STRING REVIEW: LENGTH() & SUBSTRING()

public String first3Letters(String str)

Given a String, return **ONLY** its **First 3** letters.

Review:
You've had lots of practice using **substring()** to extract the first or last letters of a string.

String str = "ABCDEF";

str.substring(**0,1**) creates a substring with just the 1st letter: **"A"**
str.substring(**1,2**) creates a substring with just the 2nd letter: **"B"**
str.substring(**0,2**) creates a substring with the first 2 letters: **"AB"**
str.substring(**1,4**) creates a substring with the second 3 letters: **"BCD"**

str.substring(**1**) creates a substring without the first letter: **"BCDEF"**
str.substring(**2**) creates a substring without the first 2 letters: **"CDEF"**
str.substring(**3**) creates a substring without the first 3 letters: **"DEF"**

Notice that the **number** of characters you extract is the **difference** between the 2 parameters of substring(), e.g.
str.substring(**2,5**) extracts **5**-**2**=**3** characters.

If you want to extract the final characters, you need to know the length, e.g.

String str = "TUVWXYZ";
int len = str.length();

str.substring(**len-1**) creates a substring with just the last letter: **"Z"**
str.substring(**len-2**) creates a substring with just the last 2 letters: **"YZ"**
str.substring(**len-4**) creates a substring with just the last 4 letters: **"WXYZ"**

str.substring(**len-2,len-1**) creates a substring with just the 2nd-to-last letter: **"Y"**
str.substring(**len-3,len-2**) creates a substring with just the 3rd-to-last letter: **"X"**
str.substring(**len-5,len-2**) creates the substring: **"VWX"**

str.substring(**0,len-2**) creates a substring without the last 2 letters: **"TUVWX"**
str.substring(**0,len-3**) creates a substring without the last 3 letters: **"TUVW"**

The parameters for substring have to comply with the 2 rules below:
(1) They must be in the range **0** through **len**. That is, they cannot be **negative** and they cannot be **greater than len**.
(2) If you are using 2 parameters, the 2nd must be **greater than or equal to** the 1st - it **cannot be smaller**.

STRING REVIEW: LENGTH() & SUBSTRING()

Now to solve the problem. Normally you would solve it as follows:

```
public String first3Letters(String str) {
  String s = str.substring(0,3);
  return s;
}
```

But suppose the **input** parameter has **FEWER** than 3 letters.
For example: "AB" or "A".
When you run the program with these inputs, it will "blow up" when it tries to execute the line:
String s = str.substring(0,3);
because the 2nd parameter **3** violates the rule that both parameters have to be in the range(0,len).

The way to solve this problem is use an **if** statement that will
check whether the string is long enough to use in the substring statement.

```
public String first3Letters(String str) {
  int len = str.length();
  if (len >= 3) {
    String s = str.substring(0,3);
  }
  return s; // ERROR: Can't find s !
}
```

Notice the ERROR: **s** has now become a local variable in the body of the **if** statement.
You have to declare it **outside the curly brackets** so that the **return** statement can see it:

```
public String first3Letters(String str) {
  String s = "";
  int len = str.length();
  if (len >= 3) {
    s = str.substring(0,3);
  }
  return s;
}
```

Now if the string is shorter than 3 letters, the method will return the empty string.
However, the specs for the method don't tell us what to do in this case.
Here is a new spec:

public String first3Letters(String str) Given a String, return ONLY its First 3 letters.
If the string has fewer than 3 letters, just return the string itself.

This is easy to implement. Simply initialize **s** with **str** rather than the empty string ("").

```
public String first3Letters(String str) {
  String s = str;
  int len = str.length();
  if (len >= 3) {
    s = str.substring(0,3);
  }
  return s;
}
```