Topics

- Important String Methods

Getting length of Strings

- int length() → Returns the length of this String reference
- Example

1. “abc”.length() returns 3
2. String s1 =“Object”; s1.length() returns 6

String Concatenation Using '+' operator

- + operator can be used for string concatenation
- + operator can concatenate not only two strings but also a string type value with any other type
- Suppose 's' is String reference and 't' is any type other than string. Then an expression 's + t' will convert t to string type and final result will be of String type
- Examples

1. "abc" + 10 → results in "abc10"
2. 10 + "abc" → results in "10abc"
3. "abc" + "def" → results in "abcdef"
4. 4.56 + "Object" → results in "4.56Object"
5. 10 + 20 + "Java" → results in 30Java
6. "Java"+10+20 → Java1020

String Concatenation Using 'concat()' Method

- String concat(String s) → Appends 's' with this String and returns a new String with updated contents
- However, preferred way for string concatenation is via '+' operator
- Examples

1. String s1 = "Object";
   String s2 = s1.concat("-Oriented Programming);
2. System.out.println("Java ".concat(" World"));

Character Extraction : chartAt()

- char chartAt(int index) → Returns the character at index. The value of index should be 0<=index<=L-1, where 'L' is length of string.
- Examples:

1. "abc".charAt(0) → returns 'a'
2. "abc".charAt(3) → results in StringIndexOutofBoundsException
3. String s1 = "Object-Oriented Programming"; s1.charAt(10) → returns 'e'

Character Extraction : getChars()

- void getChars(int sourceStart, int sourceEnd, char target[], int targetStart)
- Charcaters are extracted from invoking (this) string reference
- sourceStart → start index of invoking (this) string reference from where character extraction will start (0 <= sourceStart <= L-1, where L is length of invoking string)
- sourceEnd→ end index (exclusive) of invoking (this) string reference. (0 <= sourceEnd <= L-1, where L is length of invoking string and also sourceStart<=sourceEnd)
- Characters are extracted from sourceStart to sourceEnd-1
- target represents the target character array where extracted characters will be stored
- targetStart → start index of target from where characters will be stored. Note: target array should be large enough to store the extracted characters.

Character Extraction : getChars() : Example 1


```
//File Name: StringDemo.java
class StringDemo
{
        public static void main(String[] args)
        {
                String  s       = "When Java converts data";
                char[]  chars   = new char[100];
                s.getChars(10,30,chars,0);
                System.out.println(chars);
        }// End of Method
}// End of class StringDemo
```

Start index =10
End index = 29

```
F:\>java StringDemo
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String ind
ex out of range: 30
        at java.lang.String.getChars(Unknown Source)
        at StringDemo.main(xyz.java:8)
```

Character Extraction : getChars() : Example 2


```
//File Name: StringDemo.java
class StringDemo
{
        public static void main(String[] args)
        {
                String  s       = "When Java converts data";
                char[]  chars   = new char[100];
                s.getChars(10, 5 ,chars,0);
                System.out.println(chars);
        }// End of Method
}// End of class StringDemo
```

endIndex < startIndex

```
F:\>java StringDemo
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String ind
ex out of range: -5
        at java.lang.String.getChars(Unknown Source)
        at StringDemo.main(xyz.java:8)
```

Character Extraction : getChars() : Example 3


```
//File Name: StringDemo.java
class StringDemo
{
        public static void main(String[] args)
        {
                String s = "When Java converts data";
                char[] chars = new char[20];
                s.getChars(4, 14 ,chars,14);
                System.out.println(chars);
        }// End of Method
}// End of class StringDemo
```

Characters will be extracted from index 4 to 13 (10 characters)

To store 10 chacters from index 14 the last index should be 23
However the last index of chars is 19

```
F:\>java StringDemo
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
        at java.lang.System.arraycopy(Native Method)
        at java.lang.String.getChars(Unknown Source)
        at StringDemo.main(xyz.java:8)
```

Character Extraction : getChars() : Example 4


```
//File Name: StringDemo.java
class StringDemo
{
        public static void main(String[] args)
        {
                String  s       = "When Java converts data";
                char[] chars = new char[20];
                s.getChars(4, 14 ,chars,4);
                System.out.println(chars);
        }// End of Method
}// End of class StringDemo
```

Characters will be extracted from index 4 to 13 (10 characters)

<<OUTPUT>>

```
F:\>java StringDemo
Java conv
```

byte[] getBytes() : Method

- byte[] getBytes() → Converts each character of string to byte and returns a byte arrays
- Example

1.
```
byte[]      values      =      "Object".getBytes();
for(int i =0; i < values.length; i++)
        System.out.print(values[i] + " ");
```
79 98 106 101 99 116

2.
```
byte[]      bytes    =      new byte[10];
String      s1       =      "Java";
bytes                =      s1.getBytes();
for(int i =0; i < bytes.length; i++)
        System.out.print(bytes[i] + " ");
```
74 97 118 97

char[ ] toCharArray()

- char[ ] toCharArray() → converts a string into a char type array and returns char[]

```
//File Name: StringDemo.java
class StringDemo
{
        public static void main(String[] args)
        {
                System.out.println("Thomas Alva Edison".length());
                char[] name = "Thomas Alva Edison".toCharArray();
                System.out.println(name.length);
                System.out.println(name);
        }// End of Method
}// End of class StringDemo
```

```
F:\>java StringDemo
18
18
Thomas Alva Edison
```

String Comparisons: equals() and equalsIgnoreCase()

- boolean equals(String other) → case-sensitive comparison [upper and lower-case letters are not equal]
- Example : “abc”.equals(“Abc”) returns false, “abc”.equals(“abc”) returns true
- boolean equalsIgnoreCase(String other) → Comparison by ignoring the case of characters
- Example : “abc”.equalsIgnoreCase (“ABC”) or “aBc”. equalsIgnoreCase (“abC”) returns true

String Comparisons: regionMatches()

- regionMatches() → compares the regions of two string references for equality [returns true if regions matches otherwise false]
- Syntax:

1. ***boolean regionMatches(int startIndex, String str2, int str2startIndex, int numChars) → case-sensitive comparison***

start index of invoking string

Second String for comparison

start index of second String

Number of characters to be compared [str2startIndex to str2startIndex + numChars – 1]

2. ***boolean regionMatches(boolean ignoreCase, int startIndex, String str2, int str2startIndex, int numChars) → Can ignore case if the first parameter value is true.***

- **Note : Index parameters should be within range otherwise StringIndexOutofBoundsException will be thrown**

String Comparisons: regionMatches() : Example


```
String s1 = "India finally has started preparations for Rio Olympics";

String s2 = "David has started preparations for Final Exams";

String s3 = "David has Started PREPARATIONS FOR Final Exams";

System.out.println(s1.regionMatches(14,s2,6,24));          // Case-Sensitive
System.out.println(s1.regionMatches(14,s3,6,24));          // Case-Sensitive
System.out.println(s1.regionMatches(true,14,s3,6,24));     // Ignore-case
System.out.println(s1.regionMatches(false,14,s3,6,24));    // Case-Sensitive
```

true

false

true

false

String Comparisons: startsWith() and endsWith()

- boolean startsWith(String str) → Returns true if this string starts with 'str' otherwise false. For Example, "Object"startsWith("Obj") returns true.
- boolean startsWith(String str, int startIndex) → Overloaded Flavor. Returns true if this string starts with 'str' from index startIndex otherwise false. For Example, "Object"startsWith("ect",3) returns true.
- boolean endsWith(String str) → Returns true if this string ends with 'str' otherwise false. For Example, "FooBar".endsWith("Bar"); returns true

String Comparisons: compareTo()

- int compareTo(String str) → Case-Sensitive Comparison. returns a +ive value if this string is > str , -ive value if this string is < str, otherwise a 0 (zero) value. Used For sorting an arrays of String values.
- Comparison is done character by character. If all characters of both strings matches then 0 is returned otherwise a difference of the first non-matched character is returned
- Examples
  1. System.out.println("Ram".compareTo("RAM")); **// Prints 32**
  2. System.out.println("Java".compareTo("Object")); **// Prints -5**
  3. System.out.println("Hello".compareTo("Welcome")); **// Prints -15**
- int compareToIgnoreCase(String str) → Ignore case comparison.
- Examples
  1. System.out.println("Ram".compareToIgnoreCase("RAM")); **// Prints 0**
  2. System.out.println(“oBjeCt”.compareTo(“OBJECT")); **// Prints 0**

Thank You