Java Intro

1 Aufgaben Quick Start & Grundlagen

Aufgabe 1: Java-Installation prüfen

Öffnen Sie ein Kommandozeilenfenster und prüfen Sie die installierte Java-Version durch Aufruf von

```
java –version
```

Nun wollen wir prüfen, ob auch der interaktive Java-Kommandozeileninterpreter namens JShell einsatzfähig ist. Geben Sie dazu

```
jshell
```

ein.

```
|  Welcome to JShell -- Version 17
|  For an introduction type: /help intro

jshell>
```

Bei Bedarf verlassen Sie die JShell durch das Kommando `/exit`:

```
jshell> /exit
|  Goodbye
```

Aufgabe 2: IDE-Installation prüfen

Starten Sie Eclipse und legen Sie ein neues Java-Projekt an, wie es in „Einfach Java“ ab S.12 beschrieben ist.

Aufgabe 3: Mathematische Berechnungen

Nehmen wir an, wir hätten eine Spedition. Wir bekommen einen Großauftrag und müssen 2.222 Bücherkisten ausliefern. In unseren Lkw passen pro Fahrt jedoch nur 75 Kisten. Berechnen Sie, wie oft wir fahren müssen und wie viele Kisten in der letzten Fahrt transportiert werden. Verwenden Sie sprechende Variablennamen. **Denken Sie an Wiederverwendung. Wie kann man das hier erreichen?**

Aufgabe 4: Schleifen mit variabler Schrittweite

Geben Sie 10 mal Ihren Namen aus. Zudem soll der Wert jedes Schleifendurchlauf auf der Konsole ausgegeben werden, etwa wie folgt (gekürzt):

```
Michael: 0
Michael: 1
Michael: 2
Michael: 3
...
```

Aufgabe 5: Schleifen mit variabler Schrittweite

Nutzen Sie eine Schleife, die beim Wert 0 und mit einer Schrittweite von 1 startet. Bei jedem Schleifendurchlauf soll der Wert um die Schrittweite erhöht werden und die Schrittweite wird jeweils um zwei erhöht. Geben Sie die beiden Werte aus, solange die Schleifenvariable kleiner als 70 ist.

```
Wert: 0 / Schrittweite: 1
Wert: 1 / Schrittweite: 3
Wert: 4 / Schrittweite: 5
Wert: 9 / Schrittweite: 7
Wert: 16 / Schrittweite: 9
Wert: 25 / Schrittweite: 11
...
```

Aufgabe 6: Teiler

Schreiben Sie eine Methode `void find_proper_divisors(num)`, die alle echten Teiler einer natürlichen Zahl berechnet, also diejenigen Zahlen ohne die Zahl selbst und diese auf der Konsole ausgibt. Für die 12 würden als echte Teiler die folgenden Werte 1, 2, 3, 4 und 6 ermittelt.

Aufgabe 7: Verschachtelte Schleifen – Variante 1

Schreiben Sie eine Methode `void printNumberTriangle(int row)`, die eine mehrzeilige Ausgabe bis zur übergebenen maximalen Zeilenanzahl wie folgt erzeugt:

```
1
12
123
1234
```

Als Kür soll eine Funktion geschrieben werden, deren Ausgabe wie folgt aussieht:

```
1
2 3
4 5 6
7 8 9 10
```

Aufgabe 8: Verschachtelte Schleifen – Variante 2

Implementieren Sie eine Methode, um Buchstaben in folgendem Muster auszugeben:

```
A
BB
CCC
DDDD
```

Tipp: Einzelne Zeichen kann man mit `char` modellieren und darauf eine Addition ausführen.

```
jshell> char a = 'A'
a ==> 'A'

jshell> char b = (char)(a + 1)
b ==> 'B'

jshell> char result = (char)(a + 7)
result ==> 'H'
```

Aufgabe 9: Berechnen Sie die Fakultät per Rekursion

Die Fakultät ist die Multiplikation aller Zahlen bis zu einem Grenzwert, für den Wert 5 gilt folgendes:

$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

Dies lässt sich wie folgt verallgemeinern:

$$n! = n * (n-1) * (n-2) * ... * 2 * 1$$

Basierend darauf ergibt sich die rekursive Definition:

$$n! = \begin{cases} 1, & n = 0,\ n = 1 \\ n \cdot (n-1)!, & \forall n > 1 \end{cases}$$

Dabei steht das umgedrehte »A« (∀) für »für alle« .

Aufgabe 10: Berechnen Sie die Fibonacci-Zahlen per Rekursion

Auch die ***Fibonacci-Zahlen*** lassen sich hervorragend rekursiv definieren, wobei die Formel schon ein klein wenig komplexer ist:

$$fib(n) = \begin{cases} 1, & n = 1 \\ 1, & n = 2 \\ fib(n-1) + fib(n-2), & \forall n > 2 \end{cases}$$

Es ergibt sich für die ersten n folgender Werteverlauf:

n	1	2	3	4	5	6	7	8
fib(n)	1	1	2	3	5	8	13	21

Implementieren Sie eine Methode `int fib(int n)`.

Was könnte an der rekursiven Umsetzung nicht optimal sein. Probieren Sie einmal einen Aufruf für den Wert 35, 42 und danach für 47.

2 Aufgaben zu Strings

Aufgabe 1: Länge, Zeichen und Enthaltensein

In dieser Aufgabe geht es darum, grundlegende Funktionalität der Strings anzuwenden. Sie sollen die Länge eines Texts abfragen, dann ein Zeichen an einer beliebigen Position, etwa der 13., ermitteln und schließlich prüfen, ob ein gewünschtes Wort im String enthalten ist.

Aufgabe 2: Zeichen wiederholen

Bei dieser Aufgabe geht es darum, die Buchstaben eines Worts gemäß ihrer Position zu wiederholen, aus ABC wird dann ABBCCC. Schreiben Sie dazu eine Methode `String repeatChars(String input)`.

Aufgabe 3: Vokale raten

Bei dieser Aufgabe werden einige etwas ältere Leser sich vielleicht an die Sendung »Glücksrad« erinnern, bei der es genau um das Erraten von Wörtern, Sätzen oder Redewendungen ging, in denen Vokale fehlten.

a) Als Erstes sollen in einem gegebenen String mithilfe der selbst geschriebenen Methode `String removeVowels(String input)` alle Vokale entfernt werden.

b) Als Zweites soll eine Methode `String replaceVowels(String input)` implementiert werden, die einen Vokal durch ein '_' ersetzt, damit wir für ein kleines Ratespiel einen Hinweis auf einen entfernten Vokal bekommen.

```
removeVowel("Es gibt viel zu entdecken!") => s gbt vl z ntdckn!
replaceVowel("Es gibt viel zu entdecken!") => _s g_bt v__l z_ _ntd_ck_n!
```

Aufgabe 4: Reverse String

Schreiben Sie eine rekursive Methode `String reverse_string(String input)`, die die Buchstaben des übergebenen Eingabetexts umkehrt.

Eingabe	Resultat
"A"	"A"
"ABC"	"CBA"
"abcdefghi"	"ihgfedcba"

Aufgabe 5: Palindrom-Prüfung

Schreiben Sie eine Methode `boolean isPalindrome(String input)`, die überprüft, ob ein gegebener String unabhängig von Groß- und Kleinschreibung ein Palindrom ist. Erinnern wir uns: Ein Palindrom ist ein Wort, das sich von vorne und von hinten gleich liest.

Eingabe	Resultat
"Otto"	True
"ABCBX"	False
"ABCXcba"	True

Aufgabe 6: Lineal

In dieser Aufgabe wollen wir ein Lineal im englischen Stil nachahmen. Dabei wird ein Bereich von einem Inch in 1/2 und 1/4 sowie 1/8 unterteilt. Dabei nimmt die Länge der Striche jeweils um eins ab.

```
---- 0
-
--
-
---
-
--
-
---- 1
```

Aufgabe 7: Summe der Ziffern

In dieser Aufgabe geht es darum die Ziffer eines Texts zu summieren. Für die folgende Eingabe

```
"a7b5c2d4ef12stop"
```

soll der Wert 21 ermittelt und wie folgt ausgegeben werden:

```
sum of digits: 21
```

Aufgabe 8: Suchen und extrahieren

In dieser Aufgabe wollen wir Texte finden bzw. extrahieren. An welcher Position findet sich das erste `a` und das letzte? Mit welchem Aufruf kann man `car` ausschneiden? Gegeben sein folgender Text:

```
"Was it a car or a cat I saw?"
```

Aufgabe 9: String Merge

Schreiben Sie eine Methode `String stringMerge(String input1, String input2)`, die die beiden Texte Buchstabe für Buchstabe (»wie ein Reißverschluss«) miteinander verbindet. Wenn der eine Text ACE lautet und der andere BDFGH, dann soll ABCDEFGH als Ergebnis geliefert werden. Es wird also immer abwechselnd ein Buchstabe aus dem ersten und dann einer aus dem zweiten Text genommen. Ist ein Text vollständig verarbeitet, werden alle Buchstaben aus dem verbliebenen anderen Text übernommen.

Aufgabe 10: Lerne StringBuilder kennen

Die Seite https://www.dotnetperls.com/stringbuilder-java gibt einen guten Überblick über die Möglichkeiten mit `StringBuilder`. Vollziehe ein paar der Beispiele nach und wandle diese ab.

Lösche beispielsweise aus dem Text

```
“Oracle Code One “
```

Das Wort Code und danach ersetze Oracle mit Java. Hänge noch die Zahl 2001 sowie den Text „`was great`“. Was ist das Ergebnis?

Aufgabe 11: Print Tower

Schreiben Sie eine Methode `void printTower(int n)`, die einen Turm aus n übereinander gestapelten Slices als ASCII-Grafik darstellt, symbolisiert durch das Zeichen #, und eine untere Begrenzungslinie zeichnet, etwa wie folgt:

```
    |
   #|#
  ##|##
 ###|###
---------
```

Aufgabe 12: RegEx First Steps

Schreiben Sie einen regulären Ausdrucks, der prüft, ob die Zeichenfolge nur Ausbuchstaben und Ziffern besteht. Somit sollte der erste Test erfolgreich sein, der zweite nicht.

```
System.out.println("ABCDef1234".matches(regex));
System.out.println("xy!".matches(regex));
```

Aufgabe 13: RegEx Date Checker

Überprüfen Sie einen Datumsstring mit Tag. Monatsname und Jahresangabe mithilfe eines regulären Ausdrucks. Folgende drei sollten alle `true` ergeben:

```
System.out.println("7. Februar 1971".matches(regex));
System.out.println("23. November 2020".matches(regex));
System.out.println("14. Sept 21".matches(regex));
```

Aufgabe 14: RegEx Date Checker

Überprüfen Sie einen String aus Buchstaben, Spezialzeichen und Klammern mithilfe eines regulären Ausdrucks. Folgende drei sollten alle `true` ergeben:

```
System.out.println("abc123xyz".matches(regex));
System.out.println("def myfunc(value)".matches(regex));
System.out.println("var g = 123;".matches(regex));
```

Aufgabe 15 – Text Blocks

Vereinfachen Sie folgenden Sourcecode mit einem herkömmlichen String, der über mehrere Zeilen geht und nutzen Sie die neu eingeführte Syntax.

```
String multiLineStringOld = "THIS IS\n" +
            "A MULTI\n" +
            "LINE STRING\n" +
            "WITH A BACKSLASH \\\n";

String java14FeatureObjOld = ""
                + "{\n"
                + "    version: \"Java13\",\n"
                + "    feature: \"text blocks\",\n"
                + "    attention: \"preview!\"\n"
                + "}\n";
```

Aufgabe 16 – Text Blocks mit Platzhaltern

Vereinfachem Sie folgenden Sourcecode mit einem herkömmlichen String, der über mehrere Zeilen geht und nutzen Sie die neu eingeführte Syntax.

```
String multiLineStringWithPlaceHoldersOld =
       String.format("HELLO \"%s\"!\n" +
                     "  HAVE %s\n" +
                     "  NICE \"%s\"!",
                     new Object[]{"WORLD", "A", "DAY"});

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

Produzieren Sie folgende Ausgaben mit der neuen Syntax:

```
HELLO "WORLD"!
  HAVE A
  NICE "DAY"!
```