JAVA

What you learn ?

Java	
➢ String	➢ String Pool
➢ String methods	➢ StringBuffer
➢ StringBuilder	➢ String VS StringBuffer Vs StringBuilder
➢ Regular Expression	

String

- In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string.

- The java.lang.String class implements *Serializable*, *Comparable* and *CharSequence* interfaces.

- 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.

String in JAVA vs String in C++

In Java strings are immutable reference types.
In C++ strings are mutable and employ value semantics.

Two strings in C++ will evaluate true to a “==” operation if they contain the same value, regardless of whether they are the same object or not.

In Java comparing two String variables with ‘==’ does a reference equality test. Surprisingly you might declare the same two Strings and initialize them with the same string literal — then you’d expect them not to have reference equality — but they might still because of string interning.

String

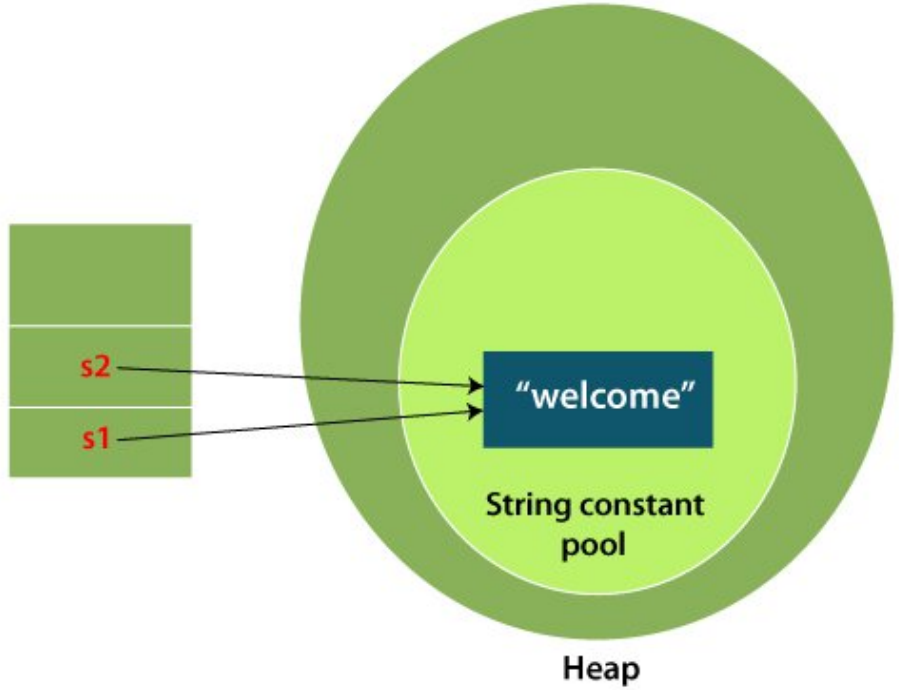
There are two ways to create String object:

1.By string literal

2.By new keyword

1) 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.



For example:

```
String s1="Welcome";
String s2="Welcome";//It doesn't create a new instanc
e
```

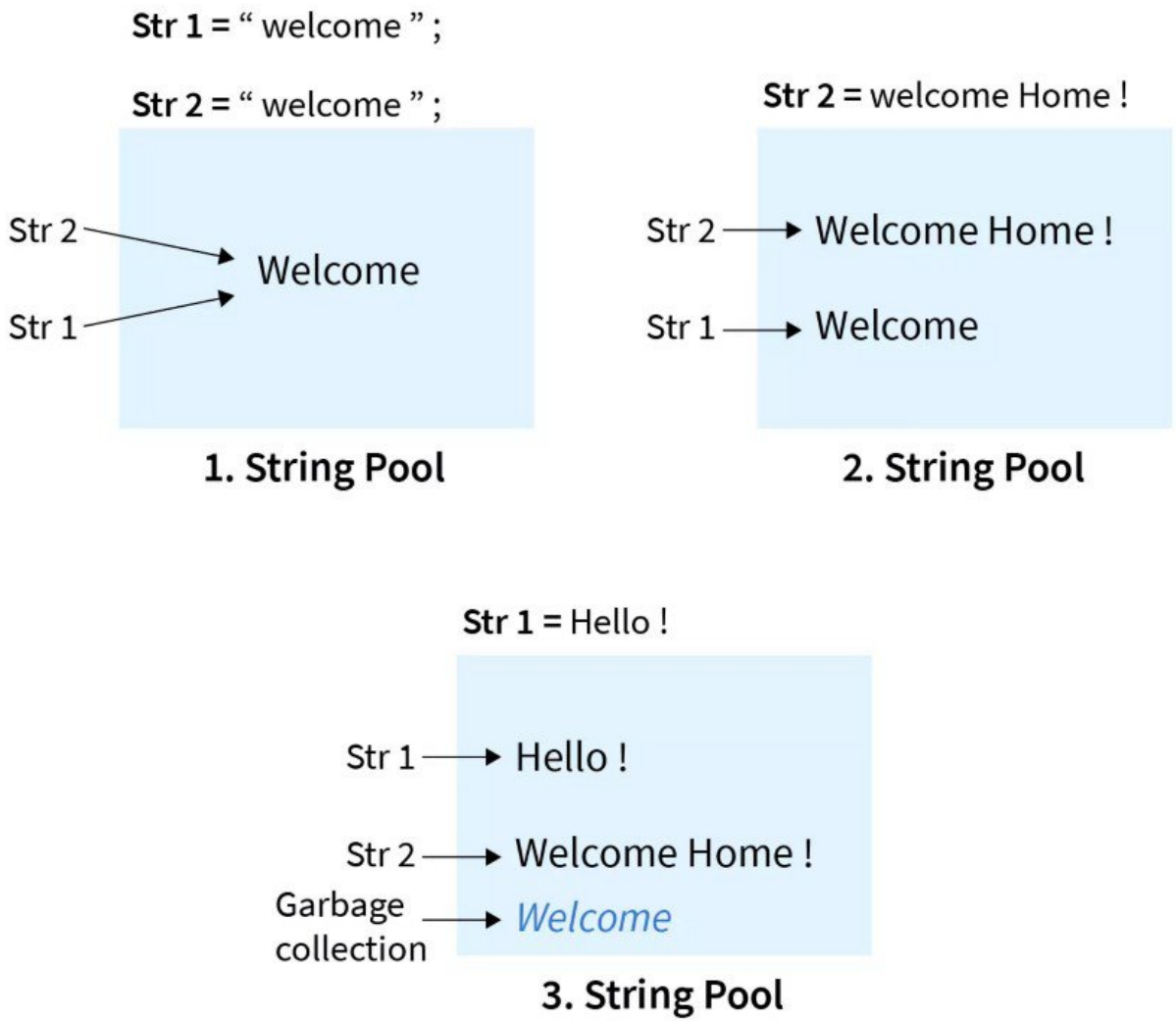
String

2) By new keyword

String s=new String("Welcome"); //creates two objects and one reference variable

In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

String Pool


Str 1 = " welcome " ;
Str 2 = " welcome " ;
Str 2
Str 1
Welcome
1. String Pool
Str 2 = welcome Home !
Str 2
Welcome Home !
Str 1
Welcome
2. String Pool
Str 1 = Hello !
Str 1
Hello !
Str 2
Welcome Home !
Garbage collection
Welcome
3. String Pool

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

String methods

Method	Description	Return Type
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean
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

String methods

Method	Description	Return Type
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

String methods

Method	Description	Return Type
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
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

String methods

Method	Description	Return Type
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
substring()	Returns a new string which is the substring of a specified string	String

String methods

Method	Description	Return Type
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
substring()	Returns a new string which is the substring of a specified string	String
subSequence()	Returns a new character sequence that is a subsequence of this sequence	CharSequence

StringBuffer

StringBuffer is a peer class of String that provides much of the functionality of strings. The string represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.

StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

StringBuffer

Important constructors of StringBuffer class:

Constructor	Description
StringBuffer()	It creates an empty String buffer with the initial capacity of 16.
StringBuffer(String str)	It creates a String buffer with the specified string..
StringBuffer(int capacity)	It creates an empty String buffer with the specified capacity as length.

StringBuffer

Methods of StringBuffer class:

Methods	Action Performed
append()	Used to add text at the end of the existing text.
length()	The length of a StringBuffer can be found by the length( ) method
capacity()	the total allocated capacity can be found by the capacity( ) method
charAt()	This method returns the char value in this sequence at the specified index.
delete()	Deletes a sequence of characters from the invoking object
deleteCharAt()	Deletes the character at the index specified by loc
ensureCapacity()	Ensures capacity is at least equals to the given minimum.
insert()	Inserts text at the specified index position
length()	Returns length of the string
reverse()	Reverse the characters within a StringBuffer object
replace()	Replace one set of characters with another set inside a StringBuffer object

StringBuilder

StringBuilder in Java represents a mutable sequence of characters.

Since the String Class in Java creates an immutable sequence of characters, the StringBuilder class provides an alternative to String Class, as it creates a mutable sequence of characters.

The function of StringBuilder is very much similar to the StringBuffer class, as both of them provide an alternative to String Class by making a mutable sequence of characters.

However, the StringBuilder class differs from the StringBuffer class on the basis of synchronization. The StringBuilder class provides no guarantee of synchronization whereas the StringBuffer class does.

StringBuilder

Important constructors of StringBuilder class:

Constructor	Description
StringBuilder()	It creates an empty String Builder with the initial capacity of 16.
StringBuilder(String str)	It creates a String Builder with the specified string.
StringBuilder(int length)	It creates an empty String Builder with the specified capacity as length.

StringBuilder

Methods of StringBuilder class:

Methods	Action Performed
public StringBuilder append(String s)	It is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
public StringBuilder insert(int offset, String s)	It is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
public StringBuilder replace(int startIndex, int endIndex, String str)	It is used to replace the string from specified startIndex and endIndex.
public StringBuilder delete(int startIndex, int endIndex)	It is used to delete the string from specified startIndex and endIndex.
public StringBuilder reverse()	It is used to reverse the string.

StringBuilder

Methods of StringBuilder class:

Methods	Action Performed
public int capacity()	It is used to return the current capacity.
public void ensureCapacity(int minimumCapacity)	It is used to ensure the capacity at least equal to the given minimum.
public char charAt(int index)	It is used to return the character at the specified position.
public int length()	It is used to return the length of the string i.e. total number of characters.
public String substring(int beginIndex)	It is used to return the substring from the specified beginIndex.
public String substring(int beginIndex, int endIndex)	It is used to return the substring from the specified beginIndex and endIndex.

String VS StringBuffer VS StringBuilder

String	StringBuffer	StringBuilder
Immutable/not changed	Objects are mutable/we can change	Objects are mutable/we can change
have **concat** method	have **append** method	have **append** method
equals() method meant for content comparison	**equals()** method meant for reference/address comparison	**equals()** method meant for reference/address comparison
there is no any **capacity** concept	default **capacity** is 16	default **capacity** is 16
	every method present in StringBuffer is Synchronized	no method present in StringBuilder is Synchronized
String is thread safe (All immutable object bydefault thread safe because no one can change value)	at a time only one thread is allow to operate on StringBuffer object and hence it is thread safe	at a time multiple thread are allowed to operate on StringBuilder object and hence it is not thread safe
	threads are required to wait to operate on StringBuffer object and hence relatively performance slow	threads are not required to wait to operate on StringBuilder object and hence relatively performance is high
	introduced in **1.0v**	introduced in **1.5v**

String VS StringBuffer VS StringBuilder

String	StringBuffer	StringBuilder
Immutable/not changed	Objects are mutable/we can change	Objects are mutable/we can change
have **concat** method	have **append** method	have **append** method
equals() method meant for content comparison	**equals()** method meant for reference/address comparison	**equals()** method meant for reference/address comparison
there is no any **capacity** concept	default **capacity** is 16	default **capacity** is 16
	every method present in StringBuffer is Synchronized	no method present in StringBuilder is Synchronized
String is thread safe (All immutable object bydefault thread safe because no one can change value)	at a time only one thread is allow to operate on StringBuffer object and hence it is thread safe	at a time multiple thread are allowed to operate on StringBuilder object and hence it is not thread safe
	threads are required to wait to operate on StringBuffer object and hence relatively performance slow	threads are not required to wait to operate on StringBuilder object and hence relatively performance is high
	introduced in **1.0v**	introduced in **1.5v**

Regular Expression

- Regular Expressions or Regex (in short) in Java is an API for defining String patterns that can be used for searching, manipulating, and editing a string in Java.
- Email validation and passwords are a few areas of strings where Regex is widely used to define the constraints.
- Regular Expressions are provided under java.util.regex package.
- This consists of 3 classes and 1 interface.

S. No.	Class/Interface	Description
1.	Pattern Class	Used for defining patterns
2.	Matcher Class	Used for performing match operations on text using patterns
3.	PatternSyntaxException Class	Used for indicating syntax error in a regular expression pattern
4.	MatchResult Interface	Used for representing the result of a match operation

A Regular Expression (RegEx) is a sequence of characters that defines a search pattern.
A pattern defined using RegEx can be used to match against a string.

Expression	String	Matched?
^a...s$	abs	No match
	alias	Match
	abyss	Match
	Alias	No match
	An abacus	No match

The pattern is: **any five letter string starting with '*a*' and ending with '*s*'**

^ Caret

The caret symbol ^ is used to check if a string starts with a certain character.

Expression	String	Matched?
^a	a	1 match
	abc	1 match
	bac	No match
^ab	abc	1 match
	acb	No match (starts with a but not followed by b)

. Period

A period matches any single character (except newline '\n').

Expression	String	Matched?
..	a	No match
	ac	1 match
	acd	1 match
	acde	2 matches (contains 4 characters)

MetaCharacters

Metacharacters are characters that are interpreted in a special way by a RegEx engine. Here's a list of metacharacters:

[] . ^ $ * + ? {} () \ |

[] - Square brackets

Square brackets specifies a set of characters you wish to match

Expression	String	Matched?
[abc]	a	1 match
	ac	2 matches
	Hey Juderr	No match
	abc de caty	5 matches

[] - Square brackets

[abc] will match if the string you are trying to match contains any of the a, b or c.

You can also specify a range of characters using - inside square brackets.

- [a-e] is the same as [abcde].
- [1-4] is the same as [1234].
- [0-39] is the same as [01239].

You can complement (invert) the character set by using caret ^ symbol at the start of a square-bracket.

- [^abc] means any character except a or b or c.
- [^0-9] means any non-digit character.

$ Dollar

The dollar symbol $ is used to check if a string ends with a certain character.

Expression	String	Matched?
a$	a	1 match
	formula	1 match
	cab	No match

* Star

The star symbol * matches zero or more occurrences of the pattern left to it.

Expression	String	Matched?
ma*n	mn	1 match
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

+ Plus

The plus symbol + matches one or more occurrences of the pattern left to it.

Expression	String	Matched?
ma+n	mn	No match (no a character)
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

? Question Mark

The question mark symbol ? matches zero or one occurrence of the pattern left to it.

Expression	String	Matched?
ma?n	mn	1 match
	man	1 match
	maaan	No match (more than one a character)
	main	No match (a is not followed by n)
	woman	1 match

{} Braces

Consider this code: {n,m}. This means at least n, and at most m repetitions of the pattern left to it.

Expression	String	Matched?
a{2,3}	abc dat	No match
	abc daat	1 match (at daat)
	aabc daaat	2 matches (at aabc and daaat)
	aabc daaaat	2 matches (at aabc and daaaat)

Expression	String	Matched?
[0-9]{2,4}	ab123csde	1 match (match at ab123csde)
	12 and 345673	3 matches (12, 3456, 73)
	1 and 2	No match

| Alternation

Vertical bar | is used for alternation (or operator).

Expression	String	Matched?
a|b	cde	No match
	ade	1 match (match at ade)
	acdbea	3 matches (at acdbea)

Here, a|b match any string that contains either a or b

() Group

Parentheses () is used to group sub-patterns. For example, (a|b|c)xz match any string that matches either a or b or c followed by xz

Expression	String	Matched?
(a|b|c)xz	ab xz	No match
	abxz	1 match (match at a<u>bxz</u>)
	axz cabxz	2 matches (at <u>axz</u>bc ca<u>bxz</u>)

\ Backslash

Backlash \ is used to escape various characters including all metacharacters. For example,

\$a match if a string contains **$** followed by a. Here, **$** is not interpreted by a RegEx engine in a special way.
If you are unsure if a character has special meaning or not, you can put \ in front of it. This makes sure the character is not treated in a special way.

\A - Matches if the specified characters are at the start of a string.

\b - Matches if the specified characters are at the beginning or end of a word.

\B - Opposite of \b. Matches if the specified characters are not at the beginning or end of a word.

\d - Matches any decimal digit. Equivalent to [0-9]

\ Backslash

\D - Matches any non-decimal digit. Equivalent to [^0-9]

\s - Matches where a string contains any whitespace character. Equivalent to [\t\n\r\f\v].

\S - Matches where a string contains any non-whitespace character. Equivalent to [^ \t\n\r\f\v].

\w - Matches any alphanumeric character (digits and alphabets). Equivalent to [a-zA-Z0-9_]. By the way, underscore _ is also considered an alphanumeric character.

\W - Matches any non-alphanumeric character. Equivalent to [^a-zA-Z0-9_]

\Z - Matches if the specified characters are at the end of a string.

\ Backslash

In many programming languages, the backslash character (\) is used to indicate an escape sequence, which allows special characters to be interpreted in a specific way. For example, \n is an escape sequence that represents a newline character, and \t represents a tab character.

However, the backslash character itself also needs to be escaped when it is used as a literal character. So if you want to match a backslash character in a regular expression, you would need to use two backslashes in a row (\\).

Thanks

Anirudha Gaikwad