A "string" to a computer scientist is a sequence of characters. (Characters are single letters, symbols, or punctuation marks.) Some examples of strings are "ABC", "1.23", "123", "$", and even "". In Java, strings are implemented as objects, instances of the String class, but in many ways strings act unlike other objects. For example, you can instantiate a string without using the keyword new.

s

hello

String s = "hello"; is short for String s = **new** String("hello");

The only overloaded operator for objects in Java is the “+” sign, used for string *concatenation*. "Concatenate" means to paste together. The three commands shown below produce the string "Concatenation".

String str = "Concat";

str

str = str + "enat";

Concatenation

str += "ion";

If you want to compare strings for equality, do not use ==. Instead, use the equals method. Whenever you compare objects, you must use equals. Any object can call equals because there is a default implementation in Object. The concrete class, in this case String, overrides that equals by defining an equals that is appropriate to strings, i.e., the letters match.

In the case above, str.equals("Concatenation") returns true, and str.equals(s) returns false. Unfortunately for you, str == s is legal and it compiles. (It compares the hexadecimal addresses of the two objects, which in this case returns false, because the objects are stored in different hex addresses.)

The case of the letters matter, so that the string "Hello" and the string "hello" are not the same. s.equals("hello") returns true but s.equals("Hello")returns false.

You could also compare them the other way around, because string literals are objects themselves. For instance, "hello".equals(s) returns true but "Hello".equals(s) returns false.

Strings are *immutable,* meaning that once a string object is created it can never be changed, which is equivalent to saying that strings do not have set methods. In the concatenation example above, it looks like there is only one string, but actually there are three different immutable string objects that exist at one time or another: "Concat", "Concatenat", and "Concatenation". The str reference changes to point to three different objects.

As we said above, strings are sequences of characters. Characters are single letters, symbols, or punctuation marks, and are stored in primitive type char variables. Java denotes a char value with single quotation marks, such as 'a', '>', or '.'. Every String object has a private field holding an array of characters. You should picture String s as a reference pointing to a String object that stores an array of characters, like this:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s |  | **String** |  |  |  |  |  |  |  |
|  |  | myChars |  | 'h' | 'e' | 'l' | 'l' | 'o' |  |
|  |  |  |  | [0] | [1] | [2] | [3] | [4] |  |

The reason programmers go to the trouble of making classes is not only to store data but also to provide methods to work on that data. Java’s String class has many methods which are all, of course, listed in the String API. Be sure to look up equals, equalsIgnoreCase, and compareTo.

Exercises

test

greeting

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| H | o | w |  | a | r | e |  | y | o | u | ? |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

Given the strings:

String test = "abcde";

String greeting = "How are you?";

What is the output of the following code fragments?

1. System.out.println(greeting + " " + test);

2. System.out.println("ab\ncd\te");

3. System.out.println(""+test.length());

4. System.out.println(test.charAt(2));

5. System.out.println(test.indexOf("c"));

6. System.out.println(greeting.substring(4));

7. System.out.println(greeting.substring(0, 4));

1. System.out.println(test.substring(0,1));

9. System.out.println(test.substring(2,3));

10. String s1 = greeting.substring(0,8);

String s2 = s1.substring(4);

String s3 = greeting.substring(4,7);

System.out.println(s2 + s3);

11. System.out.println(test.toUpperCase());

12. System.out.println(test.equals("abcd"));

13. System.out.println(test.equals("ABCDE"));

14. System.out.println(test.equalsIgnoreCase("ABCDE"));

15. What type of argument is required by the String method charAt()?

16. What is the return type of the String method charAt()?

17. How do you change a char into a String?

18. How do you change a single-character String into a char?