COLBY COLLEGE
Computer Science Department

CS151 **Allen B. Downey**
Fall 1999

Assignment 7: Strings and things

Due: Wednesday 27 October

Experiments!

In lecture we talked about various ways `chars`, `Strings` and `ints` get combined, but we haven't quite filled in the details.

1. Create a new project named `Test.prj`, based on the `cs151HelloStationery`.
2. In `main`, write some simple expressions that combine various types using the `+` operator. For example, what happens when you "add" a `String` and a `char`? Does it perform addition or concatenation? What is the type of the result? (How can you determine the type of the result?)
3. Make a big copy of the following table and fill it in. At each intersection, you should indicate what operation is performed (addition or concatenation), and what the type of the result is.

	boolean	char	int	String
boolean				
char				
int				
String				

4. Think about some of the choices the designers of Java made when they filled in this table. How many of the entries seem reasonable, as if there were no other reasonable choice? How many seem like arbitrary choices from several equally reasonable possibilities? How many seem stupid?

   You don't have to write answers to any of those questions, but you can if you want.
5. Read Section 7.9.1. Type in the code there and see if you can figure out what is going on.

Documentation

1. Go to the following web page:

   `http://java.sun.com/products/jdk/1.1/docs/api/packages.html`
2. Click on `java.lang`. This is the package that contains many of the built-in Java classes.
3. Click on `String`. A pretty long page will appear containing some general information about Strings along with documentation of *all* the String methods. Print this page and add it to your three-ring binder of class materials. Highlight the methods we have used so far.

Palindromes

1. Create a new project named Palindrome.prj based on cs151HelloStationery.
2. Type in the method named `first` from last week, the one that takes a String and returns the first letter.
3. Write a method named `last` that takes a String and returns the last letter. Test this method.
4. Write a method named `middle` that takes a String and returns a substring that contains everything *except* the first and last characters.

   Hint: use the substring method. Here is the documentation:

   ```
   public String substring (int beginIndex, int endIndex)

   Returns a new string that is a substring of this string. The substring
   begins at the specified beginIndex and extends to the character at
   index endIndex - 1.

   Parameters:
   beginIndex - the beginning index, inclusive.
   endIndex - the ending index, exclusive.

   Returns: the specified substring.
   ```

   Try to figure out what they mean by "inclusive" and "exclusive." Run a couple of experiments to see if you are right.
5. A palindrome is a word that reads the same both forward and backward, like "otto" and "palindromeemordnilap." An alternative, recursive definition of a palindrome is "either an empty string or a word whose first letter is the same as the last, and whose middle is a palindrome."

   Write a recursive method named `isPalindrome` that takes a `String` and that returns a boolean indicating whether the word is a palindrome or not.
6. On a piece of paper, figure out a strategy for checking palindromes iteratively. Think about what variables to use and what role they will play. Show me your plan before you proceed.
7. Once I have ok'ed your plan, write it up in Java.

Names

When names are recorded in a computer, they are sometimes written with the first name first, like "Allen Downey," and sometimes with the last name first and a comma, like "Downey, Allen." That can make it difficult to compare names and put them in alphabetical order.

A related problem is that some names contain capital letters in funny places. When computers compare characters, they usually put all the capital letters before the lower-case letters. We discussed the reasons for that in class, but the consequence is that computers often put names with capitals in the wrong order.

A good solution to this problem is to convert all names to a common format so that they can be compared using the built-in `compareTo` method. In this homework we will do that conversion. Later we will use this technique to sort a list of names.

1. Create a new project called `Name.prj` based on the `cs151HelloStationery`.

2. Write a method called `hasComma` that takes a name as an argument and that returns a boolean indicating whether it contains a comma. If it does, you can assume that it is in last name first format.

3. Write a method called `convertName` that takes a name as an argument. It should check whether it contains a comma. If it does, it should just return the string.

   If not, then it should assume that the name is in first name first format, and it should return a new string that contains the name converted to last name first format.

4. Write a method called `compareName` that takes two names as arguments and that returns -1 if the first comes before the second alphabetically, 0 if the names are equal alphabetically, and 1 otherwise.

   You will want to use the `compareTo` method, so you should read the documentation of that method carefully. Test it out with some simple cases to be sure you understand how it works.

What to turn in

1. As usual, please take a minute to indent your code correctly, remove obsolete code, and add any comments you think are appropriate. Print your programs and show them to me. If possible, take a minute to make the code presentable; for example, you might want to circle important methods or write the names next to them, or whatever you can do to make it easier for me to look at them.