Topic 4.0: Strings & File I/O

© Lauren Himbeault 2024

Learning Goals (Week 4):

- Uses, throw, and catch built-in exceptions.
- Understand and use Finally block when handling exceptions.
- Order of operations in a try/catch/finally block
- Write code that can read and write text files
- Write code that uses data from a file to instantiate objects
- **trim() & split() functions**
- **charAt() & substring()**
- **Additional helpful String manipulation expressions**

© Lauren Himbeault 2024

- **Remember:** Strings are **Immutable ->** Strings need to be returned a variable somewhere to “save” the changes

```
System.out.println(s.toLowerCase( )); //s not changed
s = s.toLowerCase( ); //changes s
```

© Lauren Himbeault 2024

Practice String Methods

- Check to see if a String is a palindrome
- Reads the same forward and backward
- We want to do this while ignoring non-letters, leading/trailing space, and case
- e.g.
  - “tacocat”
  - “nurse runs”
  - “ Was it a car or a cat I saw? ”

```
s.replace(char search, char newC); // replace search with newC
s.replaceAll(String reg, String r); // replace anything matching reg with r
s.trim(); // remove all leading.trailing whitespace
```

© Lauren Himbeault 2024

Practice String Methods

- Check to see if a String is a palindrome
- Reads the same forward and backward
- We want to do this while ignoring non-letters, leading/trailing space, and case
- e.g.
  - “tacocat”
  - “nurse runs”
  - “ Was it a car or a cat I saw? ”

```
String1.indexOf(String2); // position or -1
```

- Finds the position of the first appearance of String2 within String1
- If not there at all, -1 is the result

© Lauren Himbeault 2024

Practice String Methods

- Check to see if a String is a palindrome
- Reads the same forward and backward
- We want to do this while ignoring non-letters, leading/trailing space, and case
- e.g.
  - “tacocat”
  - “nurse runs”
  - “ Was it a car or a cat I saw? ”

```
String1.charAt(int);
```

- returns the character at a position
- Position must exist or you will get an error!

- **Now we can solve this problem!**

© Lauren Himbeault 2024

More String Methods

- Concatenation: + (we've already seen that) `String s = " hello " + "world";`
- compareTo: which is first "in alphabetical order" (dictionary order)
  - "less" or "greater" is determined by the numeric codes (ASCII or Unicode) (a value of 0 means the words are the same)
  - **Be careful: uppercase is always less than lowercase**

```
int value = String1.compareTo(String2);
```

- Substring
  - start in the first position in the string
  - end is the position to stop at (not including)
  - so you get characters at the positions starts to (end-1)
  - If you do not specify an end (str1.substring(start)), the substring goes to the end

```
String s = String1.substring(int start, int end);
```

© Lauren Himbeault 2024

More String Methods

- Split()

```
String[] tokens = String1.split("\\s+");
```

- String[] split(String) will break up a String into an array of Strings at the positions of the given substring.
  - i.e. What character, symbol , etc. (regex) do you want to split on?

- "Test 12 34".split(" ") gives {"Test", "12", "34"}

- “Hello friends   what’s up?”.split("\\s+") will split on any whitespace of any kind

- The parameter is actually a “regular expression”, in the form of a String. Beware!

© Lauren Himbeault 2024

Pause & Practice

- Validate an Email address:
- An email address has the form:
  - name@part1.part2.part3...
- Write a function that checks that an email inputted is valid (1 @ sign, no more than 3 characters after the last '.', etc. Pick your own validation rules and code it up!

- Validate a phone number
- 204-555-1111
- Write a function that checks that a phone number inputted is valid
- no more than 10 numbers
- IF an area code is present, it should have 3 numbers
- each part separated by '-' should have the correct number of characters

© Lauren Himbeault 2024