Experimenting with the String class.

Consider the following code:

```
String s = "Sara";

String r = s;

String t = new String(s);
```

The above code has the corresponding object diagram

Since r and s are both referring to the same String object, we say that r and s are aliases, or that r is an alias of s (or vice versa). And since t refers to a copy of the String object "Sara", it is neither an alias of r nor s. In general, any change made to the copy won't be reflected on the original object. This is very often the motivation for making a copy of an object. However, this argument has no value when discussing Strings. The String class doesn't have one single method that modifies the String object it is applied to. For instance, the method toLowercase() applied to a String creates a new String whose content is the original String all lowercase. This is why the following code prints ABC twice (and **not** ABC and abc)

```
String s = "ABC";

System.out.println(s);

s.toLowercase();

System.out.println(s);
```

To get “abc” to be printed, we need to write

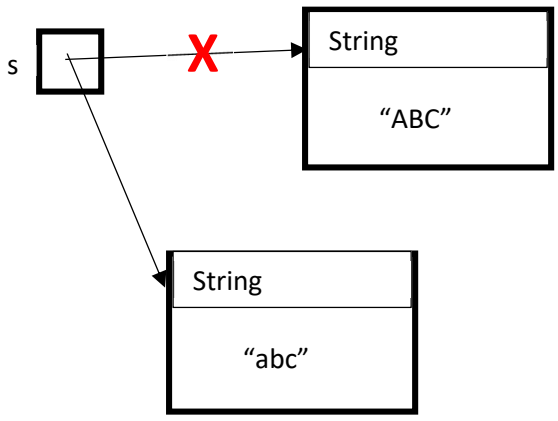
```
String s = “ABC”;

System.out.println(s);

s = s.toLowercase();

System.out.println(s);
```

The key point is the assignment s = s.toLowercase(), which is illustrated by the following object diagram.



Note that the String object “ABC” can’t be accessed after s is pointing to the String “abc” and will be eventually garbage collected.

String objects are said to be immutable, meaning that once created a String object can’t be modified.

You might wonder why the String class was made immutable. It is to allow for the use of aliases without having to worry about possible modifications of the object via the alias.

A word of caution: if in a program, you need to create a String and then modify it many times by for instance adding characters to it, prefer a class other than the String class such as the StringBuffer class. Using the String class will trigger many object creations and deletions, which might hinder the performance of your program. The StringBuffer class is mutable and allows object modifications.