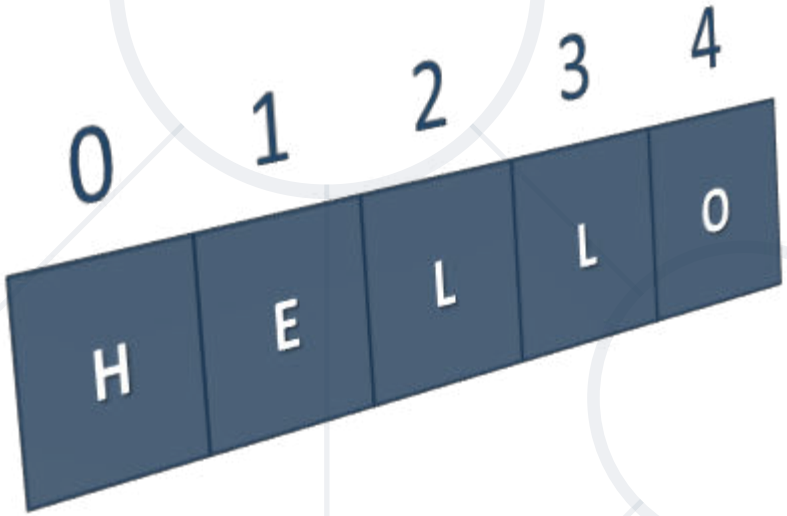
Text Processing and Regular Expressions

Manipulating Text
Using the .NET String Class and using RegEx



SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

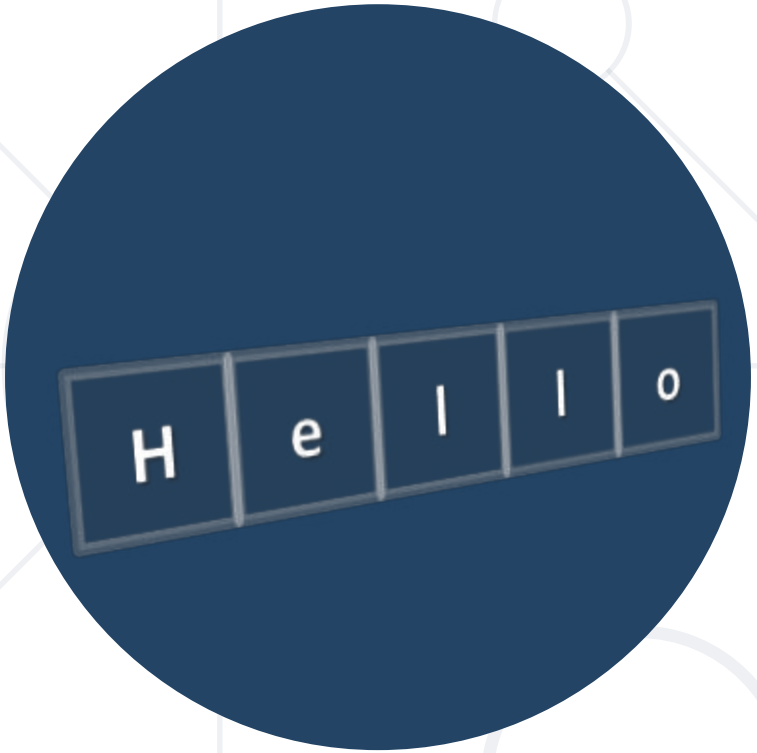
Questions?

SoftUni Foundation

sli.do

#tech-java

Table of Contents



Strings

What is a String?

What is a String?

- Strings are sequences of characters (texts)
- The string data type in Java
  - Declared by the `String`
- Strings are enclosed in quotes:

```
String text = "Hello, Java";
```

Strings Are Immutable, Use Unicode

- Strings are **immutable** (read-only) sequences of characters
- Accessible by index (read-only)

```
String str = "Hello, Java";
char ch = str.charAt(2); // l
```

- Strings use **Unicode** (can use most alphabets, e.g. Arabic)

```
String greeting = "你好"; // (lí-hó) Taiwanese
```

Initializing a String

- Initializing from a string literal:

```
String str = "Hello, Java";
```

- Reading a **string** from the console:

```
String name = sc.nextLine();
System.out.println("Hi, " + name);
```

- Converting a **string** from and to a **char array**:

```
String str = new String(new char[]{'s', 't', 'r'});
char[] charArr = str.toCharArray();
// ['s', 't', 'r']
```

Manipulating Strings

Concatenating

- Use the + or the += operators

```
String text = "Hello" + ", " + "world!";
// "Hello, world!"
```

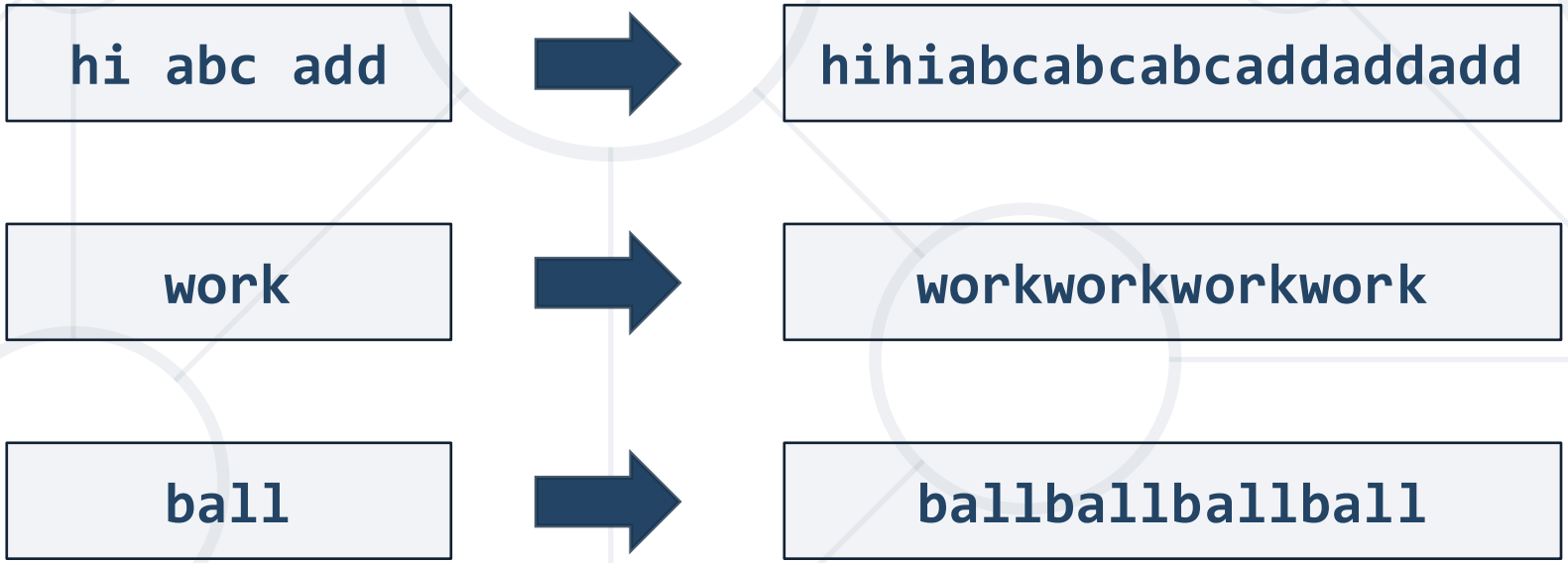
```
String text = "Hello, ";
text += "John"; // "Hello, John"
```

- Use the `concat()` method

```
String greet = "Hello, ";
String name = "John";
String result = greet.concat(name);
System.out.println(result); // "Hello, John"
```

Problem: Repeat Strings

- Read an array from strings
- Repeat each word **n** times, where **n** is the length of the word



Check your solution here: https://judge.softuni.bg/Contests/1330

Solution: Repeat Strings

```
String[] words = sc.nextLine().split(" ");
String result = "";
for (String word : words) {
  int repeatTimes = word.length();
  for (int i = 0; i < repeatTimes; i++)
    result += word;
}
System.out.println(result);
```

Check your solution here: https://judge.softuni.bg/Contests/1330

Substring

- `substring(int startIndex, int endIndex)`

```
String card = "10C";
String power = card.substring(0, 2);
System.out.println(power); // 10
```

- `substring(int startIndex)`

```
String text = "My name is John";
String extractWord = text.substring(11);
System.out.println(extractWord); // John
```

Searching (1)

- `indexOf()` - returns the first match index or -1

```
String fruits = "banana, apple, kiwi, banana, apple";
System.out.println(fruits.indexOf("banana"));    // 0
System.out.println(fruits.indexOf("orange"));    // -1
```

- `lastIndexOf()` - finds the last occurrence

```
String fruits = "banana, apple, kiwi, banana, apple";
System.out.println(fruits.lastIndexOf("banana")); // 21
System.out.println(fruits.lastIndexOf("orange")); // -1
```

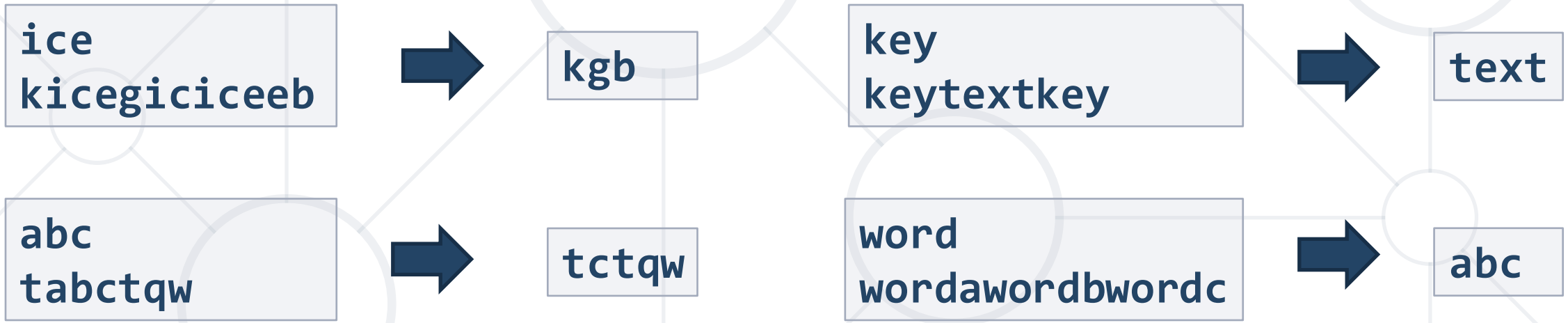
Searching (2)

- `contains()` - check whether one string contains other string

```
String text = "I love fruits.";
System.out.println(text.contains("fruits")); // true
System.out.println(text.contains("banana")); // false
```

Problem: Substring

- You are given a **remove word** and a **text**
- Remove all substrings that are equal to the remove word



Check your solution here: https://judge.softuni.bg/Contests/1330

Solution: Substring

SoftUni Foundation

```
String key = sc.nextLine();
String text = sc.nextLine();

int index = text.indexOf(key);
while (index != -1) {
  text = text.replace(key, "");
  index = text.indexOf(key);
}

System.out.println(text);
```

Check your solution here: https://judge.softuni.bg/Contests/1330

Splitting

- **Split** a string by given **pattern**

```
String text = "Hello, john@softuni.bg, you have been using
john@softuni.bg in your registration";
String[] words = text.split(", ");
// words[]: "Hello", "john@softuni.bg","you have been…"
```

- **Split** by **multiple separators**

```
String text = "Hello, I am John.";
String[] words = text.split("[, .]+");
// "Hello", "I", "am", "John"
```

Replacing

- `replace(match, replacement)` – replaces all occurrences
  - The result is a new **string** (strings are immutable)

```
String text = "Hello, john@softuni.bg, you have been using john@
softuni.bg in your registration.";
String replacedText = text
                    .replace("john@softuni.bg", "john@softuni.com");
System.out.println(replacedText);
// Hello, john@softuni.com, you have been using john@softuni.com
in your registration.
```

Problem: Text Filter

- You are given a text and a string of banned words
  - Replace all banned words in the text with asterisks

```
Linux, Windows
It is not Linux, it is GNU/Linux. Linux is merely the
kernel, while GNU adds the functionality...
```

```
It is not *****, it is GNU/*****. ***** is merely
the kernel, while GNU adds the functionality...
```

Check your solution here: https://judge.softuni.bg/Contests/1330

Solution: Text Filter (1)

```
String[] banWords = sc.nextLine.split(", ");
String text = sc.nextLine();
for (String banWord : banWords) {
  if (text.contains(banWord)) {
    String replacement = repeatStr("*",
banWord.length());
    text = text.replace(banWord, replacement);
  }
}
System.out.println(text);
```

contains(…) checks if string contains another string

replace() a word with a sequence of asterisks of the same length

Check your solution here: https://judge.softuni.bg/Contests/1330

Solution: Text Filter (2)

```
private static String repeatStr(String str, int length) {
  String replacement = "";
  for (int i = 0; i < length; i++) {
    replacement += str;
  }
  return replacement;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1330

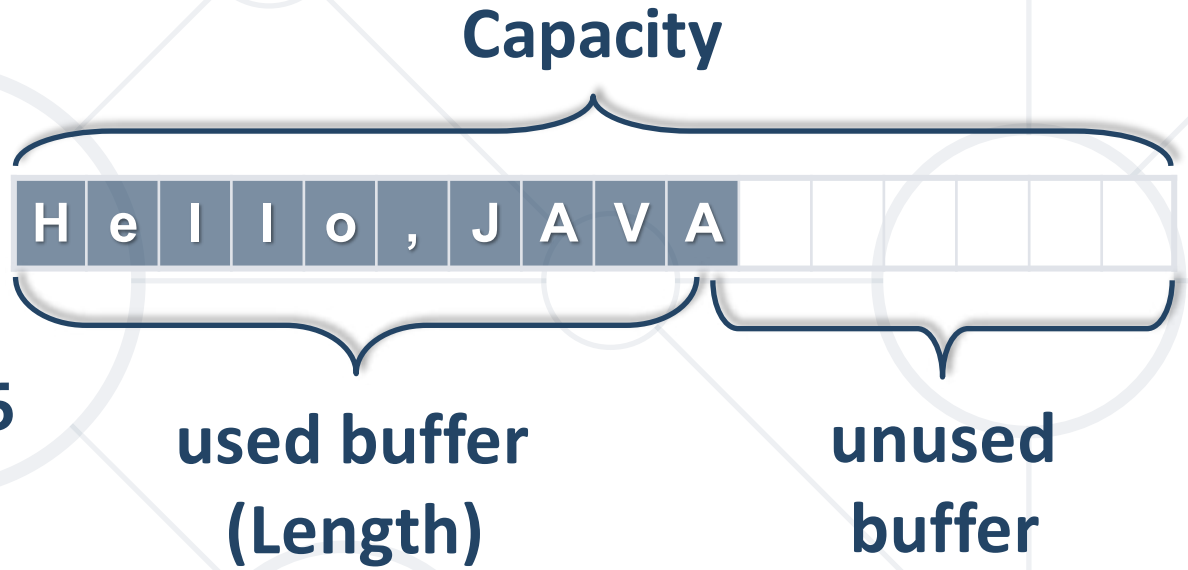
Live Exercises

Building and Modifying Strings

Using the StringBuilder Class

StringBuilder: How It Works?

```
StringBuilder:
    length() = 10
    capacity() = 16
```



- **StringBuilder** keeps a buffer space, allocated in advance
  - Do not allocate memory for most operations → performance

Using StringBuilder Class

SoftUni Foundation

- Use the **StringBuilder** to build / modify strings

```
StringBuilder sb = new StringBuilder();
sb.append("Hello, ");
sb.append("John! ");
sb.append("I sent you an email.");
System.out.println(sb.toString());
// Hello, John! I sent you an email.
```

Concatenation vs StringBuilder (1)

- **Concatenating** strings is a **slow** operation because each iteration **creates** a **new string**

```
System.out.println(new Date());
String text = "";
for (int i = 0; i < 1000000; i++)
   text += "a";
System.out.println(new Date());
```

```
Tue Jul 10 13:57:20 EEST 2018
Tue Jul 10 13:58:07 EEST 2018
```

Concatenation vs StringBuilder (2)

- Using **StringBuilder**

```
System.out.println(new Date());
StringBuilder text = new StringBuilder();
for (int i = 0; i < 1000000; i++)
  text.append("a");
System.out.println(new Date());
```

```
Tue Jul 10 14:51:31 EEST 2018
Tue Jul 10 14:51:31 EEST 2018
```

StringBuilder Methods (1)

SoftUni Foundation

- **append(…)** – appends the string representation of the argument

```
StringBuilder sb = new StringBuilder();
sb.append("Hello Peter, how are you?");
```

- **length(…)** – holds the length of the string in the buffer

```
sb.append("Hello Peter, how are you?");
System.out.println(sb.length()); // 25
```

- **setLength(0)** – removes all characters

StringBuilder Methods (2)

- **char At(int index)** – return char on current index

```
StringBuilder sb = new StringBuilder();
sb.append("Hello Peter, how are you?");
System.out.println(sb.charAt(1)); // e
```

- **insert(int index, String str)** – inserts a string at the specified character position

```
sb.insert(11, " Ivanov");
System.out.println(sb);
// Hello Peter Ivanov, how are you?
```

StringBuilder Methods (3)

SoftUni Foundation

- **replace(int startIndex, int endIndex, String str)**
  - replaces the characters in a substring

```
sb.append("Hello Peter, how are you?");
sb.replace(6, 11, "George");
```

- **toString()**
  - converts the value of this instance to a String

```
String text = sb.toString();
System.out.println(text);
// Hello George, how are you?
```

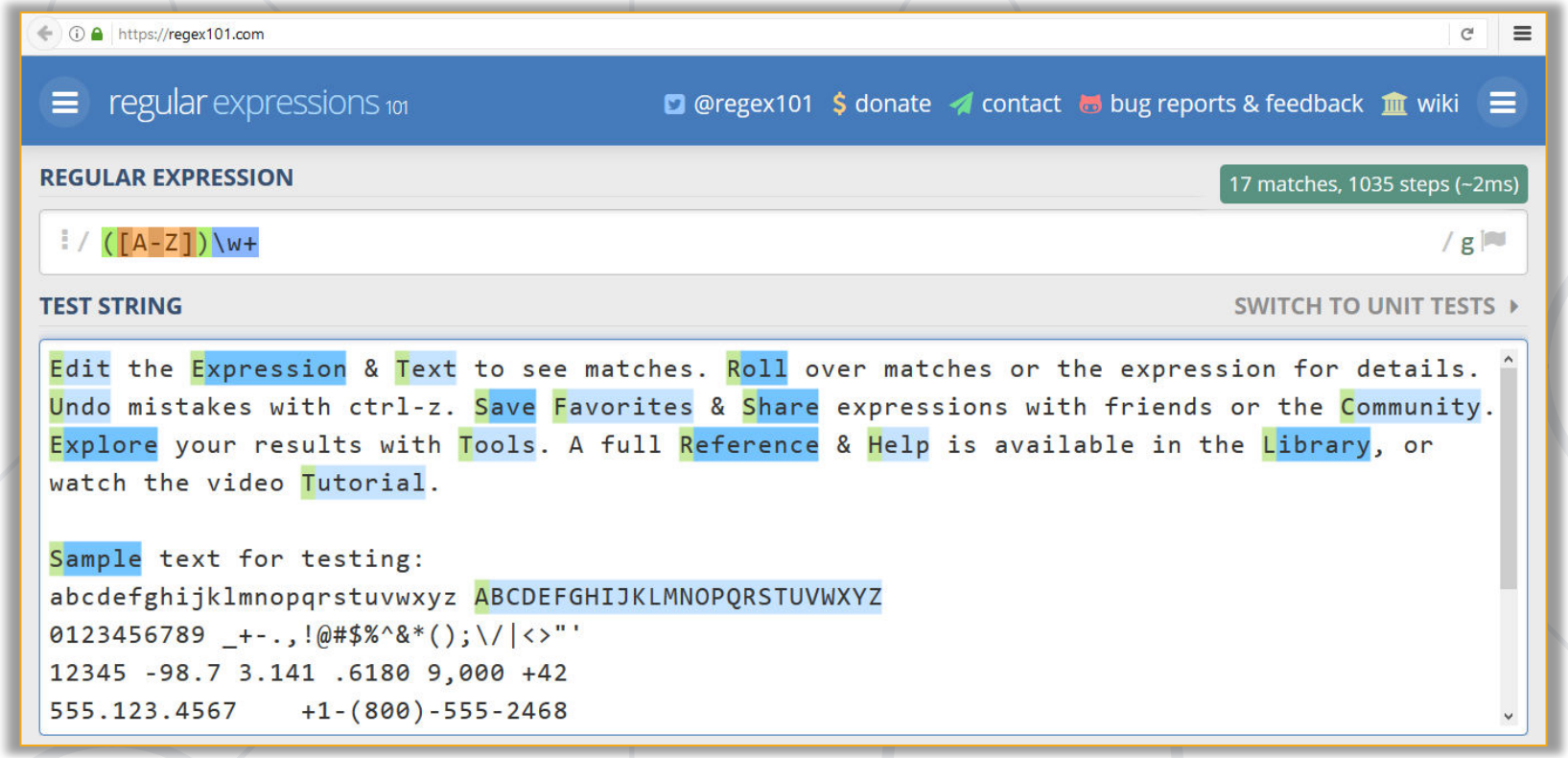
Live Exercises

Regular Expressions

Definition and Classes

What are Regular Expressions?

- Regular expressions (regex)
  - Match text by pattern
- Patterns are defined by special syntax, e.g.
  - `[0-9]+` matches non-empty sequence of digits
  - `[A-Z][a-z]*` matches a capital + small letters
- Play with regex live at: regexr.com, regex101.com



www.regex101.com

Live Demo

Regular Expression Pattern – Example

- Regular expressions (regex) describe a search pattern
- Used to find / extract / replace / split data from text by pattern

```
[A-Z][a-z]+ [A-Z][a-z]+
```

```
John Smith
```

```
Linda Davis
```

```
Contact: Alex Scott
```

Character Classes: Ranges

- `[nvj]` matches any character that is either `n`, `v` or `j`

```
node.js v0.12.2
```

- `[^abc]` – matches any character that is **not** `a`, `b` or `c`

```
Abraham
```

- `[0-9]` – character range: matches any digit from `0` to `9`

```
John is 8 years old.
```

Predefined Classes

SoftUni Foundation

- **\w** – matches any **word character** (a-z, A-Z, 0-9, _)
- **\W** – matches any **non-word character** (the opposite of \w)
- **\s** – matches any **white-space** character
- **\S** – matches any **non-white-space** character (opposite of \s)
- **\d** – matches any **decimal digit** (0-9)
- **\D** – matches any **non-decimal character** (the opposite of \d)

Quantifiers

- `*` – matches the previous element zero or more times

```
\+\d*   ➡   +359885976002 a+b
```

- `+` – matches the previous element one or more times

```
\+\d+   ➡   +359885976002 a+b
```

- `?` – matches the previous element zero or one time

```
\+\d?   ➡   +359885976002 a+b
```

- `{3}` – matches the previous element exactly 3 times

```
\+\d{3}   ➡   +359885976002 a+b
```

Grouping Constructs

- `(subexpression)` – captures the matched subexpression as numbered group

```
\d{2}-(\w{3})-\d{4}
```

```
22-Jan-2015
```

- `(?:subexpression)` – defines a non-capturing group

```
^(?:Hi|hello),\s*(\w+)$
```

```
Hi, Peter
```

- `(?<name>subexpression)` – defines a named capturing group

```
(?<day>\d{2})-(?<month>\w{3})-
(?<year>\d{4})
```

```
22-Jan-2015
```

Problem: Match All Words

- Write a regular expression in www.regex101.com that extracts all word char sequences from given text

```
_ (Underscores) are also word characters!
```

```
_|Underscores|are|also|word|characters
```

Problem: Email Validation

- Write a regular expression that performs simple **email validation**
  - An email consists of: **username @ domain name**
  - **Usernames** are **alphanumeric**
  - **Domain names** consist of **two strings**, separated by a **period**
  - **Domain names** may contain only **English letters**

Valid: `valid123@email.bg`

Invalid: `invalid*name@emai1.bg`

Lookahead

- Positive lookahead
  - Find expression A where expression B follows

```
A(?=B)
```

➡

```
[a-z]+(?=\d+)
```

- Negative lookahead
  - Find expression A where expression B does not follow

```
A(?!B)
```

➡

```
[a-z]+(?!\d+)
```

Lookbehind

- Positive lookbehind
  - Find expression A where expression B precedes

```
(?<=B)A  ➡  (?<=\d)[a-z]+
```

- Negative lookbehind
  - Find expression A where expression B does not precede

```
(?<!B)A  ➡  (?<!\d)[a-z]+
```

Backreferences Match Previous Groups

- `\number` – matches the value of a numbered capture group

```
<(\w+)[^>]*>.*?<\/\1>
```

```
<b>Regular Expressions</b> are cool!
<p>I am a paragraph</p> … some text after
Hello, <div>I am a<code>DIV</code></div>!
<span>Hello, I am Span</span>
<a href="https://softuni.bg/">SoftUni</a>
```

Regular Expressions

Using Built-In Regex Classes

Regex in Java

- Regex in Java library
  - `java.util.regex.Pattern`
  - `java.util.regex.Matcher`

```
Pattern pattern = Pattern.compile("a*b");
Matcher matcher = pattern.matcher("aaaab");

boolean match = matcher.find();
String matchText = matcher.group();
```

Searches for the next match

Gets the matched text

Checking for a Single Match

SoftUni Foundation

- `find()` - Gets the first pattern match

```
String text = "Andy: 123";
String pattern = "([A-Z][a-z]+): (?<number>\\d+)";

Pattern regex = Pattern.compile(pattern);
Matcher matcher = regex.matcher(text);

System.out.println(matcher.find());              // true
System.out.println(matcher.group());             // Andy: 123
System.out.println(matcher.group(0));            // Andy: 123
System.out.println(matcher.group(1));            // Andy
System.out.println(matcher.group(2));            // 123
System.out.println(matcher.group("number")); // 123
```

+ - Matches the element one or more times

Replacing With Regex

SoftUni Foundation

```
String regex = "[A-Za-z]+";
String string = "Hello Java";

Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(string);

String res = matcher.replaceAll("hi");    // hi hi
String res2 = matcher.replaceFirst("hi"); // hi Java
```

Splitting With Regex

- `split(String pattern)` – splits the text by the pattern
  - Returns `String[]`

```
String text = "1   2 3      4";
String pattern = "\\s+";

String[] tokens = text.split(pattern);
```

Matches whitespaces

tokens = {"1", "2", "3", "4"}

Helpful Resources

- https://regex101.com and http://regexr.com – websites to test Regex using different programming languages
- https://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html – a quick reference for Regex from Oracle
- http://regexone.com – interactive tutorials for Regex
- http://www.regular-expressions.info/tutorial.html – a comprehensive tutorial on regular expressions

Live Exercises

Summary

- Strings are **immutable** sequences of Unicode characters
- String processing methods
  - `concat()`, `indexOf()`, `contains()`, `substring()`, `split()`, `replace()`, …
- `StringBuilder` efficiently builds / modifies strings
- Regular expressions describe patterns for searching through text
- Can utilize character classes, groups, quantifiers and more

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

SoftUni Organizational Partners

SoftUni Foundation

Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

SoftUni Foundation

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license