Json in Java

- Json definiert ein lesbares Format, dass Informationen in Objekten, Arrays oder Variablen beinhalten kann.
- Damit eignet sich Json für die einheitliche Gestaltung von Datenformaten
- Json Strukturen lassen sich in fast jeder Programmiersprache mit build-in Funktionen generieren und lesen
- Json Strukturen eignen sich auch hervorragend dazu, um Daten in einem einheitlichen Format in Files abzulegen oder auszulesen.

Gemeinsames Beispiel-Newsletteranmeldung

Schreibe ein Programm, dass es ermöglicht, Personen zu einem Newsletter hinzuzufügen. Dabei muss dein Programm ein Menu anbieten, in dem man Personen hinzufügen und anzeigen kann. Von den Personen müssen der volle Name und die Emailadresse im JSON Format in Files gespeichert und wieder ausgelesen werden können.

Reguläre Ausdrücke (Regular Expressions)

- Ein Regulärer Ausdruck beschreibt ein Pattern, dass überprüft ob ein bestimmter Input einem bestimmten Muster entspricht.
- Eine Liste an möglichen Ausdrücken für reguläre Ausdrücke erhält man hier: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Cheatsheet
- Beispiel für einen regulären Ausdruck, der ein Passwort nach einem bestimmten Pattern untersucht:
  `^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$` = „Minimum 8 Charakter und dabei zumindest ein Buchstabe und eine Zahl, wobei mit einem Buchstaben begonnen werden soll"

Regex Formelsammlung

- **^Hallo** entspricht jedem String der mit "Hallo" beginnt
- **Servus$** entspricht jedem String der mit "Servus" endet
- **^Hallo Servus$** entspricht genau dem String "Hallo Servus"
- **Test** entspricht jedem String der den Text "Test" beinhaltet.
- abc* entspricht einem String mit "ab" gefolgt von 0 oder mehreren "c"
- abc+ entspricht einem String mit "ab" gefolgt von 1 oder mehreren "c"
- abc? entspricht einem String mit "ab" gefolgt von 0 oder 1 "c"

Regex Formelsammlung

- abc**{2}** entspricht einem „ab“ gefolgt von 2 „c“
- abc**{2,}** entspricht einem „ab“ gefolgt von mindestens 2 „c“
- abc**{2,5}** entspricht einem „ab“ gefolgt von 2 bis 5 „c“
- a**(bc)*** entspricht einem „a“ gefolgt von 0 oder mehreren „bc“
- **a(b|c)** entspricht einem „a“ gefolgt von 1 „b“ oder „c“ (mit Capture)
- **a[bc]** entspricht einem „a“ gefolgt von 1 „b“ oder „c“ (ohne Capture)
- Capture bedeutet, dass der Ausdruck später noch einmal verwendet werden kann. (ab*c)\1 entspräche zum Beispiel abc:abc während [ab*c]:\1 nicht möglich wäre (falsche Regular Expression)

Regex Formelsammlung

- \d entspricht einer Zahl mit nur einer Stelle
- \w entspricht jedem alphanumerischen Charakter inklusive Underscore
- \s entspricht einem Whitespace
- . entspricht jedem beliebigen Charakter
- \D entspricht jedem nicht Zahl Charakter
- \$\d entspricht jedem String der mit „$“ beginnt und darauf muss eine einstellige Zahl folgen

Regex Formelsammlung

- a(?:bc)* mit ? deaktiviert man das Capturing
- [abc] entspricht einem string mit einem „a" oder einem „b" oder einem „c" = a|b|c = [a-c]
- [a-fA-F0-9] entspricht einer einstelligen hexadezimal Zahl (case sensitive)
- [0-9]% entspricht einem String, der vor dem „%" Zeichen eine einstellige Zahl von 0 -9 hat.
- [^a-zA-Z] in diesem Fall wird ^ als Negierung verwendet - dieser Ausdruck entspricht Strings ohne einem Buchstaben zwischen a-z und A-Z - also ein String der keine Buchstaben aktzeptiert

Regex Formelsammlung

- **\b**abc**\b** entspricht genau abc
- **\B**abc**\B** entspricht allen Strings die genau diese Zeichenfolge in der mittendrinn enthalten
- d**(?=r)** entspricht einem String der ein „d" enthält, dass von einem „r" gefolgt wird
- d**(?!**r**)** entspricht einem String der ein „d" enthält, dass von einem „r" gefolgt wird
- **(?<=**r**)**d entspricht einem String in dem ein "r" einem „d" vorausgeht
- **(?<!**r**)**d entspricht einem String in dem ein "r" nicht einem „d" vorausgeht

Gemeinsames Beispiel

Aufgabe 1: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens einen Großbuchstaben enthält.

Aufgabe 2: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens einen Kleinbuchstaben enthält.

Aufgabe 3: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens eine Zahl enthält.

Aufgabe 4: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens ein Sonderzeichen (z.B. @, #, $) enthält.

Aufgabe 5: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens einen Vokal (a, e, i, o, u) enthält.

Gemeinsames Beispiel

Aufgabe 6: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens ein Leerzeichen enthält.

Aufgabe 7: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens einen Buchstaben (egal ob groß oder klein) enthält.

Aufgabe 8: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens eine Kombination aus Buchstaben und Zahlen enthält.

Aufgabe 9: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens ein Wort enthält, das mit einem Vokal beginnt.

Aufgabe 10: Erstelle einen regulären Ausdruck, der sicherstellt, dass ein Text mindestens ein Wort enthält, das mit einem Konsonanten beginnt.