Java

Strings

Department of Data Science
& Engineering, DCA, MIT

String related classes

- Java provides four String related classes
- java.lang package
  - ***String*** class: Storing and processing Strings but Strings created using the String class cannot be modified (**immutable**)
  - ***StringBuffer/StringBuilder*** class: Create flexible Strings that can be modified
- java.util package
  - ***StringTokenizer*** class: Can be used to extract tokens from a String

String

String

- String class provide many constructors and more than 40 methods for examining in individual characters in a sequence
- You can create a String from a String value or from an array of characters.
  - *String newString = new String(stringValue);*
- The argument stringValue is a sequence of characters enclosed inside double quotes
  - String message = new String (“Welcome”);
  - String message = “Welcome”;

String Constructors

```java
public class StringConstructorTest {
    public static void main(String[] args) {
        char charArray[ ] = { 'b',  'i',  'r',  't',  'h',  ' ',  'd',  'a',  'y'} ;
        byte byteArray[ ] = { ( byte)  'n',  ( byte)  'e',  ( byte)  'w', ( byte)  ' ',
                ( byte)  'y',  ( byte)  'e',  ( byte)  'a',  ( byte)  'r'} ;

        String s = new String("hello"); // hello
        String s1 = new String() ; //
        String s2 = new String(s) ; // hello
        String s3 = new String(charArray); // birth day
        String s4 = new String(charArray, 6, 3); // day
        String s5 = new String(byteArray, 4, 4); // year
        String s6 = new String(byteArray); // new year
        String s7 = "Wel"  +  "come"; // Welcome

        System.out.println(s);
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
        System.out.println(s4);
        System.out.println(s5);
        System.out.println(s6);
        System.out.println(s7);
    }
}
```

String Length

- Returns the length of a String
  - ***length()***
- Example:

```
String s1="Hello";
System.out.println(s1.length());
```

Extraction

- Get the character at a specific location in a string
  - ***s1.charAt(1)***
- Get the entire set of characters in a string
  - ***s1.getChars(0, 5, charArray, 0)***

The ***getChars()*** method is used to copy characters from a given string into the destination character array

```
getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
```

Parameters:

Name	Description	Type
srcBegin	index after the last character in the string to copy.	int
srcEnd	index of the first character in the string to copy.	int
dst	the destination array.	char
dstBegin	the start offset in the destination array.	int

Extraction

```
1  class getCharsDemo {
2  public static void main(String args[]) {
3          String s = "This is a demo of the getChars method.";
4          int start = 10;
5          int end = 14;
6          char buf[] = new char[end - start];
7          s.getChars(start, end, buf, 0);
8          System.out.println(buf);
9      }
10 }
```

Output:

demo

Extracting Substrings

- substring method enable a new String object to be created by copying part of an existing String object
  - ***substring (int startIndex)*** - copies the characters form the starting index to the end of the String
  - ***substring(int beginIndex, int endIndex)*** - copies the characters from the starting index to one beyond the endIndex

String Comparisons

- ***equals***
  - Compare any two string objects for equality using lexicographical comparison.
  - ***s1.equals("hello")***
- ***equalsIgnoreCase***
  - ***s1.equalsIgnoreCase(s2)***
- ***compareTo***
  - ***s1.compareTo(s2)***
  - s1 > s2 (positive), s1 < s2 (negative), s1 = s2 (zero)

String Comparisons

```java
// Demonstrate equals() and equalsIgnoreCase().
class equalsDemo {
public static void main(String args[]) {
        String s1 = "Hello";
        String s2 = "Hello";
        String s3 = "Good-bye";
        String s4 = "HELLO";
        System.out.println(s1 + " equals " + s2 + " -> " +
        s1.equals(s2));
        System.out.println(s1 + " equals " + s3 + " -> " +
        s1.equals(s3));
        System.out.println(s1 + " equals " + s4 + " -> " +
        s1.equals(s4));
        System.out.println(s1 + " equalsIgnoreCase " + s4 + " -> " +
        s1.equalsIgnoreCase(s4));
}
}
```

```
Hello equals Hello -> true
Hello equals Good-bye -> false
Hello equals HELLO -> false
Hello equalsIgnoreCase HELLO -> true
```

String Comparisons

```java
public class StringEqualsTest {
    public static void main(String[] args) {
        String s1 = "Hello";
        String s2 = new String( original: "Hello");
        String s3 = "Hello";
        System.out.println("s1 == Hello " + s1.equals("Hello")); // true
        System.out.println("s1 == s2 " + s1.equals(s2)); // true
        System.out.println("s1 == s3 " + s1.equals(s3)); // true
        System.out.println("s2 == s3 " + s2.equals(s3)); // true
        System.out.println(s1 == s2); // false
        System.out.println(s1 == s3); // true
        System.out.println(s2 == s3); // false
    }
}
```

Note: The **"equals()"** method compares the characters within the strings for equality.

The **"=="** operator compares two object references to see whether they refer to the same instance.

String Comparisons

- *regionMatches* compares portions of two String objects for equality

```
boolean regionMatches(int startIndex, String str2,
                      int str2StartIndex, int numChars)
```

  - ***s1.regionMatches (0, s2, 0, 5)***

```
boolean regionMatches(boolean ignoreCase,
                      int startIndex, String str2,
                      int str2StartIndex, int numChars)
```

  - ***s1.regionMatches (true, 0, s2, 0, 5)***
- If the first argument is true, the method ignores the case of the characters being compared
- *startsWith* and endsWith check whether a String starts or ends with a specified String
  - ***s1.startsWith (s2)***
  - ***s1.endsWith (s2)***

String Methods

- String methods (Character Extraction):

char	charAt (int index)
void	getChars (int sourceStart, int sourceEnd, char target [], int targetStart)
byte [ ]	getBytes () //stores characters in an array of bytes
char [ ]	toCharArray ()

- String methods (Case conversion):

String	toLowerCase ()
String	toUpperCase ()

String Methods

```
3     public static void main(String
4     {
5       String strOb1 = "First Strin
6       String strOb2 = "Second Stri
7       String strOb3 = strOb1;
8
9       System.out.println("Length of strOb1: " +
10                         strOb1.length());
11
12      System.out.println("Char at index 3 in strOb1: " +
13                         strOb1.charAt(3));
14
15      if(strOb1.equals(strOb2))
16        System.out.println("strOb1 == strOb2");
17      else
18        System.out.println("strOb1 != strOb2");
19
20      if(strOb1.equals(strOb3))
21        System.out.println("strOb1 == strOb3");
22      else
23        System.out.println("strOb1 != strOb3");
24    }
```

```
Length of strOb1: 12
Char at index 3 in strOb1: s
strOb1 != strOb2
strOb1 == strOb3
```

String Methods

```
1   class StringDemo3
2   {
3     public static void main(String args[])
4     {
5       String str[] = { "one", "two", "three" };
6
7       for(int i=0; i<str.length; i++)
8         System.out.println("str[" + i + "]: " +
9                                    str[i]);
10    }
11  }
```

```
str[0]: one
str[1]: two
str[2]: three
```

String Concatenation

- Java provide the *concat* method to concatenate two strings.

```
String s1 = new String ("Happy ");
String s2 = new String ("Birthday");
String s3 = s1.concat(s2);
```

s3 will be "Happy Birthday"

String Search

- To search for the first occurrence of a character, use

  int indexOf(char ch)

- To search for the last occurrence of a character, use

  int lastIndexOf(char ch)

- To search for the first or last occurrence of a substring, use

  int indexOf(String str)

  int lastIndexOf(String str)

You can specify a starting point for the search using these forms:

int indexOf(int *ch*, int *startIndex*)
int lastIndexOf(int *ch*, int *startIndex*)

int indexOf(String *str*, int *startIndex*)
int lastIndexOf(String *str*, int *startIndex*)

String Search

```java
// Demonstrate indexOf() and lastIndexOf().
class indexOfDemo {
    public static void main(String args[]) {
            String s = "Now is the time for all good men " +
            "to come to the aid of their country.";
            System.out.println(s);
            System.out.println("indexOf(t) = " +
            s.indexOf('t'));
            System.out.println("lastIndexOf(t) = " +
            s.lastIndexOf('t'));
            System.out.println("indexOf(the) = " +
            s.indexOf("the"));
            System.out.println("lastIndexOf(the) = " +
            s.lastIndexOf("the"));
            System.out.println("indexOf(t, 10) = " +
            s.indexOf('t', 10));
            System.out.println("lastIndexOf(t, 60) = " +
            s.lastIndexOf('t', 60));
            System.out.println("indexOf(the, 10) = " +
            s.indexOf("the", 10));
            System.out.println("lastIndexOf(the, 60) = " +
            s.lastIndexOf("the", 60));
        }
}
```

StringClass/indexOfDemo.java

String Search

```
Now is the time for all good men to come to the aid of their country.
indexOf(t) = 7
lastIndexOf(t) = 65
indexOf(the) = 7
lastIndexOf(the) = 55
indexOf(t, 10) = 11
lastIndexOf(t, 60) = 55
indexOf(the, 10) = 44
lastIndexOf(the, 60) = 55
```

String Split

- split() method splits a String against given regular expression and returns a character array
- ***String test = "abc,def,123";***

  String[] out = test.split(",");

  Output:

  out[0] - abc, out[1] - def, out[2] - 123

StringClass/StringSplit.java

String Conversions

- Generally, the contents of a String cannot be changed once the string is created.
- Java provides conversion methods

UpperLowerExample.java

- ***toUpperCase()*** and ***toLowerCase()***
  - Converts all the characters in the string to lowercase or uppercase

UseTrim.java

- ***trim()***
  - Eliminates blank characters from both ends of the string
- ***replace(oldChar, newChar)***

StringReplace.java

  - Replaces a character in the string with a new character

String Conversions

```
1   class StringTrimEg
2   {
3       public static void main(String[] args)
4       {
5           String s1 = "  Welcome to planet Earth  ";
6
7           System.out.println("Before trimming:"+s1);
8           System.out.println(s1.length());
9
10          String s2 = s1.trim();
11
12          System.out.println("After trimming:"+s2);
13          System.out.println(s2.length());
14      }
15  }
```

```
Before trimming:  Welcome to planet Earth
27
After trimming:Welcome to planet Earth
23
```

String to Other Conversions

- The String class provides ***valueOf*** methods for converting a character, an array of characters and numeric values to strings
  - ***valueOf*** method take different argument types

String to Other Conversions

Type	To String	From String
boolean	String.valueOf(boolean)	Boolean.parseBoolean(String)
byte	String.valueOf(int)	Byte.parseByte(String, int base)
short	String.valueOf(int)	Short.parseShort (String, int base)
Int	String.valueOf(int)	Integer.parseInt (String, int base)
long	String.valueOf(long)	Long.parseLong (String, int base)
float	String.valueOf(float)	Float.parseFloat(String)
double	String.valueOf(double)	Double.parseDouble(String)

StringValueOfExample.java

StringBuffer/StringBuilder

StringBuffer

- Can be used wherever a string is used
  - More flexible than String
  - Can add, insert, or append new contents into a string buffer
- The StringBuffer class has three constructors and more than 30 methods for managing the buffer and for modifying strings in the buffer
- Every StringBuffer is capable of storing a number of characters specified by its capacity

StringBuffer Constructors

StringBuffer ()
StringBuffer (int capacity)
StringBuffer (String str)
StringBuffer (CharSequence chars)

StringBuffer

```
1   class StringBufferDemo
2   {
3     public static void main(String args[])
4     {
5       StringBuffer sb = new StringBuffer("Hello..");
6
7       System.out.println("buffer = " + sb);
8       System.out.println("length = " + sb.length());
9
10      sb = sb.append("welcome..");
11      System.out.println("buffer = " + sb);
12      System.out.p
13
14      sb = sb.appe
15      System.out.p
16      System.out.p
17
18    }
19  }
```

```
buffer = Hello..
length = 7
buffer = Hello..welcome..
length = 16
buffer = Hello..welcome..manipal
length = 23
```

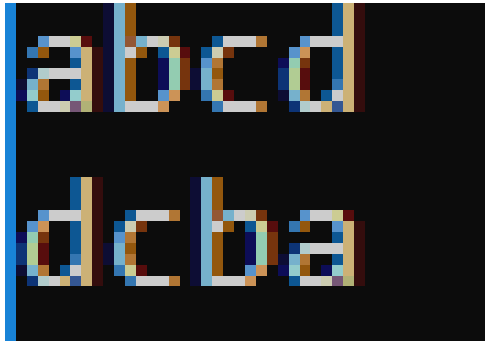
StringBuffer methods

int	capacity ()
int	length ()
char	charAt (int index)
int	indexOf (String str)
int	indexOf (String str, int fromIndex)
int	lastIndexOf (String str)
int	lastIndexOf (String str, int fromIndex)
StringBuffer	reverse()

StringBufferDemo.java

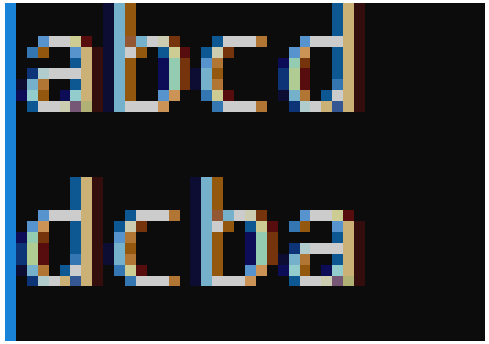
StringBuffer

```
1   // Using reverse() to reverse a StringBuffer.
2   class ReverseDemo
3   {
4     public static void main(String args[])
5     {
6       StringBuffer s = new StringBuffer("abcd");
7
8       System.out.println(s);
9       s.reverse();
10      System.out.println(s);
11    }
12  }
```



StringBuffer

```
1   // Using reverse() to reverse a StringBuffer.
2   class ReverseDemo
3   {
4     public static void main(String args[])
5     {
6       StringBuffer s = new StringBuffer("abcd");
7
8       System.out.println(s);
9       s.reverse();
10      System.out.println(s);
11    }
12  }
```



StringBuffer

```
1   // Demonstrate append().
2   class appendDemo
3   {
4     public static void main(String args[])
5     {
6       StringBuffer sb = new StringBuffer();
7
8       sb.append("Rough seas make good Sailors");
9
10      System.out.println(sb);
11    }
12  }
```

```
Rough seas make good Sailors
```

StringBuffer Methods

void	setCharAt (int index, char ch) // **to set the specified character at the given index**
void	setLength (int newLength) //**To set the length of the string within a StringBuffer object**
String	substring (int start)
String	substring (int start, int end)
void	trimToSize()
StringBuffer	delete (int start, int end)
StringBuffer	deleteCharAt (int index)

```java
// Demonstrate delete() and deleteCharAt()
class deleteDemo
{
  public static void main(String args[])
  {
    StringBuffer sb = new StringBuffer("This is a test.");

    sb.delete(4, 7);
    System.out.println("After delete: " + sb);

    sb.deleteCharAt(0);
    System.out.println("After deleteCharAt: " + sb);
  }
}
```

```
After delete: This a test.
After deleteCharAt: his a test.
```

```java
public class TrimToSizeExample
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer();
        sb.append("Testing");
        System.out.println("string: "+sb);
        int length = sb.length();
        int capacity = sb.capacity();
        System.out.println("length: "+length);
        System.out.println("capacity: "+capacity);
        sb.trimToSize();
        length = sb.length();
        capacity = sb.capacity();
        System.out.println("length after trimtosize: "+length);
        System.out.println("capacity after trimtosize: "+capacity);
    }
}
```

```
string: Testing
length: 7
capacity: 16
length after trimtosize: 7
capacity after trimtosize: 7
```

StringBuffer

- StringBuffer Methods:

StringBuffer	append (arg)
StringBuffer	append (char[] str, int offset, int len)
StringBuffer	insert (int offset, arg)
StringBuffer	insert (int index, char[] str, int offset, int len)
StringBuffer	replace (int start, int end, String str)

arg = boolean, char, char[], double, float, int, long, string, StringBuffer

```java
// Demonstrate insert().
class insertDemo
{
  public static void main(String args[])
  {
    StringBuffer sb = new StringBuffer("I Java!");

    sb.insert(2, "like ");
    System.out.println(sb);
  }
}
```

```
I like Java!
```

```java
// Demonstrate replace()
class replaceDemo
{
  public static void main(String args[])
  {
    StringBuffer sb = new StringBuffer("This is a test.");

    sb.replace(5, 7, "was");
    System.out.println("After replace: " + sb);
  }
}
```

```
After replace: This was a test.
```

```java
// Demonstrate replace()
class IndexOfDemo
{
  public static void main(String args[])
  {
    StringBuffer sb = new StringBuffer("one two one");
    int i;

    i = sb.indexOf("one");
    System.out.println("First index: " + i);

    i = sb.lastIndexOf("one");
    System.out.println("Last index: " + i);
  }
}
```

```
First index: 0
Last index: 8
```