Useful methods of the String class

For the second assignment we ask you to do some string manipulation. For this you must work with some of the methods provided by the String class in Java.

Remember, every string is an object of the String class. To invoke a method on an object, you specify: `theObject.methodName()`. Some of the useful (for this assignment) methods in the String class are:

```
length()
substring(int beginIndex)
substring(int beginIndex, int endIndex)
toUpperCase()
toLowerCase()
```

Let's work with this variable:

```
String courseName = null; // declare the variable and set to null
```

The `null` keyword means there is no object yet for the variable to refer to. If we want to know if an object variable is null we can test with an if statement.

```
if (courseName == null) {
   System.out.println("Course name not valid");
}
```

Let's give it something to refer to:
`courseName = "COMP 1409";`

The `length()` method returns an int corresponding to the number of characters in the string. `courseName.length()` will return the number `9`. You can assign the return value to an appropriate variable, e.g.
`int lengthOfCourseName = courseName.length();`

The overloaded `substring()` methods return a substring of the original string. The parameters are index positions in the string, with 0 being the index of the first character.

`courseName.substring(5)` will return the string "`1409`" because the "1" is the character in position 5 in the string (counting from zero) and this version of the method returns everything from position 5 to the end of the string.

`courseName.substring(2, 7)` will return the string "`MP 14`" because the "`M`" is in position 2 and the "`4`" is in position 6. The character in position 7 is not included.

The methods `toUpperCase()` and `toLowerCase()` return the uppercase or lowercase version of the original string. Non-alphabetic characters in a string are unchanged.
`courseName.toLowerCase()` will return the string "`comp 1409`".

Keep in mind that these String methods return a new String object. They do not alter the original. The easiest way to work with them is to use local variables to hold the return values, e.g.

```
String lastPart = courseName.substring(5);
String lowerName = courseName.toLowerCase();
```

One of the assignment requirements is this:

The first letter of each name component must be returned from the accessor methods as uppercase and the rest of the letters must be lowercase, no matter how the names are passed to the constructor.

Here's one way to handle this problem. Write a method that formats a name. Use it for any name that needs formatting.

```
/** Formats the input string and returns it with the first letter in
upper case and the rest in lower case /*
private String formatName(String name)
{
   // pick out the first letter and make it uppercase
   String firstLetter = name.toUpperCase().substring(0, 1);
   // make all the other letters lowercase
   String theRest = name.toLowerCase().substring(1);
   // put the two parts together and return
   return firstLetter + theRest;
}
```

Now if you pass a name to this method you will get back the properly-formatted name, e.g.

```
// declare a variable and assign it a name
String name = "cOLleeN";

//call the format method inside an output statement
System.out.println("Formatted name: " + formatName(name));
```

or

```
// call the format method and assign the return back to the variable
name = formatName(name);
```