Java Training
String Basics

About me

- Lead Software Engineer
- Working for EPAM more than 10 years
- Love travelling: the most remarkable countries I've visited are Iceland, Ireland and Kenia. Miss a lot shop tours to **Vilnius** :)
- Have Bentley :)

Today’s agenda

- String definition, String Creation
- String Allocation (String Pool)
- String Immutability
- String Manipulations
- Useful things: Converting, Formatting, Switch statement
- String Builder (String Buffer)

Revision

- What Java Data types have you learnt?
- What primitive type char is used for?
- What do you remember about array?
- What was the first program you've created when you started to learn Java? :)

Hello world!

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!"); //output: Hello world!
    }
}
```

"Hello world!" is a ***string literal*** -
a series of characters in your code that is enclosed in double quotes

String definition

- String is a sequence of zero or more characters.
- String is represented by an object of the java.lang.String class.
- The String class is immutable, so that once it is created String object cannot be changed

Useful links:

- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html
- https://docs.oracle.com/javase/tutorial/java/data/strings.html

String Creation

```
String greeting1 = "Hello world!"; //as string literal

String greeting2 = new String("Hello world!"); //using constructor

char[] greetingArray = { 'H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!' };
String greeting3 = new String(greetingArray); //more than 10 different constructors

System.out.println(greeting1); //output: Hello world!
System.out.println(greeting2); //output: Hello world!
System.out.println(greeting3); //output: Hello world!
```

Useful links:

- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html
- https://docs.oracle.com/javase/tutorial/java/data/strings.html

What if.. I need something special

Display String with double quotes:

My friend said: "I love pizza!!!"

```
String s1 = "My friend said: "I love pizza!!!"";
```

Display text on a new line:

This is my first line.
This is my second line

```
String s2 = "This is my first line." +
            "This is my second line";
//Output: This is my first line.This is my second line
```

What if.. I need something special

Use escape characters!

```
String s1 = "My friend said: \"I love pizza!!!\"";
System.out.println(s1);


String s2 = "This is my first line.\nThis is my second line";
System.out.println(s2);
```

```
Output:
My friend said: "I love pizza!!!"

This is my first line.
This is my second line
```

Escape Sequences

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

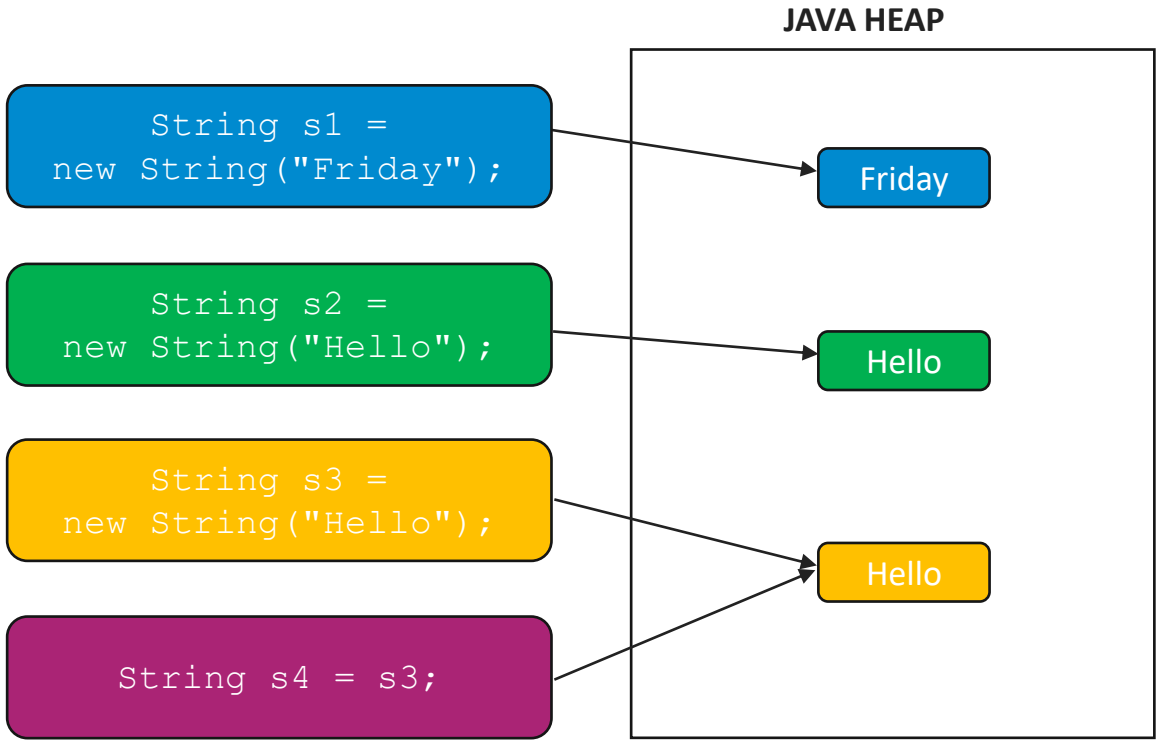
Useful link:
https://docs.oracle.com/javase/tutorial/java/data/characters.html

Strings and Java Memory

Where do we store Java objects?

What's the difference between **==** and **equals**?

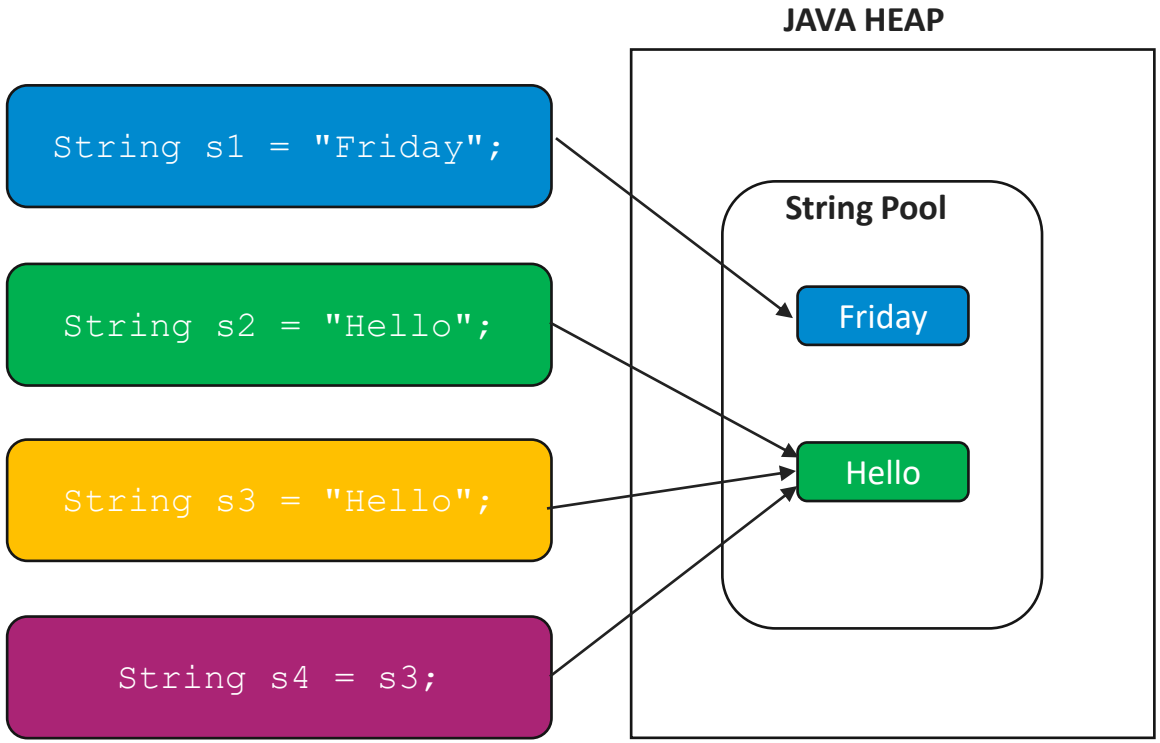
Strings and Java Memory




```
s1 == s2; // false
s2 == s3; // false
s3 == s4; // true

s1.equals(s2); //false
s2.equals(s3); // true
s3.equals(s4); // true
```

Strings and Java Memory




```
s1 == s2; // false
s2 == s2; // true
s3 == s4; // true

s1.equals(s2); //false
s2.equals(s3); // true
s3.equals(s4); // true
```

Strings and Java Memory

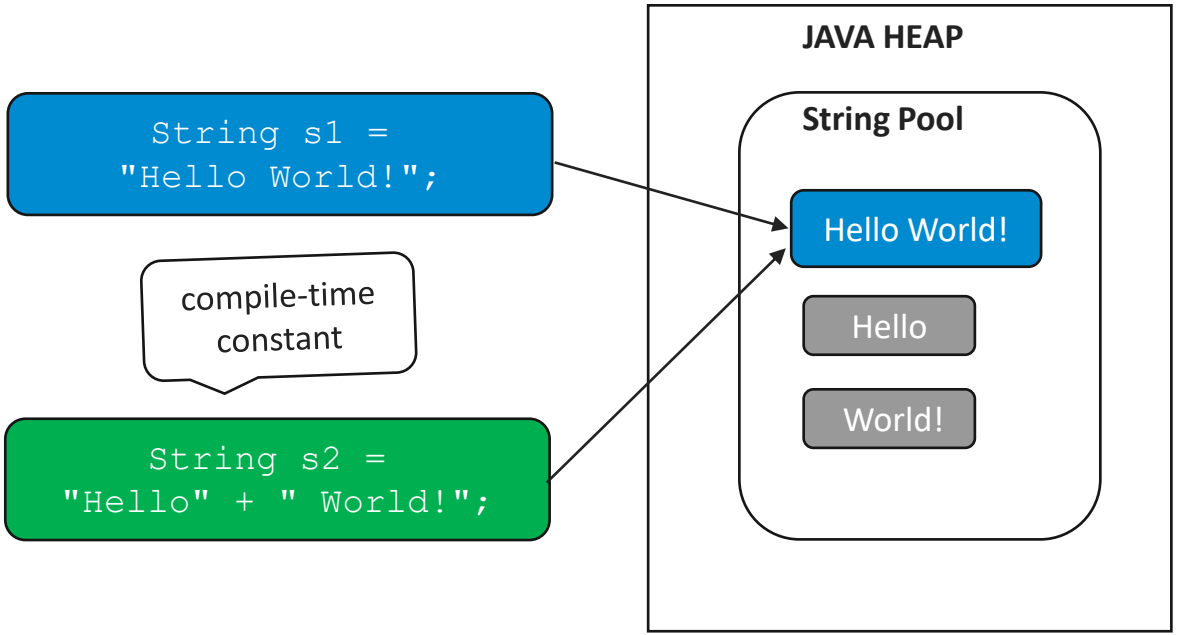
String Pool is the special memory region where all string literals are stored.

- Java maintains a pool of all string literals in order to minimize the memory usage and for better performance. It creates a String object in the string pool for every string literal it finds in a program.
- String pool helps in saving a lot of space for Java Runtime although it takes more time to create the String

How it works:

- When Java encounters a string literal, it looks for a string object in the string pool with the identical content.
- If it does not find a match in the string pool, it creates a new String object with that content and adds it to the string pool.
- Finally, it replaces the string literal with the reference of the newly created String object in pool.
- If it finds a match in the string pool, it replaces the string literal with the reference of the String object found in the pool.

Strings and Java Memory




```
s1 == s2; // true
s1.equals(s2); //true
```

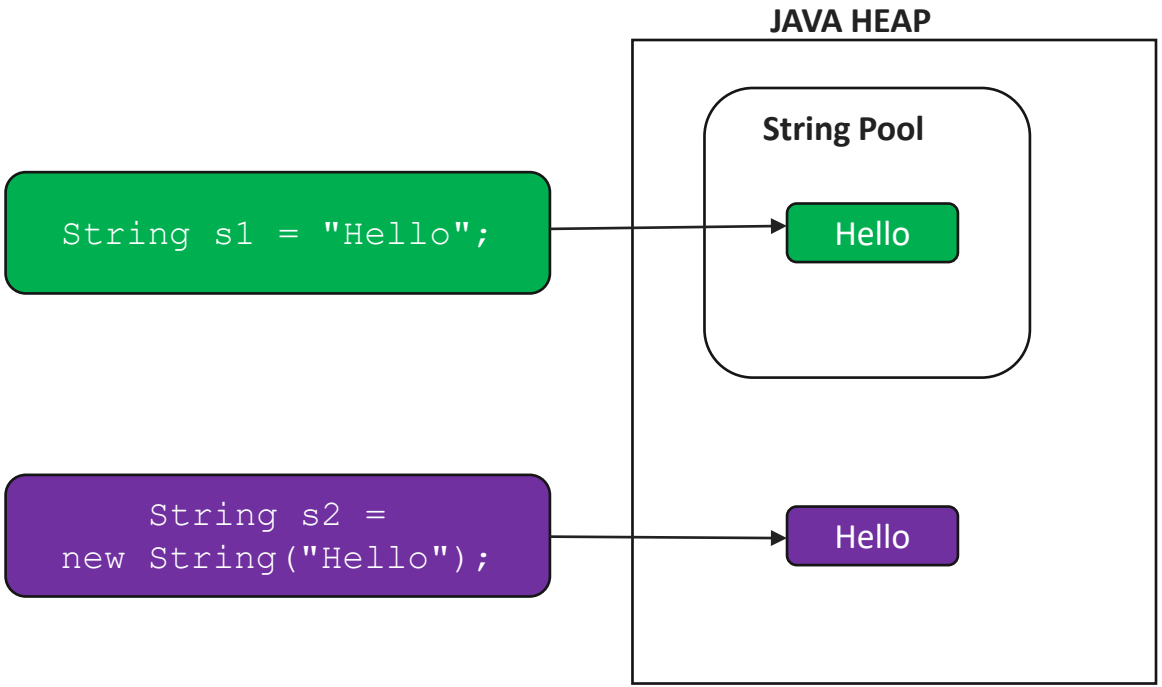
All string literals and string literals resulting from compile-time constant expressions are added to the string pool!

Strings and Java Memory

```
final String constStr = "Constant"; // constStr is a constant
// "Constant is pooled" will be added to the string pool
String s1 = constStr + " is pooled";
System.out.println("Constant is pooled" == s1); //true


// Concatenated string will not be added to the string pool
String s2 = "Variable" + " is not pooled";
String varStr = "Variable"; // varStr is not a constant
String s3 = varStr + " is not pooled";
System.out.println("Variable is not pooled" == s2); //true
System.out.println("Variable is not pooled" == s3); //false
```

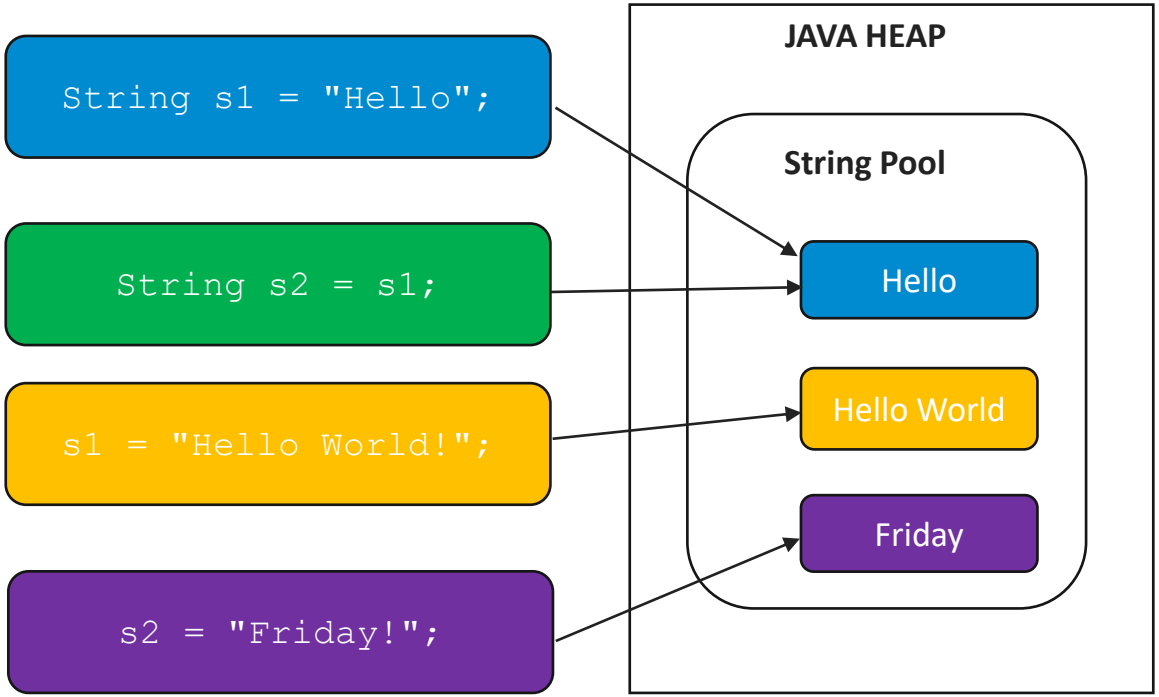
Strings and Memory




```
s1 == s2; // false
s1.equals(s2); //true
s1 == s2.intern(); //true
```

`intern()` method puts the string into the pool or refer to another String object from the string pool having the same value

String Immutability



String objects are immutable!

- Once you have assigned a String a value, that value can never change – it's Immutable, frozen solid
- But while the String object is immutable, it's reference variable is not.

What benefits does String Immutability give us?

String Manipulations

The class String includes methods for:

- examining individual characters of the sequence
- comparing strings
- searching strings
- extracting substrings
- creating a copy of a string with all characters translated to uppercase or to lowercase

Note:

Since strings are immutable, what methods that appear to modify string really do is create and return a new string (read new OBJECT) that contains the result of the operation.

String Manipulations: Basic methods

- length() Returns the number of characters in a String
- charAt() Returns the character located at the specified index.
- concat() Appends one String to the end of another ("+" also works)
- toString() Returns the value of a String

String s = “Hello World”;

	H	E	L	L	O		W	O	R	L	D	!
position/index	0	1	2	3	4	5	6	7	8	9	10	11

Indexes count from 0!

12 characters – the length
First character (H) is on the **ZERO** position
Last chnaracters (!) on **ELEVENTH** position

Basic methods

```java
//length() - Returns the number of characters in a String
System.out.println("Hello".length()); //5
System.out.println("Hello World!!!".length()); //14
System.out.println("01234567".length()); //8
//TODO  ""

//charAt(int index) - Returns the character located at the specified index
System.out.println("01234567".charAt(0)); //0
System.out.println("01234567".charAt(1)); //1
//System.out.println("01234567".charAt(8)); //StringIndexOutOfBoundsException
String hello = "Hello!";
System.out.println(hello.charAt(4)); //?
//TODO print last character

//concat(String s) Appends one String to the end of another ( "+" also works)
String welcome = "Welcome";
System.out.println(welcome.concat(" home!")); //Welcome home!.
System.out.println(welcome + " home!"); //Welcome home!
welcome += " home!";
System.out.println(welcome); //Welcome home!

//toString() Returns the value of a String
System.out.println("Looks odd to use, but possible".toString());
```

String manipulations

The substring Methods	Description
String substring(int beginIndex, int endIndex)	Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1.
String substring(int beginIndex)	Returns a new string that is a substring of this string. The integer argument specifies the index of the first character. Here, the returned substring extends to the end of the original string.

```
//substring() Returns a new string that is a part of the original string
//substring(int beginIndex) Returned substring from beginIndex to the end
//substring(int beginIndex, int endIndex) Returned substring from beginIndex to the endIndex - 1
String str = "0123456789";
System.out.println(str.substring(5)); // "56789"
System.out.println(str.substring(5, 8)); // "567"
//TODO what will be output: "Niagara. O roar again!".substring(11, 15);
```

String manipulations

More Methods for Manipulating	Description
String[] split(String regex) String[] split(String regex, int limit)	Searches for a match as specified by the string argument (which contains a regular expression) and splits this string into an array of strings accordingly. The optional integer argument specifies the maximum size of the returned array. Regular expressions are covered in the lesson titled "Regular Expressions."
String trim()	Returns a copy of this string with leading and trailing white space removed.
String toLowerCase() String toUpperCase()	Returns a copy of this string converted to lowercase or uppercase. If no conversions are necessary, these methods return the original string.

Useful link:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html#sum

String manipulations

```java
//split(String regex)  splits this string into an array by regexp
String longString = "Hello my dear friend";//TODO ask to provide possible example of usage?
String[] arr = longString.split(" ");
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}

//join Returns a new String composed of copies of the CharSequence elements joined together with a copy
of the specified delimiter.
String[]  arr2 = {"Hello", "my", "dear", "friend"};
String joinedArr = String.join("_", arr2);
System.out.println(joinedArr);

//trim() Returns a copy of this string with leading and trailing white space removed.
String myString = "        Too many spaces from both ends    ";
System.out.println("'" + myString + "'");
System.out.println("'" + myString.trim() + "'");

//toLowerCase()/toUpperCase() Returns a copy of this string converted to lowercase or uppercase
String abc = "AbbCddEffGhhIjjKllM";
System.out.println(abc.toLowerCase()); //abbcddeffghhijjkllm
System.out.println(abc.toUpperCase()); //ABBCDDEFFGHHIJJKLLM
```

String manipulations

The Search Methods	Description
int indexOf(int ch) int lastIndexOf(int ch)	Returns the index of the first (last) occurrence of the specified character.
int indexOf(int ch, int fromIndex) int lastIndexOf(int ch, int fromIndex)	Returns the index of the first (last) occurrence of the specified character, searching forward (backward) from the specified index.
int indexOf(String str) int lastIndexOf(String str)	Returns the index of the first (last) occurrence of the specified substring.
int indexOf(String str, int fromIndex) int lastIndexOf(String str, int fromIndex)	Returns the index of the first (last) occurrence of the specified substring, searching forward (backward) from the specified index.
boolean contains(CharSequence s)	Returns true if the string contains the specified character sequence.

String manipulations

```
//indexOf(..)/lastIndexOf(..) return the position within the string of a specific character or substring
//indexOf() searches forward from the beginning of the string, lastIndexOf() searches backward from the
end of the string.
// IMPORTANT!!!! If a character or substring is not found, indexOf() and lastIndexOf() return -1.
String str = "I love programming!!!";
System.out.println(str.contains("love")); //true
System.out.println(str.contains("Hate")); //false

System.out.println(str.indexOf("Hate")); //-1
System.out.println(str.indexOf("love")); // 2
System.out.println(str.lastIndexOf("love")); //2

String str2 = "I love, very very love programming and I love pizza too!";
System.out.println(str2.indexOf("love")); // 2
System.out.println(str2.lastIndexOf("love")); //41

System.out.println(str2.indexOf("love", 5)); // 18

//TODO try next method: indexOf(int ch)
```

String manipulations

Method for finding characters or substrings within a string.	Description
String replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
String replace(CharSequence target, CharSequence replacement)	Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
String replaceAll(String regex, String replacement)	Replaces each substring of this string that matches the given regular expression with the given replacement.
String replaceFirst(String regex, String replacement)	Replaces the first substring of this string that matches the given regular expression with the given replacement.

String manipulations

```
//replace - Replaces occurrences of a character or string with a new one
String str3 = "I love FastFood and I love Wine!!!";
System.out.println(str3.replace("love", "HATE")); //I HATE FastFood and I HATE Wine!!!
System.out.println(str3.replaceAll("love", "HATE")); //I HATE FastFood and I HATE Wine!!!
System.out.println(str3.replaceFirst("love", "HATE")); //I HATE FastFood and I love
Wine!!!

//TODO try replace(char oldChar, char newChar), for example replace o with X
//TODO replace letter I with word People


//TODO print extension and filename from given path:  /home/user/index.html
```

String manipulations

Method for Comparing Strings	Description
boolean endsWith(String suffix) boolean startsWith(String prefix)	Returns true if this string ends with or begins with the substring specified as an argument to the method.
boolean startsWith(String prefix, int offset)	Considers the string beginning at the index offset, and returns true if it begins with the substring specified as an argument.
int compareTo(String anotherString)	Compares two strings lexicographically. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument.
int compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring differences in case. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument.
boolean equals(Object anObject)	Returns true if and only if the argument is a String object that represents the same sequence of characters as this object.
boolean equalsIgnoreCase(String anotherString)	Returns true if and only if the argument is a String object that represents the same sequence of characters as this object, ignoring differences in case.
boolean matches(String regex)	Tests whether this string matches the specified regular expression. Regular expressions are discussed in the lesson titled "Regular Expressions."

String manipulations

```java
//endWith/startWith - Returns true if this string ends with or begins with the substring specified as an argument to the method.
String protocol = "http://";
String requestedPage = "index.html";
String myServer = "http://myserver/"; //String myServer = "myserver";
//Goal: http://myserver/index.html
if (myServer.startsWith(protocol) && myServer.endsWith("/")) {
    System.out.println(myServer + requestedPage);
} else {
    System.out.println(protocol + myServer + "/" + requestedPage);
}

//equals/equalsIgnoreCase
String str1 = "Happy Friday!";
String str2 = "happy friday!";
System.out.println(str1.equals(str2)); //false
System.out.println(str1.equalsIgnoreCase(str2)); //true //TODO any idea why we need it and waht to be not so strict?

// matches(String regex) Tests whether this string matches the specified regular expression.
System.out.println("12345678".matches("\\d+")); //checks if string contains only digits - true
System.out.println("Line with numbers 12345678".matches("\\d+")); //checks if string contains only digits - false
System.out.println("Line with numbers 12345678".matches(".*\\d.*")); //checks if string contains digits -true
System.out.println("Line without numbers numbers".matches(".*\\d.*")); ////checks if string contains digits -false
```

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html#sum

Exercise

```
String s1 = "spring ";
String s2 = s1 + "summer ";
s1.concat("fall ");
s2.concat(s1);
s1 += "winter ";
System.out.println(s1 + " " + s2);
```

Questions:

1. What is the output?

2. How many objects and reference variables there are, and which ones refer to which

Useful: Converting Between Numbers and Strings

Converting Numbers to Strings

```
float f = 4.5F;
System.out.println(f); //TODO check what inside this method, output 4.5
String strFloat1 = f + "";
String strFloat2 = String.valueOf(f);
String strFloat3 = Float.toString(f); //that will convert its primitive type to a string.
```

Converting Strings to Numbers

```
//The Number subclasses that wrap primitive numeric types each provide a class method
named valueOf
// that converts a string to an object of that type.
Float f1 = Float.valueOf("4.5");

//Each of the Number subclasses that wrap primitive numeric types also provides a
parseXXXX() method
// (for example, parseFloat()) that can be used to convert strings to primitive numbers
float f2 = Float.parseFloat("4.5");
```

Useful: Converting Between Date and Strings

```
//Date to String
Date now = new Date();
System.out.println(now); //Thu Apr 22 00:43:42 MSK 2021
DateFormat df = new SimpleDateFormat("dd-MM-yyyy");
System.out.println(df.format(now)); //22-04-2021

//String to Date
DateFormat df2 = new SimpleDateFormat("dd MM yyyy");
String dateStr = "12 07 2023";
Date date = df2.parse(dateStr);
System.out.println(df.format(date)); //12-07-2023

//TODO try to parse String that has not expected date format
```

Useful link: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html

Useful: String Formatting

```
//Result of 2 + 2 * 2 = 6
System.out.println(String.format("Result of %d + %d * %d = %d", 2, 2, 2, 6));
//Output:  Result of 2 + 2 * 2 = 6
System.out.println(String.format("Result of %d + %d * %d = %d",
        2, 2, 2, "I don't know"));
//Output: IllegalFormatConversionException: d != java.lang.String


//"globojboss@test-dl-
19.corp.globoforce.com:/opt/dataloads/dataload/clientData/client5029/sourceData/";
String client = "client5029";
String dlDestServerUser = "globojboss";
String dlDestServer = "test-dl-19.corp.globoforce.com";
String dlDestDir = "/opt/dataloads/dataload";
String dlClientDataDestDir = "clientData";
System.out.println(String.format("%s@%s:%s/%s/%s/sourceData/",
        dlDestServerUser, dlDestServer, dlDestDir, dlClientDataDestDir, client));
```

Useful: Strings in a switch Statement

```
String status = "on"; //it's recommended to do lowercase or uppercase, think why?)
switch(status) {
    case "on":
        System.out.println("Turn on"); // Will execute this
        break;
    case "off":
        System.out.println("Turn off");
        break;
    default:
        System.out.println("Unknown command");
        break;
}
```

Remember:

- Avoid null in switch statement as you will get NullPointerException
- The switch statement for strings compares the switch-expression with case labels as if the equals() method of the String class has been invoked

StringBuilder/StringBuffer

- StringBuilder objects are like String objects, except that they can be modified.
- Internally, these objects are treated like variable-length arrays that contain a sequence of characters.
- At any point, the length and content of the sequence can be changed through method invocations.

Useful links:

- https://docs.oracle.com/javase/tutorial/java/data/buffers.html
- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/StringBuilder.html

StringBuilder: Creation

Constructor	Description
StringBuilder()	Creates an empty string builder with a capacity of 16 (16 empty elements).
StringBuilder(CharSequence cs)	Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence.
StringBuilder(int initCapacity)	Creates an empty string builder with the specified initial capacity.
StringBuilder(String s)	Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string.

StringBuilder: Length and Capacity

```
StringBuilder sb = new StringBuilder();
sb.append("HELLO WORLD");
```

length() = 11 capacity() = 16

H	E	L	L	O		W	O	R	L	D					
0	1	2	3	4	5	6	7	8	9	**10**	11	12	13	14	**15**

void setLength(int newLength)	Sets the length of the character sequence. If newLength is less than length(), the last characters in the character sequence are truncated. If newLength is greater than length(), null characters are added at the end of the character sequence.
void ensureCapacity(int minCapacity)	Ensures that the capacity is at least equal to the specified minimum.

StringBuilder: Length and Capacity

```
StringBuilder sb = new StringBuilder("Hello");
System.out.println("length=" + sb.length()); // len is assigned 5
System.out.println("capacity=" + sb.capacity()); //capacity is 21 (5 + 16)

sb.setLength(2);
System.out.println(sb); //He
System.out.println("length=" + sb.length()); // len is assigned 2
System.out.println("capacity=" + sb.capacity()); // capacity is assigned 21

sb.append("more-more-more");
System.out.println("length=" + sb.length()); // len is assigned 16
System.out.println("capacity=" + sb.capacity()); // capacity is assigned 21

System.out.println("capacity=" + sb.append("more-more-more-more-more-
more").capacity()); //45
sb.ensureCapacity(50);
System.out.println("capacity=" + sb.capacity()); // 92 ((45*2)+2)
```

StringBuilder: Operations

Method	Description
StringBuilder append(boolean b) .... StringBuilder append(int i) StringBuilder append(Object obj) StringBuilder append(String s)	Appends the argument to this string builder. The data is converted to a string before the append operation takes place.
StringBuilder delete(int start, int end) StringBuilder deleteCharAt(int index)	The first method deletes the subsequence from start to end-1 (inclusive) in the StringBuilder's char sequence. The second method deletes the character located at index.
StringBuilder insert(int offset, boolean b) ... StringBuilder insert(int offset, int i) StringBuilder insert(int offset, Object obj) StringBuilder insert(int offset, String s)	Inserts the second argument into the string builder. The first integer argument indicates the index before which the data is to be inserted. The data is converted to a string before the insert operation takes place.
StringBuilder replace(int start, int end, String s) void setCharAt(int index, char c)	Replaces the specified character(s) in this string builder.
StringBuilder reverse()	Reverses the sequence of characters in this string builder.
String toString()	Returns a string that contains the character sequence in the builder.

StringBuilder: Operations

```
StringBuilder sb = new StringBuilder("hello");
sb.append(" ").append("people"); //append always to the end!
System.out.println(sb); //hello people

sb.insert(6, "good ");
System.out.println(sb); //hello good people

sb.delete(0, 6);
System.out.println(sb); //good people

sb.replace(5, sb.length(), "evening");
System.out.println(sb); //good evening

sb.reverse();
System.out.println(sb); //gnineve doog
```

EXAMPLE WITH PERFORMANCE

```java
public static void main(String[] args) {
    int count = 10000;
    Instant start = Instant.now();
    String s = "0";
    for (int i = 1; i < count; i++) {
        s += String.valueOf(i);
    }
    System.out.println(s);
    Instant end = Instant.now();
    System.out.println("Duration with pure String: " +Duration.between(start, end).toMillis());

    start = Instant.now();
    StringBuilder sb = new StringBuilder("0");
    for (int i = 1; i < count; i++) {
        sb.append(String.valueOf(i));
    }
    System.out.println(sb);
    end = Instant.now();
    System.out.println("Duration with StringBuilder: " +Duration.between(start, end).toMillis());
}
```

PerformanceComparison › main()

PerformanceComparison

```
D:\programms\java\java_1_8\bin\java.exe ...
0123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555
Duration with pure String: 358
0123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555
Duration with StringBuilder: 11

Process finished with exit code 0
```

String Key Points:

- String objects are immutable, and String reference variables are not.
- If you create a new String without assigning it, it will be lost to your program.
- If you redirect a String reference to a new String, the old String can be lost.
- String methods use zero-based indexes, except for the second argument of substring().
- The String class is final—its methods can't be overridden.
- When the JVM finds a String literal, it is added to the String literal pool.

StringBuilder/StringBuffer key points

- The StringBuffer's API is the same as the StringBuilder's API, except that StringBuilder's methods are not synchronized for thread safety.
- StringBuilder methods should run faster than StringBuffer methods.
- All of the following bullets apply to both StringBuffer and StringBuilder:
    - They are mutable—they can change without creating a new object.
    - StringBuilder methods act on the invoking object, and objects can change without an explicit assignment in the statement.
- Remember that chained methods are evaluated from left to right.

Recommended materials

- https://docs.oracle.com/javase/tutorial/java/data/strings.html
- Beginning Java 8 Fundamentals. Language Syntax, Arrays, Data Types, Objects, and Regular Expressions. Author Sharan, Kishori. Chapter 11
- Effective Java, Bloch
- Thinking in Java, Eckel

Next class topics

- Collections

Homework

- https://www.hackerrank.com/challenges/java-strings-introduction/problem
- https://www.hackerrank.com/challenges/java-substring/problem
- https://www.hackerrank.com/challenges/java-string-reverse/problem

Optional

- https://www.hackerrank.com/challenges/java-anagrams/problem

Thanks