Computer Language

String APIs

Agenda

- String APIs

String

String class

- java.lang.String
- String representation

```
// generation of a string object using string literals
String str1 = "abcd";

// generation of a string object using String class
char data[] = {'a', 'b', 'c', 'd'};
String str2 = new String(data);
String str3 = new String("abcd");
```

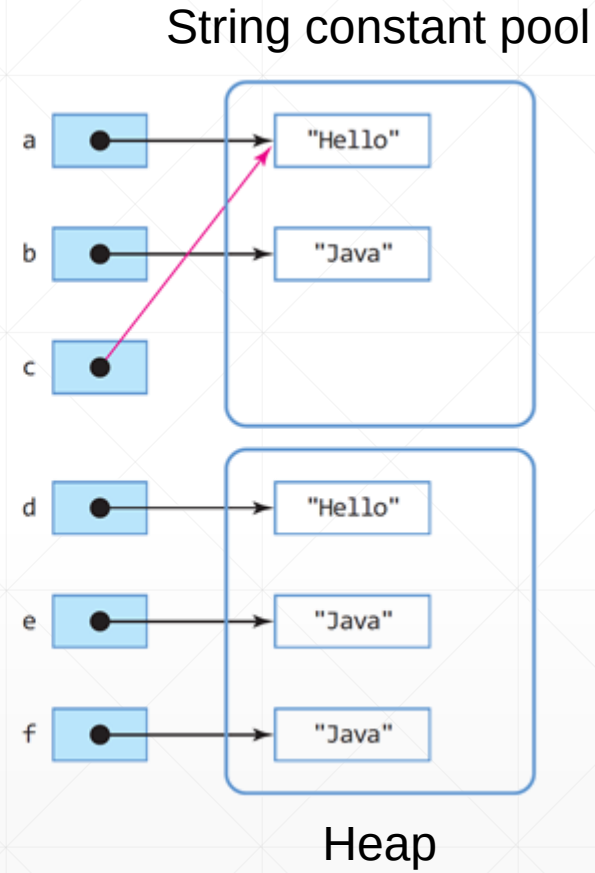
String constructor

constructor	description
`String()`	Create an empty string object
`String(char[] value)`	Create a String object with text values in char[]
`String(String original)`	Create a String object with the given String value
`String(StringBuffer buffer)`	Create a String object with text values in StringBuffer

String: Literal vs new String()

- How to create a String?
  - Literal (String str = “Hello”;)
    - String by literal is sharable (String constant pool)
    - Managed by JVM

```
String a = "Hello";
String b = "Java";
String c = "Hello";
String d = new String("Hello");
String e = new String("Java");
String f = new String("Java");
```

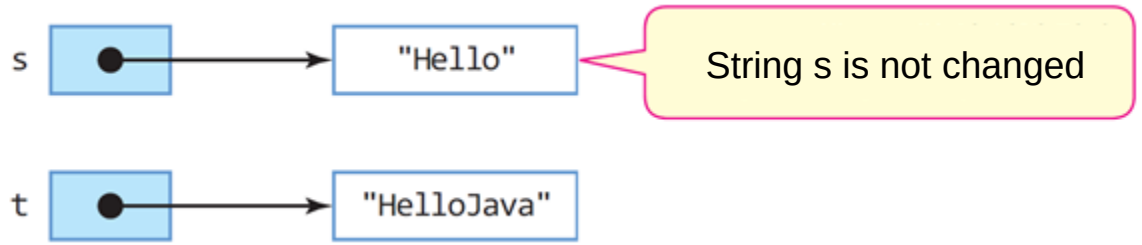


  - String object (String str = new String(“Hello”);)
    - A new object is created in the Heap memory area

String: Characteristics

- Immutable
  - String object CANNOT be modified

```
String s = new String("Hello");
String t = s.concat("Java"); // Concatenation of two strings. Returns a NEW string
```



- Comparison
  - String should be compared using equals() method
  - == operator compares two objects based on their address

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html#method.summary

String: Methods

Method	Description
`char charAt(int index)`	Returns the char value at the specified index.
`int codePointAt(int index)`	Returns the character (Unicode code point) at the specified index.
`int compareTo(String anotherString)`	Compares two strings lexicographically.
`String concat(String str)`	Concatenates the specified string to the end of this string.
`boolean contains(CharSequence s)`	Returns true if and only if this string contains the specified sequence of char values.
`int length()`	Returns the length of this string.
`String replace(Charsequence target, Charsequence replacement)`	Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
`String[] split(String regex)`	Splits this string around matches of the given regular expression.
`String subString(int beginIndex)`	Returns a string that is a substring of this string.
`String toLowerCase()`	Converts all of the characters in this String to lower case using the rules of the default locale.
`String toUpperCase()`	Converts all of the characters in this String to upper case using the rules of the default locale.
`String trim()`	Returns a string whose value is this string, with all leading and trailing space removed.

String: Comparison

- int compareTo(String anotherString)
  - Returns:
    - 0, if the argument string is equal to this string
    - A value less than 0 if this string is lexicographically less than the string argument
    - A value greater than 0 if this string is lexicographically greater than the string argument

```
String java= "Java";
String cpp = "C++";
int res = java.compareTo(cpp);
if(res == 0)
   System.out.println("the same");
else if(res <0)
   System.out.println(java + " < " + cpp);
else
   System.out.println(java + " > " + cpp);
```

"Java" is lexicographically greater than "C++"

```
Java > C++
```

String: Concatenation

Concatenation using + operator

- In case where operands include a string or an object
  - Object is converted to String using toString() method and then concatenated
  - Primitive types are converted to String and then concatenated

```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh
String a = "Java";
String b = "Final Exam ";
System.out.println(a+b+"score:"+100);
// Java Final Exam score:100
```

Concatenation using concat(String str)

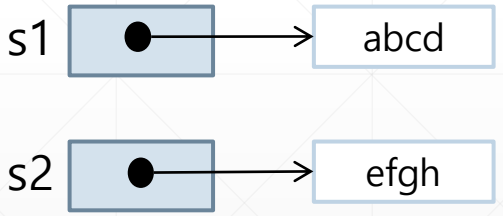
- Returns a new string where the string values are concatenated

```
System.out.println("I like ".concat("Java"));
String str = "But, I love ";
System.out.println(str.concat("Python!"));
```

String: Concatenation (cont'd)

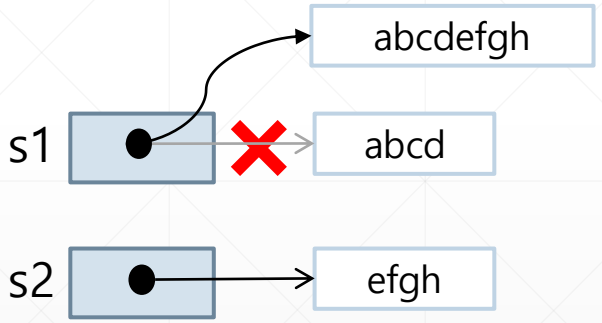
- Concatenation using concat(String str)
  - Returns a new string where the string values are concatenated

```
String s1 = "abcd";
String s2 = "efgh";
```




```
s1 = s1.concat(s2);
```

A new string object returned by s1.concat(s2)



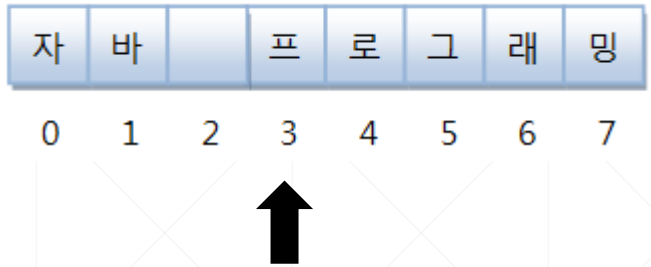
String: Accessing Values

- Accessing a character in the string
  - Char charAt(int index)
    - Returns the char value at the specified index
    - An index ranges from 0 to length() - 1

```
String a = "class";
char c = a.charAt(2); // c = 'a'
```

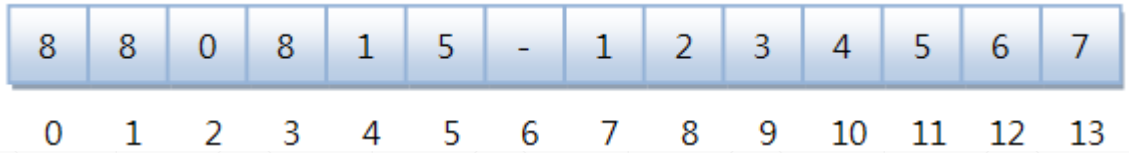
```
// to count the number of 's' in "class"
int count = 0;
String a = "class";
for(int i=0; i<a.length(); i++) { // a.length() = 5
   if(a.charAt(i) == 's')
      count++;
}
System.out.println(count); // 2
```

```
String subject = "자바 프로그래밍";
char charValue = subject.charAt(3);
```



String: Accessing Values (cont'd)

- Extracting a sub-string from the string
  - String substring(int beginIndex)
    - Returns a substring that begins with the character at the specified index and extends to the end of this string
  - String substring(int beginIndex, int endIndex)
    - Returns a substring that begins at the specified beginIndex and extends to the character at index endIndex - 1




```
String ssn  = "880815-1234567";
String firstNum  = ssn.substring(0,  6);
String secondNum  = ssn.substring(7);
```

String: Replace

- Replaces the original text with the new text
  - String replace(CharSequence target, CharSequence replacement)
    - Replaces each substring of this string that matches the *target* sequence with the specified *replacement* sequence
    - The replacement proceeds from the beginning of the string to the end
      - E.g., replacing "aa" with "b" in the string “aaa" will result in "ba" rather than "ab"
  - Example)

```
String java = "Java is the best";
System.out.println(java.replace("best","worst"));
String comLang = java.replace("Java","Nothing");
System.out.println(comLang);
```

String: Split

- Splits this string around matches of the given regular expression
  - String[] split(String regex)
  - Example)
    - String “boo:and:foo”

Regex	Result
:	{ "boo", "and", "foo" }
o	{ "b", "", ":and:f" }

```
String lang ="Java,C++,C#,Python,Javascript";
String[] langs = lang.split(",");
System.out.println(langs.length);
for(String l : langs) System.out.println(l);
```

More about regular expression
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html#sum

String: Trim & Lower/Upper Case

- Removes all leading and trailing spaces (tab, enter, space)
  - String trim()

```
String a = "       abcd def       ";
String b = "       xyz₩t";
String c = a.trim(); // c = "abcd def". Space inside the string is not removed
String d = b.trim(); // d = "xyz". Spaces and '₩t' removed
```

- Case conversion
  - String toLowerCase()/toUpperCase()

```
String original = "Java Programming";
String lowerCase = original.toLowerCase();
String upperCase = original.toUpperCase();
```

String: Example

- Removes all leading and trailing spaces (tab, enter, space)

```
public class StringEx {
    public static void main(String[] args) {
        String a = new String(" C#");
        String b = new String(",C++ ");

        System.out.println(a + "'s length is" + a.length()); //
        System.out.println(a.contains("#")); // if String a contains '#'?

        a = a.concat(b); // string concatenation
        System.out.println(a);

        a = a.trim(); // string trimming
        System.out.println(a);

        a = a.replace("C#","Java"); // string replacement
        System.out.println(a);

        String s[] = a.split(","); // string split
        for (int i=0; i<s.length; i++)
            System.out.println("split string is" + i + ": " + s[i]);

        a = a.substring(5); // accessing a substring
        System.out.println(a);

        char c = a.charAt(2); // accessing a character
        System.out.println(c);
    }
}
```

StringTokenizer

- Alternative class to split a string value into tokens using delimiters
  - Delimiter: character to separate tokens
  - Token: split substring

- Constructor

Constructor	Description
`StringTokenizer(String str)`	Constructs a string tokenizer for the specified string. The tokenizer uses the default delimiter set, which is " \t\n\r\f"
`StringTokenizer(String str, String delim)`	Constructs a string tokenizer for the specified string. The characters in the delim argument are the delimiters for separating tokens.
`StringTokenizer(String str, String delim, boolean returnDelims)`	Constructs a string tokenizer for the specified string. All characters in the delim argument are the delimiters for separating tokens. If the returnDelims flag is true, then the delimiter characters are also returned as tokens.

StringTokenizer (cont'd)

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/StringTokenizer.html#method.summary

- Methods

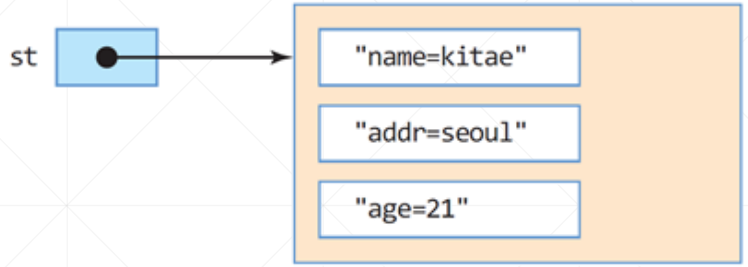
Method	Description
`int countTokens()`	Calculates the number of times that this tokenizer's nextToken() method can be called
`boolean hasMoreTokens()`	Tests if there are more tokens available from this tokenizer's string
`String nexToken()`	Returns the next token from this string tokenizer

StringTokenizer (cont'd)

- Example)

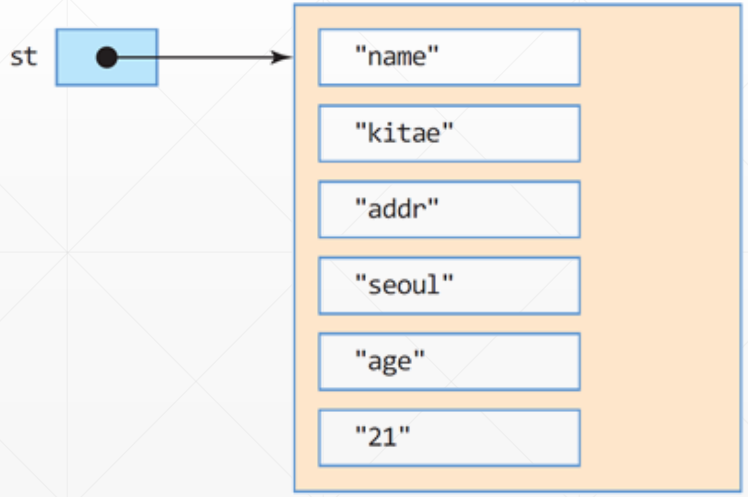
```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

Delimiter: '&'




```
StringTokenizer st = new StringTokenizer(query, "&=");
```

Delimiter: '&', '='



StringTokenizer (cont'd)

- Example)

```
import java.util.StringTokenizer;


public class StringEx {
    public static void main(String[] args) {
        String lang ="Java,C++,C#,Python,Javascript";
        StringTokenizer st = new StringTokenizer(lang,",");
        System.out.println("We have "+st.countTokens()+" tokens!");
        while(st.hasMoreTokens()) System.out.println(st.nextToken());
    }
}
```

StringBuffer

- Mutable sequence of characters (String)
  - String values can be modified

- Constructor

Constructor	Description
`StringBuffer()`	Constructs a string buffer with no characters in it and an initial capacity of 16 characters
`StringBuffer(charSequence seq)`	Constructs a string buffer that contains the same characters as the specified CharSequence
`StringBuffer(int capacity)`	Constructs a string buffer with no characters in it and the specified initial capacity
`StringBuffer(String str)`	Constructs a string buffer initialized to the contents of the specified string

StringBuffer (cont'd)

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/StringBuffer.html#method.summary

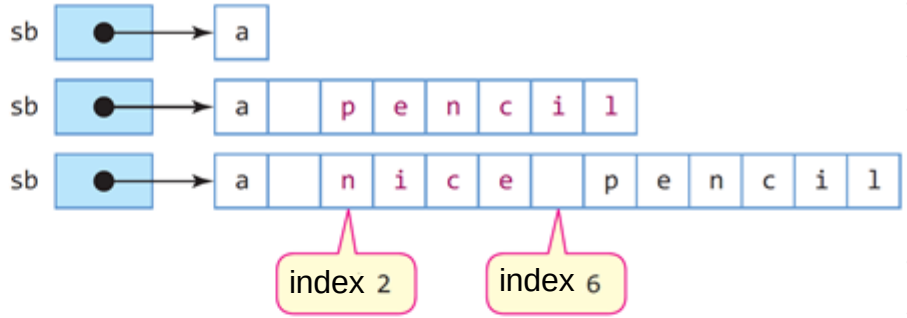
- Methods

Method	Description
`StringBuffer append(String str)`	Appends the specified string to this character sequence
`StringBuffer append(StringBuffer sb)`	Appends the specified StringBuffer to this sequence
`int capacity()`	Returns the current capacity
`StringBuffer delete(int start, int end)`	Removes the characters in a substring of this sequence
`StringBuffer insert(int offset, String str)`	Inserts the string into this character sequence
`StringBuffer replace(int start, int end, String str)`	Replaces the characters in a substring of this sequence with characters in the specified String
`StringBuffer reverse()`	Causes this character sequence to be replaced by the reverse of the sequence
`void setLength(int newLength)`	Sets the length of the character sequence

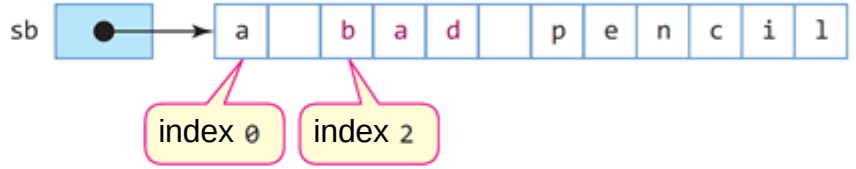
StringBuffer (cont'd)

- Example)

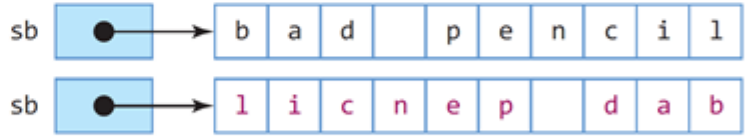
```
StringBuffer sb = new StringBuffer("a");
sb.append(" pencil");
sb.insert(2, "nice ");
```



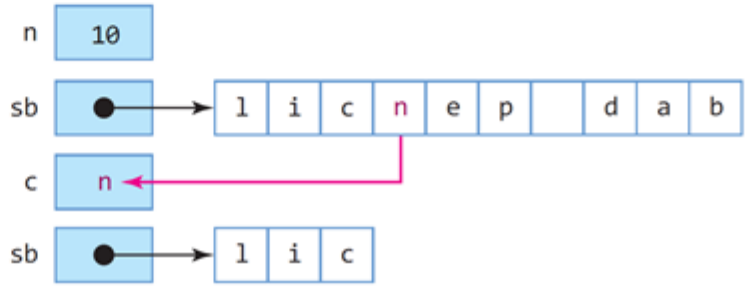

```
sb.replace(2, 6, "bad");
```




```
sb.delete(0, 2);
sb.reverse();
```




```
int n = sb.length();
char c = sb.charAt(3);
```




```
sb.setLength(3);
```

StringBuffer (cont'd)

- Example)

```
public class StringBufferEx {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("This");

      sb.append(" is pencil"); // appending a string
      System.out.println(sb);

      sb.insert(7, " my"); // inserting "my"
      System.out.println(sb);

      sb.replace(8, 10, "your"); // replacing "my" with "your"
      System.out.println(sb);

      sb.delete(8, 13); // deleting "your "
      System.out.println(sb);

      sb.setLength(4); // setting a new length
      System.out.println(sb);
   }
}
```

More APIs

- Math
- Calendar
- Date
- …