String Handling

The String Class

The String Class

- The Java String class, a part of the java.lang package, is used to represent strings of characters.
- Unlike C and C++, Java does not use an array of characters to represent a string.
- The String class is used to represent a string that is fairly static, changing infrequently if at all.
- An object of the String class represents a string of characters.
- Like other classes, String has constructors and methods.

Constructors for the String class

Constructor	*Purpose*
String()	Creates an empty string.
String(String)	Creates a string from the specified string.
String(char[])	Creates a string from an array of characters.
String(char[], int, int)	Creates a string from the specified subset of characters in an array.
String(byte[], int)	Creates a string from the specified byte array and Unicode upper byte.
String(byte[],int,int,int)	Creates a string from the specified subset of bytes and Unicode upper byte.
String(StringBuffer)	Creates a string.

- Examples of using these constructors :

1. String str1 = new String();
2. String str2 = new String("A New String");
3. char charArray[] = {'A', 'r', 'r', 'a', 'y' };
4. String str3 = new String(charArray);
5. String str4 = new String(charArray, 2, 3);
6. StringBuffer buf = new StringBuffer("buffer");
7. String str5 = new String(buf);

- In the first example, str1 is created as an empty string.
- The second string, str2, will hold the text "A New String".
- The next two examples, str3 and str4, are constructed from a character array.
- In the case of str3, the entire array is placed in the string.
- For str4, three characters starting in array position two are copied into the string.

Basic String Methods

- The Java String class is meant to hold text that does not change.
- Following table does include some of the methods for adding and inserting text.
- It does, however, include a simple concat() method for adding one string to the end of another and a replace() method for swapping one character for another.

Basic string methods	
Method	***Purpose***
concat(String)	Concatenates one string onto another.
length()	Returns the length of the string.
replace(char, char)	Replaces all occurrences of one character with a different character.
toLowerCase()	Converts the entire string to lowercase.
toUpperCase()	Converts the entire string to uppercase.
trim()	Trims both leading and trailing whitespace from the string.

- The concat() method appends the specified string onto the current string and returns the result as a new string.
- For example :

```
String str1 = new String("Hello ");
String str2 = new String("World");
String str3 = str1.concat(str2);
```

- Output : Hello World

- The replace() method can be used to change all occurrences of one character to a different character.
- For example :

```
String str = new String("Hi Mom ");
String newStr = str.replace('o', 'u');
```

- Output : Hi Mum

- Like concat(), trim() method works by returning a new string.
- In this case, all leading and trailing whitespace characters are first removed from the returned string.
- For example :

```
String str = new String("\t\t In The Middle \r\n");
String newStr = str.trim();
```

- Output : In The Middle

- The length() method simply returns the number of characters in the string.
- For example :

```
String str = new String("Length of 12");
int len = str.length();
```

- Output : 12

- The toUpperCase() and toLowerCase() methods can be used to change the case of an entire string.
- Each works by returning a new string.

- For example :

```
String str = new String("This is MiXeD caSE");
String upper = str.toUpperCase();
String lower = str.toLowerCase();
```

- Output : THIS IS MIXED CASE (for upper case)
- Output : this is mixed case (for lower case)

Using Only Part of a String

- The Java String class provides methods for accessing or creating a substring from a longer string.

- Each of the methods listed in next table can be used to retrieve a portion of a string.

Methods for using a substring.	
Method	***Purpose***
charAt(int)	Returns the character at the specified location.
substring(int)	Returns a substring beginning at the specified offset of the current string.
substring(int, int)	Returns a substring between the specified offsets of the current string.

- The charAt() method retrieves the single character at the index given.

- For example : The following will place the character a in ch:

```
String str = new String("This is a String");
char ch = str.charAt(8);
```

- Output : a

- The two substring methods can be used to create new strings that are extracted from the current string.
- For example: To create a substring from an offset to the end of the string, use the first version of substring(), as follows:

```
String str = new String("This is a String");
String substr = str.substring(10);
```

- Output : String

- To create a substring from the middle of a string, you can specify an ending offset as an additional parameter to substring.
- For example :

```
String str = new String("Wish You Were Here");
String substr = str.substring(5, 13);
```

- Output : You Were

Comparing Strings

- Java provides a number of methods for comparing one string to another.

Method	Purpose
compareTo(String)	Compares two strings. Returns 0 if they are equal, a negative value if the specified string is greater than the string, or a positive value otherwise.
endsWith(String)	Returns true if the string ends with the specified string.
equalIgnoreCase(String)	Returns true if, ignoring differences in capitalization, the string matches the specified string.
equals(Object)	Returns true if the string matches the object.
equalTo(String)	Returns true if the string matches the specified string.
startsWith(String)	Returns true if the string starts with the specified string.
startsWith(String, int)	Returns true if the string starts with the specified string at the specified offset.

- The compareTo() method returns the difference between two strings by examining the first two characters that differ in the strings.
- The difference between the characters is returned.
- For example:

```
String str1 = new String("abc");
String str2 = new String("abe");
// compare str2 against str1
int result1 = str1.compareTo(str2);
```

- Output : -2

```
// perform the same comparison in the reverse way(str1
   against str2)
int result2 = str2.compareTo(str1);
```

- Output : 2

- The endsWith() method can be used to determine whether a string ends with a given string.
- The startsWith() method can be used to determine whether a string starts with a given string or whether that string appears at a specific location within the string.
- For Example:

```
// create two Strings
String str1 = new String("My favorite language is Java");
String str2 = new String("I like the Java language");
 // see if str1 ends with "Java"
boolean result1 = str1.endsWith("Java"); // true
// see if str1 starts with "My"
boolean result2 = str1.startsWith("My"); // true
 // see if starting in offset 11 str2 starts with "Java"
boolean result3 = str2.startsWith("Java", 11); // true
```

- To perform a case-insensitive comparison use the equalIgnoreCase() method, as follows:

```
String str1 = new String("abc");
boolean result = str1.equalsIgnoreCase("ABC");
```

- The regionMatches() method can be used to see whether a region in one string matches a region in a different string.
- The second regionMatches() method allows for case-insensitive comparisons of this nature.
- For example:

```
// create two longer Strings
String str1 = new String("My favorite language is Java");
String str2 = new String("I like the Java language");
 // compare regions
// Start at offset 24 in str1 and compare against offset 11
// in str2 for 4 characters
boolean result = str1.regionMatches(24, str2, 11, 4); // true
```

- In this case the strings are compared for a length of four characters starting with index 24 in str1 and index 11 in str2. Because each of these substrings is "Java", result is set to true.

Method	Purpose
indexOf(int)	Searches for the first occurrence of the specified character.
indexOf(char, int)	Searches for the first occurrence of the specified character following the given offset.
indexOf(String)	Searches for the first occurrence of the specified string.
indexOf(String, int)	Searches for the first occurrence of the specified string following the given offset.
lastIndexOf(int)	Searches backwards for the last occurrence of the specified character.
lastIndexOf(char, int)	Searches backwards for the last occurrence of the specified character preceding the given offset.
lastIndexOf(String)	Searches backwards for the last occurrence of the specified string.
lastIndexOf(String, int)	Searches backwards for the last occurrence of the specified string preceding the given offset.

The StringBuffer Class

The StringBuffer Class

- StringBuffer is a peer class of String class that provides much more functionality for string.
- The primary limitation of the string class is that once the string is created you cannot change it.
- If you need to store text that may need to be changed you should use the StringBuffer class.
- The StringBuffer class includes methods for inserting and appending text.
- StringBuffer allocates 16 additional characters when no specific buffer length is requested.

StringBuffer constructors

Constructor	*Purpose*
StringBuffer()	Creates an empty StringBuffer. And reserved 16 char without reallocation.
StringBuffer(int)	Creates an empty StringBuffer with the specified length.
StringBuffer(String)	Creates a StringBuffer based on the specified string and reserved 16 more characters without reallocation.

Useful StringBuffer Methods

- length() :- returns the current length of a string.
- capacity() :- returns the total allocated capacity
- charAt(int):- Returns the character located at the specified index
- setCharAt(int, char) : - Sets the value at the specified index to the specified character.
- append (string) :- append specified string to the end of string
- insert(int, String) :- insert string at specific location
- reverse() :- returns reverse characters in a string
- delete(int, int) :- delete characters from starting to end offset
- delete(charAt(int)):-delete character at specific location
- Replace(int si, int li, string str) :- It will replace substring at si to (li-1).