Programming Assignment 6-4

There is a Java library class called `java.util.Arrays`; it has a static method `sort`, which will arrange in sorted order any array of objects that have a natural ordering, like `Integers` and `Strings`.

Example: Obtain `Integer[] arr = new Integer[6]`. Place values in `arr`:

```
arr[0] = 1,  arr[1] = -2,  arr[2] = 4,  arr[3] = 3,  arr[4] = 11,  arr[5] = -8
```

Now you can sort `arr` with the call

```
Arrays.sort(arr);
```

The original array `arr` is then sorted.

If, however, the objects in an array are not naturally ordered, or if you want a different ordering from the usual one, then `Arrays.sort` expects you to pass in a `Comparator` (see the end of Lesson 6 slides). The signature of this version of `sort` when applied, for example, to `String`s is:

```
void sort(String[] arr, Comparator<String> comp)
```

For this exercise, create a class `StringSort` with a constructor

```
StringSort(Comparator<String> myComparator)
```

which sets the value of `myComparator` as an instance variable in `StringSort`. `StringSort` also has a method

```
public String[] stringSort(String[] arr)
```

which makes use of the `Comparator` stored as an instance variable to sort the given input array using `Arrays.sort`; it then returns the array in its sorted order.

For this exercise, create a `Comparator` for `Strings` in three different ways:

1. Create a separate class `StringLengthComparator` that defines a new order relationship on `Strings` as follows: For any `String`s s1, s2, declare that `s1` is "less then" `s2` if length of `s1` is less then length of `s2`. With this logic, two `String`s will be considered "equal" if they have the same length. (Better ways of doing this kind of thing will be discussed in Lesson 8.)

   Create a `main` method in a separate class `Main` which creates an array of test `String`s (any `String`s you like), instantiates `StringSort`, passing in an instance of your new `StringLengthComparator`, and invokes the `stringSort` method using the test `Strings` as input, and then prints the result to console.

2. Create a class `MainAnonymous` whose `main` method will pass to the constructor of `StringSort` an instance of an anonymous class that has the same functionality as your `StringLengthComparator`. Like the `main` method of `Main`, it invokes the `stringSort` method using test `Strings` as input and then prints the result to theconsole. Output should be the same as for part 1.

3. Create a class `MainLambda` whose `main` method will pass to the constructor of `StringSort` a lambda expression that has the same functionality as your `StringLengthComparator`. Like the main method of `Main`, it invokes the `stringSort` method using test `Strings` as input and then prints the result to the console. Output should be the same as for part 1.