String Processing

9-17-2002

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have questions about assignment #2?
- If you are interested, the ACM programming team meetings are Thursday's at 4:30. Send me an e-mail if you want to be on the e-mail list.

Comments about Assignment #2

- The design for assignment #2 is due on Thursday. Again I would like for you to generate the HTML documentation and post that then send me a link.
- It needs to include your screen implementation and the implementations of the main blocks that you will use. You can add this onto the project you did for assignment #1 or create a new one, but you will need a class with main in it.
- I would like for you to include @param and @return tags in the javadoc comments for methods.

Coding for Assignment #2

- When you write the code you need to construct your screen and block classes. You can use the BasicBlock code to help with your blocks. You also need to edit the GameSetup code to that it uses your version of Screen instead of BasicScreen.
- Let's look at the Screen interface to see what methods you need to create.

The String Class

- One of the most fundamental types in any real programming language is that of a string of characters. There is a String class in java.lang that fills this role for Java.
- Today we are going to look at that class and what it can do. I want you to bring up the JavaDoc for it in the Sun API web page.

Constructors

- Looking at the JavaDocs, you can see that there are many different constructors for String including a copy constructor and versions that create Strings from arrays of chars and bytes.
- There is also a constructor that creates a String from an object of type StringBuffer. We'll discuss the StringBuffer class a bit later.

Notes on Immutability

- The String class is immutable. As a result, once the constructor has been invoked, none of the methods will change the value of that string.
- Some methods might look like they would change the value, but what they actually do is return a new String object with the appropriate alterations.

Concatenation, the Overloaded '+'

- Java doesn't allow you to overload operators and doesn't overload many operators itself. One exception to this was the decision to allow the '+' operator to do string concatenation.
- In addition to being able to concatenate strings, you can use it with a string and a numeric type or an Object. More on this in two slides.

Indexing into Strings

- Like most things in C-family languages, Strings in Java are zero referenced. So the elements of a String str have indexes between 0 and str.length()-1, inclusive.
- This indexing is used for methods like charAt, indexOf, substring, and others.
- Note that the two argument substring specifies that the second index is the one AFTER the last character you want.

Conversions to String

- The String class has a number of static methods with the name valueOf. These take different argument types, and return strings that represent the given arguments.
- For the primitive types you can sometimes get more control over these types of conversions using the methods provided in their wrapper classes.

The StringBuffer Class

- If you have a "String" that you need to change the value of on a regular basis, you probably want to use an instance of StringBuffer instead of String.
- Go the the page for the StringBuffer API. Note that the methods like append and insert return a StringBuffer making one thing they behave similar to String. Read the description of the return parameter to see this is not the case.

Identity vs. Equality

- While the '+' operator has been overloaded for Strings, the '==' operator hasn't. This operator checks for equality of references, not the contents of what they reference.
- To check for equality of the string contents you need to use the equals() method. There is also a compareTo method that can be used, but it doesn't return a boolean.

StringTokenizer

- The java.util package has a class called StringTokenizer that can be quite helpful when you want to parse a string into pieces.
- You construct it with a String to be tokenized and an optional list of separators. It then lets you pull off tokens one at a time.

Let's Code Some

- Now we are going to play around some more with the code we started last class and see where we might be able to use Strings in it.

Minute Essay

- Write me a small method with the following signature:
  - `int countOccur(String s,char c);`
- This method should return how many times the given character occurs in the String.
- Remember that the design for assignment #2 is due on Thursday. Also read chapter 8 for Thursday's class.