Strings Manipulation

Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu).
Updated by Marcia Moraes (marcia.moraes@colostate.edu)

Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 11 (zybooks) – you should have already done that ☺
- Lab 07
- Reading 12 (zyBooks) - you should have already done that ☺
- Lab 08
- Reading 13 (zybooks) - you should have already done that ☺
- RPA 6

Keep practicing your RPAs in a spaced and mixed manner ☺

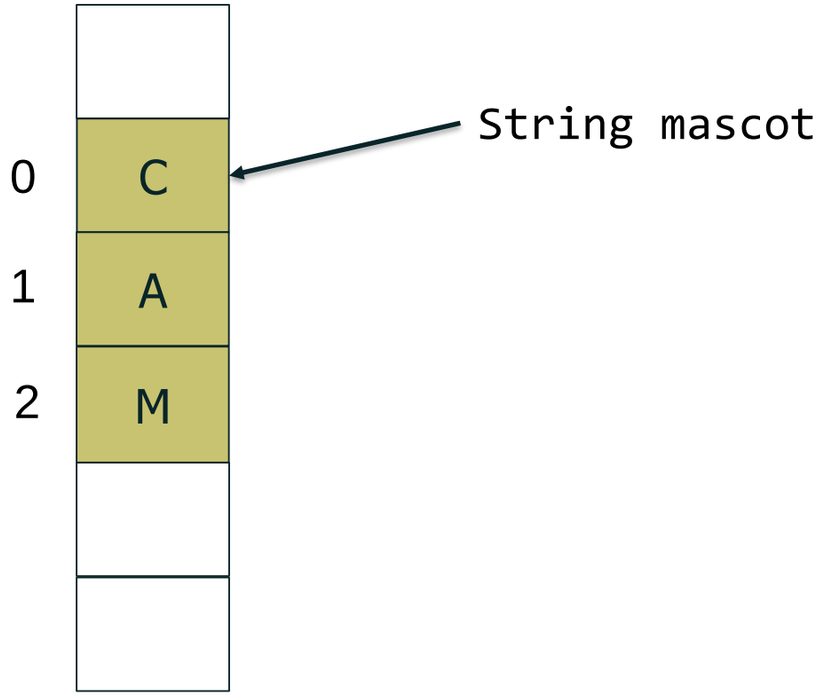
https://www.quotespedia.org/authors/h/henry-ford/whether-you-think-you-can-or-think-you-cant-youre-right-henry-ford/

Recall Activity

- Explain what is a String, describe at least three methods you can use over Strings and how those methods work.

Strings

- A String is a collection of ordered characters
  - It has data
  - It has functionality (methods)
  - It is also **immutable** (can't be directly modified)
    - Every method that builds a String, returns a copy
    - Java does this for memory efficiency
- Example
  - `String mascot = "CAM";`



Common Strings Methods

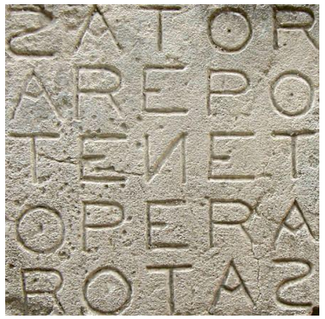
- `.charAt(int)` - gives us the character at location
- `.indexOf(char)` - gives us the location of character (what you just wrote!)
- `.indexOf(String)` - overloaded option, gives the location of the *start* of the string that matches
- `.indexOf(char, int)` or `.indexOf(String, int)` - same as above, but changes starting location
- `.lastIndexOf(char)` - gives us the index starting at the end working down (also has String version)
- `.substring(int start, int end)` - returns the substring from start - including start, to end, excluding end. (inclusive/exclusive)
- `.toLowerCase()` - returns the lowercase version of the String
- `.toUpperCase()` - returns the uppercase version of the String

Finding the index

0	k
1	i
2	n
3	n
4	i
5	k
6	i
7	n
8	n
9	i
10	k

stops searching, returns 1

stops searching, returns 2



Sator Square is a 2D palindrome

What are the outputs for the following instructions?

```
String palindrome  = “kinnikinnik”;

palidrome.length(); // returns 11

palidrome.charAt(0);  // return ‘k’

palidrome.charAt(palidrome.length()-1); // return ‘k’

palidrome.indexOf(‘i’);      // return 1

palidrome.indexOf(‘n’);     // return 2

palidrome.indexOf(‘niki’); // return 3
```

lastIndexOf

0	k
1	i
2	n
3	n
4	i
5	k
6	i
7	n
8	n
9	i
10	k

- `String palindrome  = “kinnikinnik”;`
- `palidrome.lastIndexOf(‘n’);`

What if we wanted the middle ‘k’?

Practice Loop Review

- Write a loop that :
  - Looks at each character in a String (hint: charAt(i))
  - Returns the index of that character (don't use 'indexOf')
    - Return -1 if the loop ends but the character was not found.
  - Now modify it to start at the end of the word, not the start!

Complete the In Class Activity

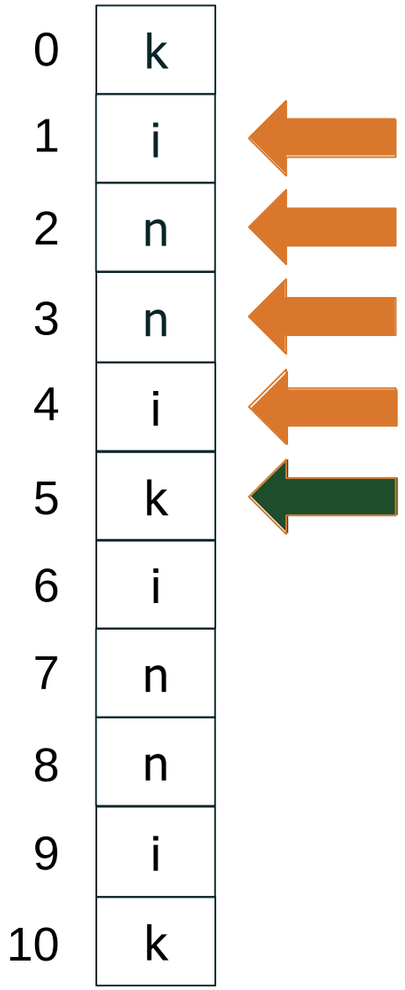
```
public class YourProgram {

 public static int find(String str, char ch) {
      return find(str, ch, 0);
   }


  public static int find(String str, char ch, int start) {
      // LOOP here
      return 0; // change this
   }

   public static void main(String[] args) {
      System.out.printf("TEST: the index is: %d", find("SATOROTAS", 'A'));
      System.out.printf("TEST: the index is: %d", find("SATOROTAS", 'T'));
      System.out.printf("TEST: the index is: %d", find("SATOROTAS", 'Z'));
   }
}
```

indexOf is overloaded



- indexOf can take in two parameters
  - Second parameter is the **start** location

```
String palindrome = “kinnikinnik”;

palindrome.indexOf(‘k’); // returns 0

palindrome.indexOf(‘k’, 1); // returns 5
```

Substring

0	k
1	i
2	n
3	n
4	i
5	k
6	i
7	n
8	n
9	i
10	k

- Returns a portion of the string
- substring(start, end)
  - includes start
  - excludes end! (end is optional, defaults to .length())

```
String palindrome = “kinnikinnik”;

String sub = palindrome.substring(0, 6); // sub is “kinnik”
```

```
int start = palindrome.indexOf(“k”);

int end = palindrome.indexOf(“k”, start+1) + 1;

String sub = palindrome.substring(start, end);
```

Exploring Patterns

- What pattern can be observe in the following Strings?

```
Fort Collins,40°35'6.9288"N,105°5'3.9084"W
Denver,39°44'31.3548"N,104°59'29.5116"W
Boulder,40°0'53.9424"N,105°16'13.9656"W
```

City,Latitude,Longitude

- How can we parse that String to have each one of those elements individually?

Using substring method combined with indexOf method!

Exploring Patterns

- What pattern can be observe in the following Strings?

```
Fort Collins,40°35'6.9288"N,105°5'3.9084"W
Denver,39°44'31.3548"N,104°59'29.5116"W
Boulder,40°0'53.9424"N,105°16'13.9656"W
```

- Example: getting the city

Complete the In Class Activity

```
public class Test {
    public static void main(String args[]){
        String coord = "Fort Collins,40°35'6.9288\"N,105°5'3.9084\"W";
        String city = coord.substring(0, coord.indexOf(","));
        System.out.println(city);
    }
}
```

Substring Practice

- Write a method that returns all characters after a given character
  - Example: sub(“SATOROTAS”, “O”) // return ROTAS
  - Example: sub(“SATOROTAS”, “A”) // returns TOROTAS

- Think about the problem that you need to solve
- Write a sequence of steps to solve that problem – your algorithm
- Translate your algorithm into a Java program

Take-away

Natural Language Processing is about understanding language, Strings express language – Learn more about NLP

- All Strings have indices from 0 to length-1
  - indexOf(char) – finds the index of a character or substring
  - charAt(int) - gives you the character at an index
  - substring(int,int) – gives you a portion of the string
  - length() – don't forget to use it!

Additional Reading

Read more about the String Class : Here
Read more about the Character Class : Here