do the same, but each time you append a character to the result, check whether there are more of the same character ("abcccde" and "xyz")?
    e. (*Tip*: would it be helpful to have a method that would always grab the whole run starting from a given index? See method **`getRun`**!)

My code's not working!!

Developing loops can be hard. If you are getting errors, a good idea is to take a simple concrete example, and trace execution of your code by hand to see if the code is doing what you want it to do. You can also trace what's happening in the code by temporarily inserting **`println`** statements to check whether variables are getting updated in the way you expect. (Remember to remove the extra **`println`**'s when you're done!) But overall, the best way to trace through code with the debugger, as we are practicing in Lab 5. Learn to use the debugger effectively, and it will be a lifelong friend.

Always remember: one of the wonderful things about programming is that within the world of your own code, you have absolute, godlike power. If the code isn't doing what you want it to do, you can decide what you really want and make it so. **You are in complete control**!

(If you are not sure what you *want* the code to do, well, that's a different problem. Go back to the "Advice" section.)

I have an infinite loop!!

Lucky for you, infinite loops are pretty easy to track down. See the new link #12 on our Canvas front page, which provides some tips and advice.

How do I make a string with a loop, as needed for `removeRuns`?

Start with an empty string and concatenate an additional character in each iteration. For example, here is one way to create the reverse of a given string:

```
public static String reverse(String s)
{
  String result = "";  // start with empty string
  for (int i = s.length() - 1; i >= 0; i = i - 1)
  {
    result += s.charAt(i); // add on characters one at a time
  }
  return result;
}
```

Alternatively you could iterate over the characters left to right, and append them on the *left*:

```
public static String reverse(String s)
{
  String result = "";
  for (int i = 0; i < s.length(); i = i + 1)
  {
    result = s.charAt(i) + result;
  }
  return result;
}
```

As an aside, experienced Java programmers would probably use a `StringBuilder` object:

```
private static String reverse(String s)
{
  StringBuilder sb = new StringBuilder();
  for (int i = s.length() - 1; i >= 0; i = i - 1)
  {
    sb.append(s.charAt(i));
  }
  return sb.toString();
}
```

The SpecChecker

A SpecChecker will posted for this assignment that will perform a few dozen functional tests. However, when you are debugging, it is usually helpful if you have simple test cases of your own.

Remember that to call a static method, you prefix it with the *class* name, not with an object reference. For example, here is simple test case for the **`removeRuns`** method:

```
import mini1.LoopInMySoup;

public class LooperTester1
{
  public static void main(String[] args)
  {
    String test = "abbbccd";
    String result = LoopInMySoup.removeRuns(test);
    System.out.println(result);
    System.out.println("Expected abcd");
  }
}
```

You can save yourself from having to type "**`LoopInMySoup`**" over and over again by using the Java feature **`import static`**:

```
import static mini1.LoopInMySoup.removeRuns;

public class LooperTester2
{
  public static void main(String[] args)
  {
    String test = "abbbccd";
    String result = removeRuns(test);
    System.out.println(result);
    System.out.println("Expected abcd");
  }
}
```

Since no test code is being turned in, you are welcome to post your tests on Piazza for others to use and comment on.

Documentation and style

Since this is a miniassignment, the grading is automated and in most cases we will not be reading your code. Therefore, there are no specific documentation and style requirements. However, writing a brief descriptive comment for each method will help you clarify what it is you are trying to do.

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`miniassignment1`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`miniassignment1`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Java examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have Java code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it. Be sure you have stated a specific question; vague requests of the form "read all my code and tell me what's wrong with it" will generally be ignored.

Of course, the instructors and TAs are always available to help you. See the Office Hours section of the syllabus to find a time that is convenient for you. We do our best to answer every question carefully, short of actually writing your code for you, but it would be unfair for the staff to fully review your assignment in detail before it is turned in.

What to turn in

Please submit, on Canvas, the zip file that is created by the SpecChecker. The file will be named **`SUBMIT_THIS_mini1.zip`**. and it will be located in the directory you selected when you ran the SpecChecker. It should contain one directory, **`mini1`**, which in turn contains one file, **`LoopInMySoup.java`**. Always LOOK in the zip file the file to check.

Submit the zip file to Canvas using the Miniassignment1 submission link and verify that your submission was successful. If you are not sure how to do this, see the document "Assignment Submission HOWTO" which can be found linked on the Canvas front page.

We strongly recommend that you just submit the zip file created by the specchecker. If you mess something up and we have to run your code manually, you will receive at most half the points.

> We strongly recommend that you submit the zip file as created by the specchecker. If necessary for some reason, you can create a zip file yourself. The zip file must contain the directory **mini1**, which in turn should contain the file **`LoopInMySoup.java`**. You can accomplish this by zipping up the **src** directory of your project. The file must be a zip file, so be sure you are using the Windows or Mac zip utility, and not a third-party installation of WinRAR, 7-zip, or Winzip.