Unit 6:

Handling Strings

Java String :

- String is basically an object that represents sequence of char values.
- The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created.
- For mutable strings, you can use StringBuffer and StringBuilder classes.

There are two ways to create String object:

1. By string literal :

- Java String literal is created by using double quotes.
- Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.
- Whenever a String Object is created as a literal, the object will be created in **String constant pool (*special memory*).**

2. By new keyword :

- The string can also be declared using **new** operator i.e. dynamically allocated.
- In case of String are dynamically allocated they are assigned a new memory location in heap. This string will not be added to String constant pool.

Conversion of String :

Conversion of any type to String we use :
1. String.valueOf
2. toString()

Conversion of String to any Type :

1. parse()

Syntax :
DataType.parseDataType(String s)

e.g.

```
double ageDouble =  Double.parseDouble("18");
```


String Character Extraction Methods :
- ❖ String class provides a number of ways in which characters can be extracted from a String object.
- ❖ Below are various ways to do so:

1. String.charAt(index)
2. String.toCharArray()
3. String.getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
4. String.getBytes()

String Comparision :

There are three ways to compare String in Java:

1. By Using equals() Method :

- The String class equals() method compares the original content of the string. It compares values of string for equality.
- String class provides the following two methods:

- public boolean equals(Object another) :
- *Compares two strings. Returns true if the strings are equal, and false if not.*

- public boolean equalsIgnoreCase(String another) :
- *Compares two strings, ignoring case considerations.*

2. By Using == Operator :

The == operator compares references not values.

true : If both the strings refer to same instance.
false : If both the strings doesn't not refer to same instance.

3. By compareTo() Method :

The String class compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two String objects. If:

- **s1 == s2** : The method returns 0.
- **s1 > s2** : The method returns a positive value.
- **s1 < s2** : The method returns a negative value.

Searching String:

❖ The String class provides 2 methods for searching a string.

They are :

1. indexOf() : ***Searches for the first occurrence of a character or substring.***
2. lastIndexOf() : ***Searches for the last occurrence of a character or substring.***

There are many overloaded forms of these methods:

Syntax	Explanation
int indexOf(char ch)	This method will return the index of the first occurrence of a character variable `ch` in the invoked string.
int lastIndexOf(char ch)	This method will return the index of the last occurrence of a character variable `ch` in the invoked string.
int indexOf(String st)	This method will return the index of the first occurrence of a `substring st` in the invoked string.
int lastIndexOf(String st)	This method will return the index of the last occurrence of a `substring st` in the invoked string.
int indexOf(char ch, int startIndex) ***int indexOf(String st, int startIndex)***	Here `startIndex` specifies the starting point of search. The search runs from `startIndex`to end of the String.
int lastIndexOf(char ch, int startIndex) ***int lastIndexOf(String st, int startIndex)***	Here `startIndex` specifies the starting point of search. The search runs from `startIndex` to zero.

String Modifying Methods :

1. substring() :
Returns a new string which is the substring of a specified string

2. concat() :
Appends a string to the end of another string

3. replace() :
Searches a string for a specified value, and returns a new string where the specified values are replaced.

4. replaceAll() :
Replaces each substring of this string that matches the given regular expression with the given replacement

5. replaceFirst() :
Replaces the first occurrence of a substring that matches the given regular expression with the given replacement.

6. trim() :
Removes whitespace from both ends of a string.

Java StringBuffer and StringBuilder class:

❖ Java provides three classes to represent a sequence of characters:

1. String
2. StringBuffer
3. StringBuilder.

The **String** class is an immutable class whereas **StringBuffer** and **StringBuilder** classes are mutable.

No.	StringBuffer	StringBuilder
1)	StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.
3)	StringBuffer was introduced in Java 1.0	StringBuilder was introduced in Java 1.5

Java toString() Method :

- The toString() method returns the String representation of the object.
- If you print any object, Java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depending on your implementation.

- ***Note:*** *Default behavior of toString() is to print class name, then @, then unsigned hexadecimal representation of the hash code of the object.*

Advantage :

- By overriding the toString() method of the Object class, we can return values of the object, so we don't need to write much code.

Java String Methods :

Method	Description	Return Type
charAt()	Returns the character at the specified index (position)	char
codePointAt()	Returns the Unicode of the character at the specified index	int
codePointBefore()	Returns the Unicode of the character before the specified index	int
codePointCount()	Returns the number of Unicode values found in a string.	int
compareTo()	Compares two strings lexicographically	int
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences	int
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean

Method	Description	Return Type
contentEquals()	Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer	boolean
copyValueOf()	Returns a String that represents the characters of the character array	String
endsWith()	Checks whether a string ends with the specified character(s)	boolean
equals()	Compares two strings. Returns true if the strings are equal, and false if not	boolean
equalsIgnoreCase()	Compares two strings, ignoring case considerations	boolean
format()	Returns a formatted string using the specified locale, format string, and arguments	String
getBytes()	Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array	byte[]
getChars()	Copies characters from a string to an array of chars	void
hashCode()	Returns the hash code of a string	int
indexOf()	Returns the position of the first found occurrence of specified characters in a string	int
intern()	Returns the canonical representation for the string object	String
isEmpty()	Checks whether a string is empty or not	boolean
lastIndexOf()	Returns the position of the last found occurrence of specified characters in a string	int
length()	Returns the length of a specified string	int

Method	Description	Return Type
matches()	Searches a string for a match against a regular expression, and returns the matches	boolean
offsetByCodePoints()	Returns the index within this String that is offset from the given index by codePointOffset code points	int
regionMatches()	Tests if two string regions are equal	boolean
replace()	Searches a string for a specified value, and returns a new string where the specified values are replaced	String
replaceFirst()	Replaces the first occurrence of a substring that matches the given regular expression with the given replacement	String
replaceAll()	Replaces each substring of this string that matches the given regular expression with the given replacement	String
split()	Splits a string into an array of substrings	String[]
startsWith()	Checks whether a string starts with specified characters	boolean
subSequence()	Returns a new character sequence that is a subsequence of this sequence	CharSequence
substring()	Returns a new string which is the substring of a specified string	String
toCharArray()	Converts this string to a new character array	char[]
toLowerCase()	Converts a string to lower case letters	String
toString()	Returns the value of a String object	String
toUpperCase()	Converts a string to upper case letters	String
trim()	Removes whitespace from both ends of a string	String
valueOf()	Returns the string representation of the specified value	String