Programming: Week 19 Lab (Strings)

Learning objectives:

- Using the Java **String** and **StringBuffer** classes.

Resources

- Week 19 Lecture Slides
- Java documentation:
  https://docs.oracle.com/javase/7/docs/api/java/lang/String.html
  https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuffer.html

Download the files **Week19Main.java** and **MyStringUtils.java**, and make a project in Eclipse containing these two files (create a new project, then use Import > FileSystem).

MyStringUtils.java defines a simple string utilities class, but the method implementations are empty (your task in this lab will be to implement the methods).

Week19Main.java defines a simple console program which runs tests on the methods in the **MyStringUtils** class. As it stands, some of the tests will pass, some will fail. By the time you have completed the exercises, they will all pass.

The methods, and their descriptions:

String Reverse(String in)	Returns a new String which is the reverse of the String **in.** For example, **Reverse**("abcd") returns "dcba".
String AlphabetOnly(String in)	Returns a new string which contains only the alphabetical characters ('a' – 'z' and 'A' – 'Z') from the String **in**, in the same order in which they appear in the input. For example, **AlphabetOnly**("! Abc ") returns "Abc".
boolean IsPalindrome(String in)	Returns **true** if the String **in** is a palindrome, **false** otherwise. We define a palindrome as a string which reads the same forwards and backwards, ignoring case, punctuation, spaces and any non-alphabetical characters. A palindrome must have at least one alphabetical character. Example **IsPalindrome**("Madam, I'm Adam") returns **true.**

Exercise 1

Implement the method **Reverse**. When you are finished, running the application should result in the first four tests passing.

Hints:

- You could use a **StringBuffer** to build up the reversed string, then return a **String** constructed from this **StringBuffer**.
- Alternatively, you could work entirely in the **String** class, using concatenation.
- Read the documentation carefully, you may find a useful method.

Exercise 2

Now implement **AlphabetOnly.**

Hints:

- Again, you could do this using **String** or **StringBuffer** – in this case, concatentating **Strings** or using **StringBuffer.append.**
- Remember that the lower case and upper case characters are in order, and use the comparison and boolean operators to build a test for an alphabetical character.

Exercise 3

Implement the final method, **IsPalindrome**. You should use the first two methods, along with any other **String** methods you need to implement the specified functionality. When you are done, the code will pass all the tests in Week20Main.java.

Extension Exercise

This is a problem from Project Euler (projecteuler.net). This a website with hundreds of programming and maths problems. Some require more maths than others. Many are programming problems which are impossible to solve with brute force so a clever solution is required. They range from fairly straightforward to extremely difficult, research-level problems (none are easy). It can be addictive.

https://projecteuler.net/problem=4

Largest palindrome product

Problem 4

A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009 = 91 × 99.

Find the largest palindrome made from the product of two 3-digit numbers.

Hint: you can use the string representation of an integer, String.valueOf(int i), along with a modified version of your palindrome checking function to test if a number is palindromic.