Java Strings

CS 1400 2024
Cody Henrichsen

1

String Details

- String is a fundamental part of Java
- The Java String is NOT a structure, nor iterable
- All the UNICODE values including emoji fit in String
- Strings are **IMMUTABLE**
  - They can only be changed by assigning a replacement value into the same variable!!
- Everything can be turned into a String!
  - All Objects have a `toString()` method
- Check out the Java String API for even more details!

2

String UML

3

AP CED subset

- We do not need to know EVERYTHING about a String to prepare for the AP Exam. As such we have a minimal subset of methods to cover for the exam

java.lang.String (CED Subset)

```
+ String() : Constructor
+ String(String) : Constructor

+ length() : int
+ substring(int, int) : String
+ substring(int) : String
+ indexOf(String) : int
+ compareTo(String) : boolean
+ split(String) : String []
+ equals(Object) : boolean
```

4

String Constructors

- Zero Parameter Constructor
  - Creates an empty String AKA ""
    - `String myText = new String();`
- String parameter
  - Creates a new String using the supplied parameter
    - `String myText = new String("");`
    - `String myText = new String(someStringVariable);`
    - `String myText = new String(myBook.getFavoriteWord());`

5

String Literal

- Creating a `String` without a constructor
- Any text surrounded by double quotes ""
- Simply assign the text to a variable
  - `String myName = "Cody Henrichsen";`
  - `String special = "Even emoji work 🤡";`

6

Escape Sequences

- How to handle special situations in Java Strings
- Inside the ""!
- The backslash symbol (\) is combined with certain other symbols to display specific things
  - Quotation marks
    - `\"`
  - New lines
    - `\n`
  - Backslashes
    - `\\`
  - Tabs
    - `\t`

7

How big is it?

- Since a Java String is immutable, the Java language specification uses the `length()` method when referring to how big a `String` is.
- Each symbol in the `String` counts, including spaces!
  - `String title = "Doctor Who";`
  - `int lengthOfTitle = title.length();`
    `//Contains 10`

8

Parts of the String

- Again, `String` is a type not a structure. Each part of a `String` is also a `String`
- When you only want part of a `String` then `substring` is the way to go. Remember that in Java, the index always starts at 0. Of course, all values need to be in the range of `0 <= length()`
  - substring(int, int)
  - substring(int)

9

Inside – substring(int, int)

- This is the more refined method of getting part of a `String`. The first parameter is the inclusive starting index and the second parameter is the **exclusive** ending index.
- This is THE way to iterate over each letter in a `String`
  - `String currentLetter = myString.substring(index, index + 1);`

10

Inside – substring(int)

- This method takes from the supplied index to the end of the `String`.
- This is great when you only want the last part of a `String`, say maybe a file extension or the conjugation of a regular verb
- `myString.substring(2)` is the same as `myString.substring(2, myString.length())`

11

Basic Location : indexOf(String)

- When you need to know the location of the supplied `String` parameter in the calling `String`
- Returns a 0-based index for the starting location if found within the calling `String` for the FIRST occurrence
- If the `String` parameter is NOT present within the calling `String`, the method will return -1
  - Includes if the parameter is too big to fit in the calling `String`

12

Comparing Strings – Equality

- This is probably one of the most helpful methods of the AP subset on `String`. It allows you to compare `String` values as an object. This is often used for `if` blocks both in the test as well as example programs.
- One of the ways to remember to use `.equals(String)` instead of `==` for a String is that `.equalsIgnoreCase(String)` also exists since `==` is **NOT** the best way to check for `String` equality in Java
  - `"Me".equals("me"); //false`
  - `"Me".equals("Me"); //true`
  - `"Me".equalsIgnoreCase("me"); //true`
  - `"me" == "me"; //false`
    - Gee, thanks `StringPool`

13

Comparing Strings - Sorting

- One of the objectives in almost every introductory computer science course involves the sorting of String data. Java provides for this with the `compareTo(String)` method. Depending on what the lexicographic (Dictionary order) relationship between the calling `String` and the parameter is starting at the **FIRST** substring, you will receive one of three values.
  - An integer less than 0 if the calling String would be found before the parameter
    - `"a".compareTo("b");`
    - `"car".compareTo("cat");`
    - `"Some text".compareTo("other text"); //Based on the first letter`
    - `"A".compareTo("a"); // Case matters!!!`
  - The integer 0 if the calling String is the same as the parameter String
    - `"a".compareTo("a");`
  - An integer greater than 0 if the calling String would be found after the parameter
    - `"b".compareTo("a");`
    - `"cat".compareTo("car");`

14

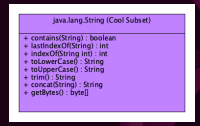
Extracting Strings

- OMG this is my FAVORITE method
- The `split` method is used to extract a String into its component Strings
- It returns an array of `String` pieces using the parameter to separate them
  - An empty `String` parameter will return an array of each symbol as a `String`
  - A space as a parameter will return an array of each word in the String as an array of `String` values
  - A comma as the parameter will return an array of each value between the commas as an array of `String` values (CSV file!)

15

Beyond AP

- There are many methods that are helpful beyond the AP curriculum. Use the Java API to find out even more!!



16

Look deeper inside

- When working with data it is often necessary to look deeper inside the String in question. These methods provide more functionality than just the first occurrence of a parameter than the `indexOf(String)` method supplies
  - contains(String)
    - Is the parameter located within the calling String. This one chains well with `toLowerCase()`
  - lastIndexOf(String)
    - What is the last index of the supplied String. This one is especially useful for locating the extension of a filename because of the period may be in a username
  - indexOf(String, int)
    - The overloaded `indexOf` is Fantastic! This version is great for looking ahead in a String for repeat occurrences

17

Change how it looks

- String values are **immutable**, so the variable only changes if you use an assignment operator, but you can use these methods to change a String provided you assign the result to a variable or simply use the temporary result.
  - toLowerCase() AND toUpperCase()
    - These two are pretty self explanatory. Just remember that String values including emoji: (😊, 🤔, 🎮), and non-Roman characters (わたし) do not have a "case" and will not be affected by these
  - trim()
    - Did you ever want to get rid of all the extra spaces before or after a String caused by resting your head/cat on the space bar? Then trim() is what you need to call
  - concat(String)
    - You want to build a String from other Strings and don't want to use an overloaded + operator? Then concat(String) is your friend.

18

Data

- When you want to treat a String as a regular chunk of electronic data AKA a bunch of bytes, then the `getBytes()` method is your friend. I find it most useful when writing to a file using a `FileOutputStream`
  - `getBytes()`
    - Returns the `byte []` represented by the calling `String`

19