P06 Palindromes

Overview

A **palindrome** reads the same regardless of whether the letters are written forward or backward. Some examples include "rotator", "racecar", and even phrases like "do geese see god" (palindromic phrases tend to allow for creative placement of spaces).

This week's program is a recursive exploration of palindromes, but we're going to keep things very simple: all you'll do is work through the letters of the alphabet in a mirrored sequence. You're tired, we're tired; let's keep things as straightforward as we can. We're not even going to do any object-oriented programming this time. Everything is familiar and static.

Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **03/22**.
15 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
20 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.
10 points	**Manual Grading Feedback**: TAs or graders will manually review your code to verify use of recursion. If you implement your methods iteratively instead of recursively, you *will* lose *all* of these points.

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

Learning Objectives

The goals of this assignment are:

- Familiarize yourself with recursive problem solving techniques.
- Relax a little. We can't give you a spring break, but we can try to give you a spring hopefully-a-little-easier.

Additional Assignment Requirements and Notes

Keep in mind:

- You are allowed to define any **local** variables you may need to implement the methods in this specification.
- You are allowed to define any **private static** helper methods you may need to help implement the methods in this specification.
- All methods, public or private, must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide.
- Any source code provided in this specification may be included verbatim in your program without attribution.
- You may not have ANY import statements. This is a very simple program.

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Academic Conduct Expectations and Advice, which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - What do I do about hardware problems?
  - and more!
- Course Style Guide, which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project in Eclipse, called something like **P06 Palindrome**.
   a. Ensure this project uses Java 11. Select "JavaSE-11" under "Use an execution environment JRE" in the New Java Project dialog box.
   b. Do **not** create a project-specific package; use the default package.

2. Create two (2) Java source file within that project's src folder:
   a. **Palindrome**.java (does NOT include a main method)
   b. **PalindromeTester**.java (includes a main method)

All methods in this program will be **static** methods, as this program focuses on procedural programming.

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

Implementation Requirements Overview

Your **Palindrome** class must contain the following four (4) static methods, which **must** be implemented recursively:

- `public static String mirrorA(char start) throws IllegalArgumentException`
  - Recursively create a simple alphabet pattern, starting at the provided character, moving backward to the beginning of the alphabet, and then forward again to the provided letter, separating each letter with a space.
    - If start is `'E'`, the method should return the string `"E D C B A B C D E"`
  - This method is only valid for capital letter input; if anything other than a capital letter is provided as an argument, throw an IllegalArgumentException with a descriptive error message.

- `public static String mirrorA(char start, int step) throws IllegalArgumentException`
  - Recursively create an alphabet pattern, starting at the provided character, and moving back and forth to the beginning of the alphabet by steps of size `step`.
    - If start is `'E'` and step is 1, the method should return the same string as `mirrorA(start)`.
    - If start is `'E'` and step is 2, the method should return `"E C A C E"`
    - If start is `'E'` and step is 3, the method should return `"E B B E"`
    - And so on.
  - As before, the method is only valid for capital letter input and *strictly positive* (not zero or negative) step sizes. For invalid input, throw an IllegalArgumentException with a descriptive error message.

- `public static String mirrorZ(char start) throws IllegalArgumentException`
  - Recursively create a simple alphabet pattern, starting the provided character, and moving forward to the end of the alphabet, and then backward again to the provided letter, separating each letter with a space.
    - If start is `'V'`, the method should return the string `"V W X Y Z Y X W V"`
  - This method is only valid for capital letter input; if anything other than a capital letter is provided as an argument, throw an IllegalArgumentException with a descriptive error message.

- `public static String mirrorZ(char start, int step) throws IllegalArgumentException`
  - Recursively create an alphabet pattern, starting at the provided character, and moving forward and back to the end of the alphabet by steps of size `step`.
    - If start is `'V'` and step is 1, the method should return the same string as mirrorB(end).
    - If start is `'V'` and step is 2, the method should return `"V X Z X V"`

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

        - If start is `'V'` and step is 3, the method should return `"V Y Y V"`
        - And so on.
    - As before, the method is only valid for capital letter input and *strictly positive* (not zero or negative) step sizes. For invalid input, throw an IllegalArgumentException with a descriptive error message.

Some hints:

- Remember that char primitives are like ints -- you can add and subtract integer values from them, and they will change value in alphabetical order.
- The ASCII value of `'A'` is 65, and the ASCII value of `'Z'` is 90.
- The expression `'A' == 65` evaluates to true, as does `'B' > 'A'`.

Your **PalindromeTester** class must contain AT LEAST:

- `public static boolean testMirrorA()`
- `public static boolean testMirrorAStep()`
- `public static boolean testMirrorZ()`
- `public static boolean testMirrorZStep()`
    - These methods should test valid AND invalid input against expected results, and must NOT throw exceptions.

- `public static boolean runAllTests()`
    - This method must call ALL of your test methods and return true if and only if all methods return true. If you add additional methods besides the four listed above, be sure to call them here.
- `public static void main(String[] args)`
    - The only line in this method should be a call to the runAllTests method.

Be aware that we will be testing your test methods on code which contains errors! Your methods should be able to detect errors in code and return false in those cases.

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

Implementation Details and Suggestions

Make sure you're implementing the methods in Palindrome.java *recursively*. You may wish to create an iterative (looping) implementation in your tester class to compare results, but if you don't write your mirror methods using recursion, you will lose all of the manual grading points.

Approaching Recursion

To design a recursive method, I recommend starting with the base case(s):

1. For the non-step methods, the base case is just that the char argument is 'A' or 'Z'.
2. For the step methods, you have two base cases: either the argument is 'A' or 'Z', or it's exited the alphabet and isn't a capital letter anymore.

Figure out what the result String will look like for these cases, and implement that.

Then work on the recursive case: add the current character to either end of the recursive string of the other characters.

Testing

We've given you some sample output for valid inputs that you're welcome to use, but we encourage you to come up with some other cases and be sure to test invalid inputs, too. Remember that your test code should be able to detect errors!

©2021 Mouna Ayari Ben Hadj Kacem and Hobbes LeGault — University of Wisconsin–Madison

Assignment Submission

Hooray, you've finished this CS 300 programming assignment!

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct and style guide requirements, submit your source code through Gradescope.

For full credit, please submit ONLY the following files (source code, *not* .class files):

- Palindrome.java
- PalindromeTester.java

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default this will be your most recent submission.

Copyright Notice

This assignment specification is the intellectual property of Mouna Ayari Ben Hadj Kacem, Hobbes LeGault, and the University of Wisconsin–Madison and may not be shared without express, written permission.

Additionally, students are not permitted to share source code for their CS 300 projects on any public site.