String in Java

String related Classes

- In Java, a String is an Object
- Java provides three String related classes
  - *java.lang package*
    - **String** class -Storing and processing Strings but Strings created using the String class cannot be modified
    - **StringBuffer** 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

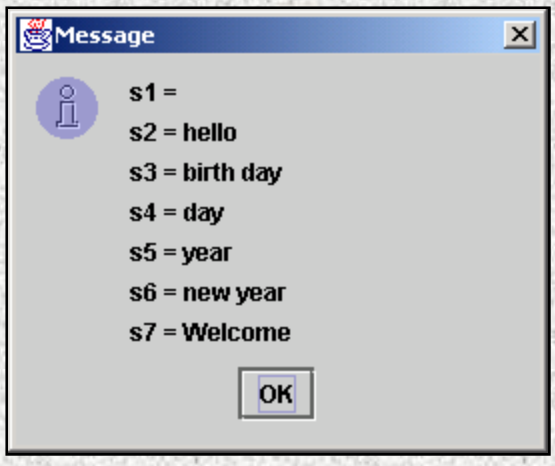
  String message = new String (“Welcome”);

  String message = “Welcome”;

String Constructors

```
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");
String s1 = new String( );
String s2 = new String(s);
String s3 = new String(charArray);
String s4 = new String(charArray, 6, 3);
String s5 = new String(byteArray, 4, 4);
String s6 = new String(byteArray);
String s7 = “Wel” + “come”;
```




```
String output = "s1 = " + s1 +  "\ns2 = " + s2 +  "\ns3 = " + s3 + "\ns4 = "
+ s4 + "\ns5 = " + s5 + "\ns6 = " + s6 +  "\ns7 = " +s7;
JOptionPane.showMessageDialog(null, output);
```

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)

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 number, s1 < s2 - negative number and s1 = s2 - zero

String Comparisons

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

  s1.regionMatches (0, s2, 0, 5);

  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 Concatenation

- Java provide the *concat* method to concatenate two strings.
- *String s1 = new String (“Happy ”);*

  String s2 = new String (“Birthday”);

  s1.concat(s2);

  s1 will be “Happy Birthday”

String Conversions

- Generally, the contents of a string cannot be changed once the string is created
- Java provides conversion methods
  - *toUpperCase()* and *toLowerCase()* -Return a new string by converting all the characters in the string to lowercase or uppercase
  - *Trim()* - Returns a new string by eliminating blank characters from both ends of the string
  - *Replace(oldChar, newChar)* - Can be used to replace a character in the string with a new character

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)

String Conversion Example

- To convert an int to a String:

  int n = 123;

  String s1 = Integer.toString(n);

  String s2=String.valueOf(n);

- To convert a string to an int:

  String s =“1234”;

  Int n = Integer.parseInt(s);

String Search

- Find the position of character/String within a String

  int indexOf(char ch);

  int lastIndexOf(char ch);

StringBuffer

StringBuffer

- Can be used wherever a string is used
  - More flexible than String
    - Can add, insert, or append new contents into a string buffer
    - However, the value of string is fixed once the string is created
- The String 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

- *public StringBuffer( )* - No characters in it and an initial capacity of 16 characters
- *public StringBuffer (int length)* -No characters in it and an initial capacity specified by the length argument
- *public StringBuffer (String string)* -Contains String argument and an initial capacity of the buffer is 16 plus the length of the argument

StringBuffer Methods

- *capacity()*- Returns the current capacity of the string buffer
- *length()* - Returns the number of characters in the string buffer
- *setLength()* -Sets the length of the string buffer
- *charAt ()* - Returns the character at a specific index in the string buffer . The first character of a string buffer is at index 0.

StringBuffer Methods

- You can append new contents at the end of a string buffer, insert new contents at a specified position in a string buffer, and delete or replace characters in a string buffer, Reverse a string buffer
  - Provides overload methods to append and insert boolean, char, char array, double, float, int, long and String into string buffer
  - *append, insert, delete, reverse, replace*

StringTokenizer

StringTokenizer

- Break a string into pieces (tokens) so that information contained in it can be retrieved and processed
  - How does the *StringTokenizer* class recognize individual words?
    - Specify a set of characters as delimiters when constructing a *StringTokenizer* object

StringTokenizer

- Constructors

  StringTokenizer(String str, String delim)

  StringTokenizer(String str)

- Methods

  hasMoreToken( ) - Returns true if there is a token left in the string

  nextToken( ) - Returns the next token in the string

  NextToken(String delim)- Returns the next token in the string after reseting the delimiter to *delim*

  countToken( ) - Returns the number of tokens remaining in the string tokenizer

Example

```
import java.util.*;
public class TestToken
{
   public static void main(String[] args)
  {
        String s = new String (“what’s your news");
        StringTokenizer tokens = new StringTokenizer(s);
        System.out.println(tokens.countTokens());
        while (tokens.hasMoreTokens())
        {
          System.out.println(tokens.nextToken() + "\n");
        }
   }
}
```