Cody Schliebe

Week 04 Research

Backend April 2023

Strings

(all string research gathered from String (Java SE 17 & JDK 17) (oracle.com) unless otherwise noted)

String methods are super helpful for many situations, and I have found that I used many before I even knew what they were. They are great tools for finding certain characteristics of strings, manipulating strings, and creating new strings. In the first few weeks, I found myself using some of these methods when I didn't know how to do something that an assignment asked for—I would instead find a method that would accomplish the same task and learn how to use it. This was cheese, but also helpful for when we actually started using these methods.

1.

endsWith()

a) String.**endsWith(String suffix)**
b) This method returns a Boolean value if the string ends with the string specified in the method's parameters.
c) If you were trying to determine if a string were a question, you could check to see if it ends with a question mark.

```
String prompt = sc.next();

boolean isQuestion = prompt.endsWith("?");
```

isEmpty()

a) String.**isEmpty()**
b) This method returns a boolean value based on whether or not the string has a length of 0. If it does, it is empty and the method returns a true value.
c) If you have an array of strings and are trying to store a value in a string that isn't already used, you could run a loop to check the strings for an open slot.

```
String msg = "Very important message";
```

```
for (int i = 0; i < bookOfMessages.length(); i++) {
        if (bookOfMessages[i].isEmpty()) {
                bookOfMessages[i] = msg;
        }
}
```

*source: [Java String isEmpty() Method (w3schools.com)]

repeat()

a) String.**repeat(int count)**
b) This method will create a string from a string concatenated with itself *count* number of times.
c) If you wanted to create artwork that used the String "Hello World!" as the background repeating forever, you could do something like this:

```
String helWor = "Hello World! "
String backGround = helWor.repeat(1200);
System.out.println(backGround);
```

toCharArray()

a) String.**toCharArray()**
b) This method will take a string and create an array with each element being a character from the original string.
c) Let's say you wanted to create a method that went through strings and found how many times the letter 'e' was present. You could do something like this:

```
public static int howManyEs(String word) {
        Char[] testLetters = word.toCharArray();
                int count = 0;
                for (char let : testLetters) {
                        if (let.equals('e')) {
                                count += 1;
                        }
```

```
                }
                return count;
        }
```

*source: [Converting String to "Character" array in Java - Stack Overflow]

toLowerCase()

a) String.**toLowerCase()**
b) This method will take a string and convert all alpha characters into lowercase.
c) This would be good if you are checking an input of a case-insensitive username against a list:

```
String userInput = sc.next();
boolean userAccepted = false;
for (i = 0; i < userName.size(); i++) {
    if userInput.toLowerCase() == userName[i] {
            System.out.println("Username accepted. Please enter password.";
            userAccepted = true;
            break;
    }
    break;
    }
```

Collections

(all information gathered from: https://docs.oracle.com/javase/tutorial/collections/interfaces/index.html unless otherwise noted)

3. Lists are ordered collections with each element having a corresponding index. They can have duplicate elements, and the user has control over where each element is in the list. Sets, on the other hand, are un-ordered and cannot contain duplicate elements. An example of a set would be paintings by a physical artist. Van Gogh may have painted a lot of sunflowers, but they are each unique and separate. A map is a collection that maps (ha) keys to values. Keys in a map are unique, but can contain duplicate values. All collections are dynamic, though, meaning you can add new elements after instantiating.

4. Two examples of lists are:

   i) a list to keep track of the order in which kids sign up for a tutoring session

   ii) a list to keep track of all the models of cars you own

   Two examples of sets are:

   i) You are keeping track of all the pets available for adoption at the Humane Society
   ii) You are keeping track of all the songs by The Beatles you can play on the guitar

   Three examples of maps are:

   i) A dictionary
   ii) Logins for a website
   iii) Company hires and their hire-dates

5. a) ArrayList<String> sampleList = new ArrayList<String>();

   b) HashSet<String> sampleSet = new HashSet<String>();

   StringBuilder sampleSB = new StringBuilder();

   (I really didn't understand the question of instantiating a HashSet of StringBuilder)

   c) HashMap<String, String> loginCredentials = new HashMap<String, String>();