Computer Science I (Java) – CSC 130 – Duke Hutchings

Day 14 Topics

- The **for** loop
- Strings

For Loops

For Loops

```
for (int i = 0; i < 4; i++) {
  rect(x,y,w,h);
  x = x + w;
}


// here, the i variable does not
// exists after the loop ends
```

(Fixed) While Loops

```
int i = 0;
while (i < 4) {
  rect(x,y,w,h);
  x = x + w;
  i++;
}
```

Strings

A String is a way to store and manipulate text in Java programs.

```
String s = "CSC 130";
```

Strings have a special status in Java:
although they are objects, they are treated like variables.

- we do not use the **new** keyword to make a new String.
- BUT, we call lots of methods on strings

String Indices

An **index** is an integer that refers to a specific **position** in a string.
The first character in a String has an **index** of 0
(like the first pixel in the drawing space is at 0, 0)

```
String s = "CSC 130";
// indices: 0123456
```

The index of the first C is 0 and the index of the second C is 2.
The index of 3 is 5 and the index of 0 is 6.

The index of H is -1. The index of 4 is -1. (i.e. -1 means not found)

String Method: indexOf

```
String s = "CSC 130";
// indices: 0123456
int i = -10;
i = s.indexOf("S", 0); // i is 1
i = s.indexOf("C", 0); // i is 0
i = s.indexOf("C", 1); // i is 2
i = s.indexOf("C", 2); // i is 2
i = s.indexOf("C", 3); // i is -1
i = s.indexOf("SC", 0); // i is 1
i = s.indexOf("130", 0); // i is 4
i = s.indexOf("c", 0); // i is -1 case matters
```

String Method: indexOf

indexOf is nit-picky: the characters have to be identical, including the case of letters.

```
String s = "elon Elon ELON";
// indices: 01234567890123
//                    10
int i = -10;
i = s.indexOf("elon", 0); // i is  0
i = s.indexOf("Elon", 0); // i is  5
i = s.indexOf("ELON", 0); // i is 10
```

String Method: indexOf

indexOf is nit-picky: the characters have to be identical, including the case of letters and number of spaces.

```
String s = "NASA   Rover   Mission";
int i = -10;
i = s.indexOf("nasa", 0);  // i is -1
i = s.indexOf("rover", 0); // i is -1
i = s.indexOf("NASA Rover", 0); // i is -1
```

String Method: toLowerCase

When you don't care about case,
use the toLowerCase method to help

```
String s = "NASA   Rover   Mission";
// indices: 0123456789
String slo = s.toLowerCase();
int i = -10;
i = slo.indexOf("nasa", 0);  // i is 0
i = slo.indexOf("rover", 0); // i is 7
i = slo.indexOf("nasa rover", 0); // i is -1
```

String Method: indexOf

In general,

- if s is a string you want to search through
- if t is a string you are looking for inside of s
- if k is the index where you want to start looking

```
int p = s.indexOf(t, k);
```

Then p is the index where t **first** occurs in s at or after index k.
p will have a value of -1 if t is not found inside of s.

String Method: indexOf

Special case:

- if s is a string you want to search through
- if t is a string you are looking for inside of s
- if you want to search from the beginning of s

```
int p = s.indexOf(t);
```

Then p is the index where t **first** occurs in s.
p will have a value of -1 of t is not found inside of s.

String Method: substring

Often when working with strings, we desire to create a copy of some portion of the string. This copied portion is called a **substring**.

```
String s = "CSC 130";
// indices: 0123456
String dept = s.substring(0, 3); // CSC
String cnum = s.substring(4, 7); // 130
```

String Method: substring

In general,

- if s is a string and you need to copy a portion of s
- if i is the index of the start of the portion
- if j is **the next character index** of the end of the portion

```
String t = s.substring(i, j);
```

Then t is a copy of the portion of s you want.

String Method: substring

Special case:

- if s is a string and you need to copy a portion of s
- you want just the character at index i in s

```
String t = s.substring(i, i+1);
```

Then t is a string that has a copy of the character at index i in s.

String Method: substring

Special case:

- if s is a string and you need to copy a portion of s
- you want n characters of s, starting at index i

```
String t = s.substring(i, i+n);
```

Then t is a string that has a copy of the character at index i in s.

String Method: substring

Special case:

- if s is a string and you need to copy a portion of s
- you want the end portion of s, starting at index i

```
String t = s.substring(i);
```

Then t is the end portion of s (starting from index i)

String Method: length

Sometimes you need to know how many characters are in a string.

```
String s = "elon Elon ELON";
int len = s.length();
```

len now has the value 14.

Strings

Quiz: what value does u have at the end of this code?

```
String s = "CHOPPERZ";
int len = s.length();
String t = s.toLowerCase();
String u = t.substring(1, len-1);
```

Strings: Concatenation

Sometimes you will want to join strings together.

```
String s = "Elon";
String t = "University";
String u = s + t;

// u has the value ElonUniversity
// so how do we fix this?
```

Testing Code; now with labels

Remember this?

```
System.out.println(i);
```

You can do this:

```
System.out.println("value of i: " + i);
```

For Loops + Strings

```
String a = "What does this code do?";
String s = "";
int x = 0;
for (int i = 0; i < a.length(); i++) {
  String c = a.substring(i, i+1);
  if (c.equals(" ")) {
    x++;
  }
  else {
    s = s + c;
  }
}
System.out.println("x is " + x);
System.out.println("s is " + s);
```

Lab: Codingbat Problems (Java String-1)

makeTags	warmup of concatenation
left2	warmup of substring
right2	intermediate (harder version of left2)
hasBad	warmup of indexOf
seeColor	warmup of indexOf
twoChar	intermediate
endsLy	intermediate
frontAgain	intermediate
without2	intermediate (but copy/paste frontAgain and call it)
conCat	tricky
withoutX	tricky