Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email – Please see syllabus

NEW! Supplemental Instruction: Sign up at my.newpaltz.edu

STRINGS

COMPUTER SCIENCE I

OBJECTIVES

- Strings

STRINGS – AN OBJECT TYPE

- String is an object type, ***not*** a primitive data type.
- Because of this, Strings have methods we can use with them.
- Strings are represented in memory as a **char array**:

String s = “hello”;

- Some useful **methods** for Strings (we'll go through some in the following slides):
  - charAt
  - compareTo
  - equals
  - equalsIgnoreCase
  - indexOf
  - length
  - split
  - substring
  - toLowerCase
  - toUpperCase
- Java docs: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

STRINGS – CHARAT() METHOD

- Unfortunately, you would think you can access the characters of the String using [ ] like with arrays.
  - In Java, you ***can't*** do this. In other programming languages, like Python, you can.

- In Java, we have to use the method, **.charAt()**, to access each letter in a String. Example:

```
String s = "hello"

char c = s.charAt(0);

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

Output:

```
h
```

STRINGS – EQUALS() METHOD

- Another important difference is when checking to see if two Strings are **equal**. Instead of using ==, we use the method, **equals()** with Strings.
  - In simple words, == checks if both objects point to the same ***memory location*** whereas .equals() evaluates to the comparison of ***values*** in the objects.
  - Because of this, unexpected results can occur sometimes if you use == when checking equality with Strings. Thus, you should always use the **equals** method instead.

STRINGS – EQUALS() METHOD

- Example:

```
String s1 = "hello";
String s2 = "hello";

if(s1.equals(s2)) {
    System.out.println(s1 + " equals " + s2);
} else {
    System.out.println(s1 + " DOES NOT equal " + s2);
}
```

Output:

```
hello equals hello
```

STRINGS – LENGTH() METHOD

- The **length()** method gives the amount of characters in a String (spaces included).
- Example:

```
String s1 = "hello";

int len = s1.length();
System.out.println(len);
```

Output:

```
5
```

STRINGS – TOLOWERCASE() AND TOUPPERCASE() METHOD

- The **toLowerCase()** method makes all the characters in a String lowercase. **Example:**

```
String s1 = "HELLO";
s1 = s1.toLowerCase();
System.out.println(s1);
```

Output:

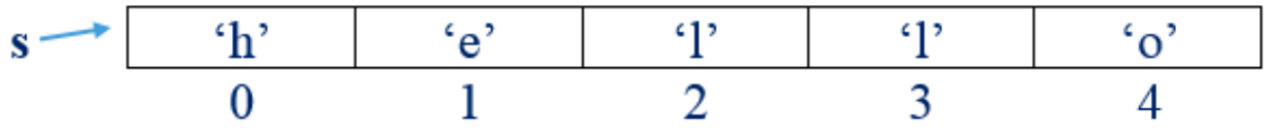
```
hello
```

- The **toUpperCase()** method makes all the characters in a String uppercase: **Example:**

```
String s1 = "hello";
s1 = s1.toUpperCase();
System.out.println(s1);
```

Output:

```
HELLO
```



STRINGS – SUBSTRING() METHOD

substring(int beginIndex)

Returns a new string that is a substring of this string.

substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string.

Example:

```
String s1 = "hello";

String end = s1.substring(3);
System.out.println(end);
```

Output:

lo

Example:

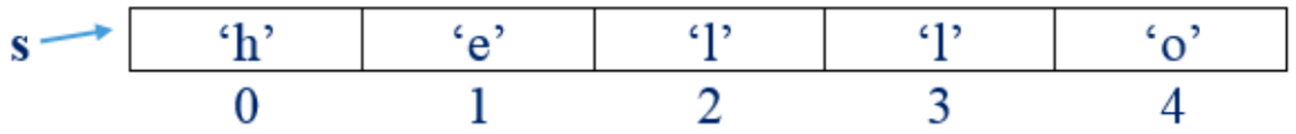
```
String s1 = "hello";

String beg = s1.substring(0,3);
System.out.println(beg);
```

Output:

hel

Important: The endIndex is not included in the substring.



STRINGS – INDEXOF() METHOD

`indexOf``(int ch)`

Returns the index within this string of the first occurrence of the specified character.

`indexOf``(`**`String`**` str)`

Returns the index within this string of the first occurrence of the specified substring.

Example:

```
String s1 = "hello";

int index = s1.indexOf('e');
System.out.println(index);
```

Output:

1

Example:

```
String s1 = "hello";

int index = s1.indexOf("lo");
System.out.println(index);
```

Output:

3

CHARACTER WRAPPER CLASS

- The **Character** wrapper class also has a lot of useful methods. When dealing with Strings, we usually want to check the individual characters.
- Some useful methods:

  toLowerCase()

  toUpperCase()

  isDigit() => checks to see if character is a number

  isLetter() => checks to see if character is a letter

  isLetterOrDigit() => checks to see if character is a letter or number

- Java Docs: https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html

STRINGS — EXAMPLE 1

- Let’s create a method that prints each letter of a String on a new line.

STRINGS — EXAMPLE 1

- Let's create a method that prints each letter of a String on a new line.

```
public class Main {
    public static void main(String[] args) {
        String cs = "programming";
        sepLines(cs);
    }
    public static void sepLines(String s) {
        for(int i = 0; i<s.length(); i++) {
            System.out.println(s.charAt(i));
        }
    }
}
```

Output:

```
p
r
o
g
r
a
m
m
i
n
g
```

STRINGS — EXAMPLE 2

- Let's ask a user for a word and create a method that makes the string have every other letter capitalized starting with the first index. We'll create a new empty String to do this and add to it with concatenation.

- **Example:**

  Give me a word:

  hello there

  HeLlO ThErE

STRINGS — EXAMPLE 2 USING CHARACTER CLASS


```java
import java.util.*;
public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Give me a word: ");
        String word = sc.next();

        String newWord = everyOtherUpper(word);
        System.out.println(newWord);

    }
    public static String everyOtherUpper(String s) {
        String newWord = "";
        for(int i = 0; i<s.length(); i++) {
            if(i%2 == 0) {
                char c = s.charAt(i);
                newWord += Character.toUpperCase(c);
            } else {
                newWord+=s.charAt(i);
            }
        }
        return newWord;
    }
}
```

Output:

```
Give me a word:
icecream
IcEcReAm
```

STRINGS — EXAMPLE 2 USING SUBSTRING


```java
import java.util.*;
public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Give me a word: ");
        String word = sc.next();

        String newWord = everyOtherUpper(word);
        System.out.println(newWord);

    }
    public static String everyOtherUpper(String s) {
        String newWord = "";
        for(int i = 0; i<s.length(); i++) {
            if(i%2 == 0) {
                newWord += s.substring(i, i + 1).toUpperCase();
            } else {
                newWord+=s.charAt(i);
            }
        }
        return newWord;
    }
}
```

Output:

```
Give me a word: planetarium
PlAnEtArIuM
```

STRINGS — EXAMPLE 3

- Ask a user to enter a word that contains a **number**. Create a method that checks to see if a number is actually in the word. If there is a number, print out "Valid" and return true. Else, print, "Not valid", and return false.
  - **HINT:** Use a for loop and check each character using the **isDigit** method from the Character class. **Remember**, you need Character in front of isDigit ==> Character.isDigit(*char*)
- **Example:**

```
Give me a word: password56
Valid
password56 contains number: true
```

STRINGS — EXAMPLE 3


```java
import java.util.*;
public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Give me a word: ");
        String word = sc.next();
        boolean b = containsDigit(word);
        System.out.println(word + " contains number: " + b);

    }
    public static boolean containsDigit(String s) {
        for(int i = 0; i<s.length(); i++) {
            char c = s.charAt(i);
            if(Character.isDigit(c)) {
                System.out.println("Valid");
                return true;
            }
        }
        System.out.println("Not valid");
        return false;
    }
}
```

Output:

```
Give me a word: password56
Valid
password56 contains number: true
```

STRINGS — REMOVING A SUBSTRING

- Let's remove a substring from a string if possible

$$\text{If s} = \text{“hello”} \text{ , s2} = \text{“e”} \implies \text{s3} = \text{“hllo”}$$
$$\text{If s} = \text{“hello”} \text{ , s2} = \text{“a”} \implies \text{s3} = \text{“hello”}$$
$$\text{If s} = \text{“hello”} \text{ , s2} = \text{“lo”} \implies \text{s3} = \text{“hel”}$$

- The general idea here is that we need three parts: **beginning**, **middle** and **end**.

- One or more could be an **empty** string, but we can handle most string manipulations by having this technique of splitting.

  - **Ex**. replace a substring with another
  - **Ex**. replace a substring with capital of itself
  - **Ex**. move substring to front
    And many other problems.

STRINGS — REMOVING A SUBSTRING

- Let's say String ***s = "hello"*** and ***s2 = "el"***
- We want **s3** to be **hlo** (removing s2 out of s)

1. First, we need to check if s2 is in s. If it is, we continue, if not, we just return **s**:

```
public static String partsOfString(String s,String s2){
    if(s.contains(s2)) {



    }
    return s;
}
```

STRINGS — REMOVING A SUBSTRING

- Let's say String ***s = "hello"*** and ***s2 = "el"***
- We want **s3** to be **hlo** (removing s2 out of s)

2. Then, we need to find where **s2** starts in **s** so we know where we need to remove from. We can use the **indexOf** method for this:

```
public static String partsOfString(String s,String s2){
    if(s.contains(s2)) {

        int i=s.indexOf(s2);

    }
    return s;
}
```

What will i = in this example?

1 since el starts at index 1 of hello

STRINGS — REMOVING A SUBSTRING

Let's say String ***s = "hello"*** and ***s2 = "el"***

We want **s3** to be **hlo** (removing s2 out of s)

3. Next, break up the String into **beginning**, **middle**, and **end**. Let's get **beginning** first:

```
public static String partsOfString(String s,String s2) {
    if(s.contains(s2)) {
        int i=s.indexOf(s2);
        String beg=s.substring(0,i);

    }
    return s;
}
```

What will beg = in this example?

h since i=1 and the end index is not included.

STRINGS — REMOVING A SUBSTRING

Let's say String ***s = "hello"*** and ***s2 = "el"***

We want **s3** to be **hlo** (removing s2 out of s)

4. Next, break up the String into **beginning**, **middle**, and **end**.
   Let's get **end** next:

```
public static String partsOfString(String s,String s2) {
    if(s.contains(s2)) {

        int i=s.indexOf(s2);
        String beg=s.substring(0,i);
        String end=s.substring(i+s2.length());   1 + 2 = 3

    }
    return s;
}
```

What will end = in this example?

lo since end will start at index **3** then go to end of String since a second index is not included.

STRINGS — REMOVING A SUBSTRING

Let's say String ***s = "hello"*** and ***s2 = "el"***

We want **s3** to be **hlo** (removing s2 out of s)

5. Next, break up the String into **beginning**, **middle**, and **end**. Let's get **middle** next (Don't need it for this question, but may for another:

```
public static String partsOfString(String s,String s2) {
    if(s.contains(s2)) {
        int i=s.indexOf(s2);
        String beg=s.substring(0,i);
        String end=s.substring(i+s2.length());
        String mid=s.substring(i,i+s2.length());

    }
    return s;
}
```

1 + 2 = 3

1, 3

What will mid = in this example?

el since middle will start at index **1** then go to index **3** but not include index 3

STRINGS — REMOVING A SUBSTRING

Let's say String ***s = "hello"*** and ***s2 = "el"***

We want **s3** to be **hlo** (removing s2 out of s)

6. Lastly, let's return our new String:

```
public static String partsOfString(String s,String s2){
    if(s.contains(s2)) {

        int i=s.indexOf(s2);
        String beg=s.substring(0,i);
        String end=s.substring(i+s2.length());
        String mid=s.substring(i,i+s2.length());
        return beg+end;
    }
    return s;
}
```

hlo will be returned

STRINGS — REMOVING A SUBSTRING

```
public class Main {

    public static void main(String[] args) {

        String s = "hello";
        String s2 = "el";

        String word = partsOfString(s, s2);
        System.out.println(word);

    }
    public static String partsOfString(String s,String s2)
    {
        if(s.contains(s2)) {

            int i=s.indexOf(s2);
            String beg=s.substring(0,i);
            String end=s.substring(i+s2.length());
            String mid=s.substring(i,i+s2.length());
            return beg+end;
        }
        return s;
    }
}
```

Output:

```
hlo
```

EXERCISE

- Write a method that takes two strings as parameters. If the second string occurs in the first string, it should return a string that has the second string capitalized and moved to the end of the string.

Sample Run 1:

```
Give me a String: hello
Give me another String: el
hloEL
```

Sample Run 2:

```
Give me a String: hello
Give me another String: aa
hello
```

EXERCISE

Write a method that takes two strings as parameters. If the second string occurs in the first string, it should return a string that has the second string capitalized and moved to the end of the string.

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Give me a String: ");
        String s1 = sc.next();
        System.out.print("Give me another String: ");
        String s2 = sc.next();

        String word = weirdString(s1, s2);
        System.out.println(word);
    }
    public static String weirdString(String s1, String s2) {
        if(s1.contains(s2)) {
            int i = s1.indexOf(s2); //find where s2 is in s1
            String beg = s1.substring(0,i); //end index, i, is not included with substring
            String end = s1.substring(i + s2.length()); //this will 'skip' over s2
            String mid = s1.substring(i, i+s2.length()); //get s2
            return beg + end + mid.toUpperCase();

        } else {
            return s1;
        }
    }
}
```

Sample Run 1:

```
Give me a String: hello
Give me another String: el
hloEL
```