Com S 227
Spring 2019
Miniassignment 1
40 points

Due Date: Monday, March 11, 11:59 pm (midnight)

5% bonus for submitting 1 day early (by 11:59 pm March 10)

10% penalty for submitting 1 day late (by 11:59 pm March 12)

No submissions accepted after March 12, 11:59 pm

General information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus, http://www.cs.iastate.edu/~cs227/syllabus.html#ad , 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.

Also note: we expect to have one more miniassignment before break, which should be available by March 9, which is prior to the due date for this one. So start early!

Overview

This is a short set of practice problems involving writing loops. You will write eight methods for the class **`mini1.FromLoopToNuts`**. 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. None of the problems requires nested loops.

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.

Another key problem-solving strategy is to try solving *part* of the problem, or solving a *related, simpler problem*. For example, here are some ideas for getting started on each problem using this strategy:

- To start solving **`countMatches`**, can you
    a. iterate over a string and just print the characters one at a time?
    b. given two strings the same length, print the characters of *both* strings one at a time?
    c. repeat (b) but print "boo" each time the strings have the same character at some index i?
    d. figure out the largest index that is valid in both strings?

- To start solving **`isArithmetic`**, can you
    a. parse the string and just print the numbers one at a time? (*Tip: use a Scanner, and call the method* **`setDelimiter(",")`** *to split the tokens at commas)*
    b. determine whether the string contains two or more values?
    c. find the difference between the first two values?
    d. parse the string and print out the difference between each value and the previous one?

- To start solving **`eliminateRuns`**, 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 onto a new string? *(Tip: see the section below on creating a string with a loop)*
    c. iterate over a string and append each character onto a new string, only if it isn't already the last character of the new string?

- To start solving **`threeInARow`**, can you
  a. write a loop to generate random numbers until you get a 7?
  b. write a loop to generate random numbers until you get a value that matches the previously generated number?

- To start solving **`findEscapeCount`**, can you
  a. write statements to update a and b for just one iteration? *(See the javadoc for some sample values.)*
  b. do it for two iterations? *(See the javadoc for some sample values.)*

- To start solving **`differByOneSwap`**, can you
  a. iterate over a string and print each character one at a time?
  b. iterate over a string and print out each pair of adjacent characters?
  c. do the same for two strings?
  d. do the same for two strings, and print "boo" whenever the two pairs have the same characters in the opposite order?
  e. count the number of times that (d) happens?

- To start solving **`countSubstringsWithOverlap`**, can you
  a. use the **`indexOf`** methods to check whether one string is a substring of another?
  b. given strings s and t, if t occurs at index i in s, make a substring containing everything in s that is *after* index i?
  c. do (b) in a loop until t does not occur?

- To start solving **`countSubstringsWithOverlap`**, can you
  a. use the **`indexOf`** methods to check whether one string is a substring of another?
  b. given strings s and t, if t is a substring of s, make a substring containing everything in s that is *after* the occurrence of t?
  c. do (b) in a loop until t does not occur?

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 (as illustrated in section 6.2 of the text) 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!)

Overall, the best way to trace through code with the debugger, as we are practicing in Lab 6. 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.)

What do you mean, a private constructor?

Just put this declaration at the top of your class:

```
/**
 * Private constructor disables instantiation.
 */
private FromLoopToNuts() { }
```

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

Start with an empty string and concatenate additional characters 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;
}
```

As an aside, experienced Java programmers would probably use a **`StringBuilder`** object, which works like a mutable type of string:

```
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();
}
```

Testing and the SpecChecker

A SpecChecker will posted shortly that will perform an assortment of functional tests. As long as you submit the assignment correctly, your score will be exactly the score reported by the specchecker.

However, when you are debugging, it is much more helpful if you have a simpler test case of your own that reproduces the error you are seeing.

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 **`countMatches`** method:

```
import mini1.FromLoopToNuts;

public class SimpleTest
{
  public static void main(String[] args)
  {
    int result = FromLoopToNuts.countMatches("abcde", "xbydcazzz");
    System.out.println(result);
  }
}
```

You can save yourself from having to type "**`FromLoopToNuts`**" over and over again by using the Java feature **`import static`**, which allows you to invoke a static method without typing the class name:

```
import static mini1.FromLoopToNuts.*;

public class SimpleTest
{
  public static void main(String[] args)
  {
    int result = countMatches("abcde", "xbydcazzz");
    System.out.println(result);
  }
}
```

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.

Any posts from the instructors on Piazza that are labeled "Official Clarification" are considered to be part of the spec, and you may lose points if you ignore them. Such posts will always be placed in the Announcements section of the course page in addition to the Q&A page. (We promise that no official clarifications will be posted within 24 hours of the due date.)

What to turn in

Note: You will need to complete the "Academic Dishonesty policy questionnaire," found on the Homework page on Blackboard, before the submission link will be visible to you.

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, **`FromLoopToNuts.java`**.

Always LOOK in the zip file the file to check what you have submitted!

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 in the Piazza pinned messages under "Syllabus, office hours, useful links."

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 **`FromLooptoNuts.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.

Optional reading: What's the point of that method `findEscapeCount`?

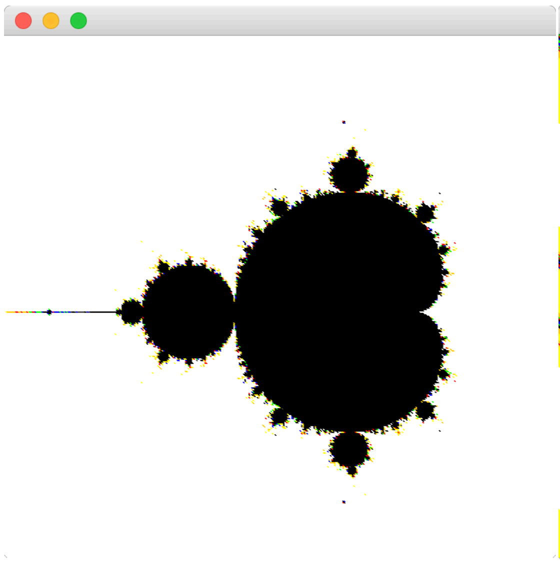
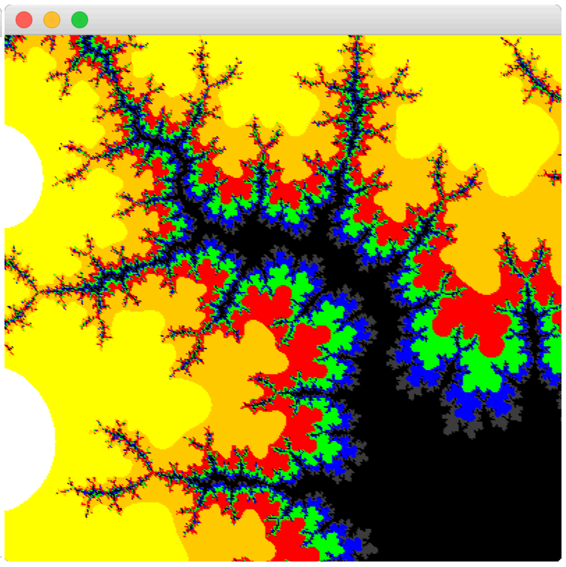
This section is not part of the assignment, but could be fun to investigate. It describes a graphical application that can be used to visualize the effect of your **`findEscapeCount`** *method.*

The short algorithm you are implementing in **`findEscapeCount`** defines a representation of what is known as the *Mandelbrot set*, which can be loosely defined[1] as the set of points (x, y) for which the value $\sqrt{a^2 + b^2}$ is always less than or equal to 2, no matter how many times the given steps are repeated. If this value ever goes over 2, we say it has "escaped" the bound. To approximate this set of (x, y) pairs, we pick a maximum number of iterations (say 100) and see whether the value escapes within that many iterations. If not, we assume it is part of the set. The number of iterations it actually takes to escape for a given (x, y) is an interesting piece of information, and the number is often used to study the *boundary* of the Mandelbrot set, which exhibits a detailed self-similar or "fractal" structure (looks the same no matter how much you zoom in). People typically create a graphical representation using different colors to depict different ranges of escape values.

To experiment with what this means, take a look at the application MandelbrotTest.java. This is a simple GUI that depicts the Mandelbrot set, based on your code for findEscapeCount. Points that are in the set are colored black, and points on the boundary are given a color depending on the escape value. You can select an area with the mouse to repeatedly zoom in. The initial screenshot on the left depicts the whole set, in a 3 x 3 region at the origin. The image on right is zoomed in to an area of roughly .005 x .005 on the top of the upper "bulb". You can edit the

[1] More precisely, it's the set of *complex* numbers $c = x + yi$ such that the modulus of z_n never exceeds 2 in any number of iterations of the steps $z_{n+1} = z_n^2 + c$. In our case we're doing the same arithmetic using the notation $c = x + yi$ and $z = a + bi$ (where i is a symbol representing the square root of -1).

CUTOFFS array near the top of the file to change the way the colors are assigned (see the getColor method).

The GUI code is based on the Java Swing libraries. This style of programming is specialized (not to mention somewhat tedious) and we don't cover it in this course. However, you might want to investigate it on your own. Much of what you find about Swing on the internet is out of date or incorrect. The official Oracle tutorial, however, is detailed and accurate,

http://docs.oracle.com/javase/tutorial/uiswing/start/index.html

and there is a also simpler collection of examples on Steve's web page,

http://web.cs.iastate.edu/~smkautz/

(scroll down to "Other stuff").