PROGRAMMIERUNG

ÜBUNGSSAMMLUNG

1. INHALTSVERZEICHNIS

HINWEISE

- Diese Materialsammlung ist ein didaktisches Hilfsmittel zur Lehrveranstaltung und nicht für ein Selbststudium der Inhalte konzipiert. Ergänzungen und Mitschriften aus den Vorlesungen sowie ggf. Bearbeitung der Literatur des jeweiligen Moduls sowie der im Anhang angegeben Literaturliste ist unerlässlich für einen erfolgreichen Abschluss.
- Sämtliche Inhalte sind urheberrechtlich geschützt (mit freundlichem Danke an Prof. Dr. Jörg Helbach). Sie dürfen nur innerhalb des begrenzten Teilnehmerkreises der Seminarteilnehmer und für private Zwecke genutzt werden. Außerhalb dieses Teilnehmerkreises dürfen die Inhalte ohne vorherige schriftliche Genehmigung des Rechteinhabers oder gesetzliche Erlaubnis weder ganz noch auszugsweise kopiert, verändert, vervielfältigt oder veröffentlicht werden.
- Bei diesem Dokument handelt es sich um eine vorläufige Fassung. Aus diesem Grund können sich jederzeit Änderungen und Ergänzungen ergeben.

I. ÜBUNGEN

1. ÜBUNGSBLATT

1.1 PROGRAMMABLAUFPLAN EC-KARTE

Erstellen Sie mit dem PAP-Designer einen Programmablaufplan für die Bargeldabhebung mit ec-Karte an einem Geldautomaten. Nach dem Einstecken der Karte soll der Nutzer zuerst gefragt werden, ob er eine Auszahlung wünscht oder ob lediglich der Kontostand angezeigt werden soll.

Hinweis: Bedenken Sie, dass die ec-Karte gedeckt sein muss, damit der Geldautomat die gewünschte Summe auszahlt.

1.2 PROGRAMMABLAUFPLAN HERON-VERFAHREN

Archimedes (um 287 - 212 v. Chr.) hat für die Berechnung der Quadratwurzel einer Zahl a die folgende Iterationsformel angegeben:

$$x_{neu} = \frac{1}{2}(x_{alt} + \frac{a}{x_{alt}})$$

Die Differenz der Werte x_{neu} und x_{alt} ergibt die Genauigkeit der Berechnung, d. h. die Formel wird solange wiederholt bis die Differenz der beiden Werte x_{neu} und x_{alt} unter eine Schwelle ε fällt. Ein Pseudocode, der diesen Algorithmus beschreibt kann in etwa so aussehen:

```
Lese Zahl a ein
Setze x_neu auf den Wert a
Wiederhole
  x_alt = x_neu
  x_neu nach Archimedes Formel berechnen
Solange der Betrag der Differenz zwischen x_alt und
x_neu groesser als Epsilon ist
```

Erstellen Sie einen Programmablaufplan, bei dem der Nutzer die Zahl a sowie eine Schwelle ε eingibt und der Algorithmus dann die Annäherung an $\sqrt{a}$ berechnet.

1.3 VORBEREITUNG JAVA

Installieren Sie auf Ihrem Laptop die folgenden Softwaretools:

a) das aktuelle OpenJDK (aktuell Version 17)
   https://jdk.java.net/17/
b) die Entwicklungsumgebung Eclipse IDE for Java Developers in der aktuellen Version
   https://www.eclipse.org/downloads/packages

2. ÜBUNGSBLATT

2.1 SCHALTJAHRE

Entwerfen Sie einen Algorithmus zur Überprüfung, ob es sich bei einem gegebenen Jahr um ein Schaltjahr handelt. Zur Bestimmung, ob es sich um ein Schaltjahr handelt, werden die folgenden drei Regeln herangezogen:

(1) Die durch 4 teilbaren Jahreszahlen sind Schaltjahre, außer wenn Regel 2 oder 3 gilt.
(2) Säkularjahre, also Jahre, an denen ein Jahrhundert abschließt (1800, 1900, 2100, 2200, ...), sind keine Schaltjahre - außer wenn Regel 3 gilt.
(3) Die durch 400 teilbaren Säkularjahre sind Schaltjahre.

2.2 BEZEICHNER

Welche der folgenden Bezeichner sind in Java-Programmen gültig und welche nicht? Begründen Sie.

- _Alles_ist_gut
- 42istDieAntwort
- Bóólean
- Behörde23c
- null
- REZE$$ION
- Übersicht der Artikel
- KostenIn€

2.3 LITERALE

Lesen Sie den folgenden Quellcode:

```
class Application {
       public static void main (String[] args) {
           System.out.print ("Was ist hier ...");
           System.out.println ("ein Literal?");
           System.out.println (17 + 4.23);
       }
}
```

a) Markieren Sie die Literale im Java-Quelltext.
b) Welche Bedeutung haben die fett gedruckten Wörter?

2.4 ZINSSATZ

Erstellen Sie einen eigenen Workspace für die Programmierübungen. Legen Sie dort ein Java-Projekt mit der Bezeichnung Uebung_2_5 an und erstellen Sie darin eine Klasse mit der Bezeichnung Aufgabe 5.

Schreiben Sie ein Programm, das für ein Kapital, einen Zinssatz und eine Laufzeit (in Jahren als Ganzzahl) berechnet, wie viel Geld sich nach Ablauf der Laufzeit angesammelt hat (Zinseszinsen werden nicht berücksichtigt).

Die Werte für das Kapital, den Zinssatz sowie die Laufzeit sollen bei Programmstart beim Benutzer abgefragt werden.

Nutzen Sie den folgenden Programmcode zum Einlesen von Werten über die Konsole:

```
int i = new java.util.Scanner(System.in).nextInt();
double d = new java.util.Scanner(System.in).nextDouble();
```

3. ÜBUNGSBLATT

3.1 UMWANDELN VON SCHLEIFEN

Wandeln Sie die folgende for-Schleife in eine while-Schleife um.

```
int n, s=1;
for(n=2; n<=5; System.out.print((s*=n++)+" "));
```

3.2 AUSGABE MIT SCHLEIFEN

a) Schreiben Sie ein Programm mit einer einfachen Schleife, das nacheinander die folgenden Zahlen ausgibt: 50; 47.5; 45; 42.5; 40; 37.5; 35; 32.5; 30; 27.5; 25
b) Erweitern Sie Ihr Programm, indem Sie zusätzlich am Ende noch Summe und Mittelwert der Zahlen ausgeben.
c) Erweitern Sie Ihr Programm derart, dass der Nutzer zu Beginn den Startwert, den Endwert und Schrittweite eingeben kann. Anschließend soll mittels Schleife Summe und Mittelwert berechnet werden.

3.3 HALBIERUNG

Schreiben Sie ein Programm, bei dem der Nutzer eine beliebige Zahl zwischen 1 und 100 eingeben kann. Die eingegebene Zahl soll schrittweise halbiert (und ausgegeben) werden, solange bis der halbierte Wert kleiner als 0,001 ist. Bei Programmende soll der Nutzer gefragt werden, ob er eine weitere Zahl eingeben möchte.

3.4 TEILBARKEIT

Schreiben Sie ein Programm, das die Zahlen von 1 bis 100 zeilenweise auf der Konsole ausgibt. Bei jeder Zahl, die durch 3 teilbar ist, soll in derselben Zeile der Zusatz “ist durch 3 teilbar” hinzugefügt werden.

3.5 TANNENBAUM

Schreiben Sie ein Programm, das auf der Konsole mittels verschachtelter Schleifen einen Tannenbaum ausgibt. Die Ausgabe soll in etwa so aussehen:

```
*
**
***
****
*****
```

ändern Sie das Programm so ab, dass die Ausgabe wie folgt aussieht:

```
  *
```

```
 ***
*****
*******
```

4. ÜBUNGSBLATT

4.1 REKURSION

Die Funktion `isPrime(p)` sei wie folgt rekursiv definiert:

```
isPrime(p) := isPrime(p, p – 1)
isPrime(p, 1) := true
isPrime(p,z) := false, falls p durch z teilbar ist
isPrime (p , z ) := isPrime (p , z – 1), falls p
nicht durch z teilbar ist
```

Implementieren Sie eine entsprechende rekursive Methode und schreiben Sie wie üblich ein Hauptprogramm, bei dem der Nutzer Zahlen mehrfach eingeben kann.

4.2 SPIEL

Ändern Sie die Übungsaufgabe PointsAndDoors aus der Vorlesung (das Finden der Tür) wie folgt ab:

- Fügen Sie ein Punkt-Objekt Geld hinzu, welches der Spieler aufsammeln muss, bevor er die Tür erreichen kann.
- Fügen Sie einen Gegenspieler hinzu, der zu Beginn des Spiels zufällig positioniert wird und der sich horizontal, vertikal und diagonal bewegen kann
- Der Spieler soll sich nur horizontal und vertikal bewegen können
- Der Spieler soll mit einer Eingabe von u (up), d (down), l (left) und r (right) bewegt werden
- Fügen Sie eine Ausgabe hinzu, bei der bei jedem Schritt die gesamte Matrix aus- gegeben wird. Der Spieler soll mit P, der Gegner mit G, das Geld mit $, und die Tür mit D markiert werden. Alle leeren Felder werden mit einem x dargestellt. Erstellen Sie für die Ausgabe eine eigene Methode.

5. ÜBUNGSBLATT

5.1 ARRAY

Schreiben Sie eine Methode, mittels der ein Nutzer drei Ganzzahlen eingeben kann, die dann in einem Array gespeichert werden. Die Methode soll dieses Array zurückgeben. Im Hauptprogramm sollen dann die Werte für zwei solcher Arrays eingelesen werden und anschließend das Skalarprodukt der beiden Arrays ermittelt werden.

5.2 2D-ARRAY

Erstellen Sie ein zweidimensionales Array der Größe 10x10 Felder. In drei zufälligen Feldern dieser Matrix speichern Sie den Wert 1, in den

restlichen den Wert 0 ab. Schreiben Sie dann eine Methode, die das erzeugte Array auf dem Bildschirm ausgibt, so dass es in etwa so aussieht:

```
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|0|0|1|
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|1|0|0|
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|0|1|0|
|0|0|0|0|0|0|0|0|0|0|
|0|0|0|0|0|0|0|0|0|0|
```

5.3 PALINDROM

Erstellen Sie ein Programm, mit dem der Nutzer prüfen kann, ob es sich bei einer eingegebenen Zeichenkette um ein Palindrom handelt.

5.4 BUBBLESORT

Entwickeln Sie eine Methode, mit der ein int-Array mit dem BubbleSort-Algorithmus sortiert wird.

5.5 CÄSAR-VERSCHLÜSSELUNG

Die Cäsar-Verschlüsselung ist ein Verschiebechiffre, d.h. Wörter werden verschlüsselt, indem die einzelnen Buchstaben um eine bestimmte Anzahl an Zeichen verschoben wer- den. Die Anzahl der Zeichen, die man bei der Verschiebung verwendet, nennt man den Schlüssel k.

Mit dem Schlüssel k = 3 ergibt sich beispielsweise:

```
Klartext:   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Geheimtext: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

So ergibt das Klartextwort HALLO nach der Verschlüsselung den Geheimtext KDOOR. Die Verschlüsselung ergibt sich also durch Verschiebung der Buchstaben um k Zeichen nach rechts. Beim Entschlüsseln werden die Zeichen entsprechend um k Stellen nach links geschoben.

a) Schreiben Sie eine Methode String encrypt(String text, int k), mit der sich Zeichenketten verschlüsseln lassen. Rückgabewert ist die verschlüsselte Zeichenkette.
b) Schreiben Sie eine Methode String decrypt(String text, int k), mit der sich Zeichenketten entschlüsseln lassen. Rückgabewert ist die entschlüsselte Zeichenkette.
c) Schreiben Sie ein Programm, beidem der Nutzer eine Zeichenkette und einen Schlüssel eingeben kann. Das Programm liefert dann die verschlüsselte Zeichenkette.

5.6 VARIABLE PARAMETERANGABE

Erstellen Sie vier Methoden, welche über eine variable Anzahl an Parametern vom Typ double aufgerufen werden kann und

- die Summe,
- das Produkt,
- den Mittelwert und
- die Varianz.

berechnen. Verwenden Sie hierzu die erweiterte For-Schleife. Versuchen Sie, den Quelltext möglichst kompakt zu halten.

5.7 STRING ARRAY

Erzeugen Sie ein 10x10-Feld für String-Objekte. Laufen Sie mit einem geeigneten Schleifenkonstrukt über das gesamte Feld. Weisen Sie jedem Wert in diesem Feld eine Zeichenkette „Spalte:Zeile“. zu. Geben Sie anschließend nur die Diagonalelemente des Feldes aus.

5.8 PALINDROM RELOADED

Betrachtet Sie die Methoden der StringBuilder-Klasse:

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

Implementieren Sie mit möglichst geringem Aufwand eine Palindromprüfung unter Verwendung der StringBuilder-Klasse.

5.9 QUERSUMME

Betrachten Sie die Dokumentation der Klasse Charakter:

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Character.html

Schreiben Sie eine Methode int digitSum(String input), die die Quersumme einer Zahl, die als Zeichenkette übergeben wird, berechnet. Alle Zeichen, die keine Ziffern sind, sollen dabei übersprungen werden.

```
digitSum(“123”); // gibt 6 zurück
digitSum(“abc12”); // gibt 3 zurück
```

Prüfen Sie, wie die Methode erweitert/angepasst werden kann, damit auch für den primitiven Datentyp long die Quersumme berechnet wird.

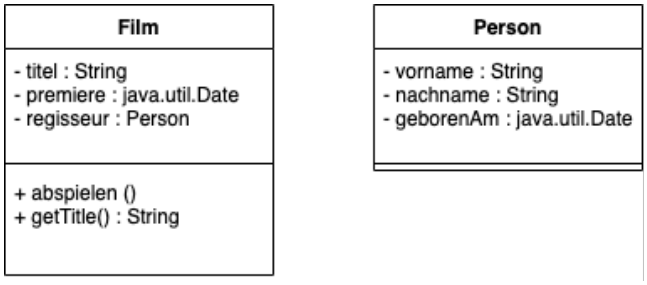
6. ÜBUNGSBLATT

6.1 AUFZÄHLUNGSTYPEN

Erstellen Sie einen typsicheren Aufzählungstypen Colors mit den Elementen ROT, GRÜN und BLAU.

6.2 KLASSENDIAGRAMME

Setzen Sie das folgende Klassendiagramm in Java-Quellcode um.



6.3 ERSTE KLASSE

Entwickeln Sie eine Klasse `NamedPoint`, welche ermöglicht, einen Punkt mit seinen x (int) und y (int) Koordinaten zu speichern und zusätzlich einen Namen (String) zu setzen. Die Attribute sollen von außen nicht sichtbar sein; definieren Sie entsprechende Getter und Setter Methoden. Entwickeln Sie sinnvoll die öffentlichen Methoden

```
void subtract(NamedPoint p)
void add(NamedPoint p)
String getStatus()
```

um diese Operationen auf die Punkte ausführen zu können. Die `getStatus()` Methode soll eine Ausgabe „`<name>: (<X>|<Y>)`“ (ohne < >) zurückgeben.

Definieren Sie ebenfalls eine Methode,

```
Point getPoint()
```

welche ein Objekt der Klasse (java.awt.)Point mit den Werten des NamedPoint-Objektes zurückliefert.

Definieren Sie einen geeigneten Konstruktor, um Koordinaten und Namen bei Erstellung zu setzen. Definieren Sie einen zweiten Konstruktur, der nur die Koordinaten aufnimmt; in diesem Fall soll als Name standardmäßig nur „Punkt“ gesetzt sein.

Erzeugen Sie in einer main-Methode einige Objekte der Klasse, führen Sie die `add()` und `subtract()` Operationen aus und prüfen Sie das Ergebnis mit der `getStatus()` Methode.

6.4 BINÄRBAUM

Entwickeln Sie eine Klasse Node, mit der Sie einen binären Baum in Java implementieren können, der Zeichenketten abspeichern kann. Gehen Sie wie folgt vor:

a) Überlegen Sie sich zunächst, welche Objektvariablen Sie umsetzen müssen. Der Inhalt eines Knotens soll in der Objektvariable content vom Typ String gespeichert werden.
b) Schreiben Sie einen Konstruktor, mit dem ein neuer Knoten erzeugt werden kann. Als Parameter wird dabei der Inhaltsstring mit übergeben.
c) Entwickeln Sie eine Getter-Methode für die Objektvariable content.

d) Schreiben Sie eine Methode public Node addRecursive (Node current, String cn), mit der Sie dem Baum weitere Knoten hinzufügen können.
e) Schreiben Sie eine Methode public boolean contains (String cn), mit der Sie prüfen können, ob ein Baum eine Zeichenkette enthält.
f) Erweitern Sie die Klasse Node um eine Methode public Node deleteRecursive(Node current, String cn), mit der ein Knoten aus dem Baum gelöscht werden kann.
g) Schreiben Sie eine Methode void printTree(), mit der Sie den Baum rekursiv ausgeben können. Durchlaufen Sie den Baum dabei inOrder.
h) Schreiben Sie jeweils einige Testzeilen in das Hauptprogramm, um die einzelnen Auf- gabenteile mit einem Baum Ihrer Wahl auszuprobieren.

7. ÜBUNGSBLATT

7.1 VERERBUNG 1

Erstellen Sie eine neue Version der Klasse `NamedPoint` (vgl. Übung 6), welche jedoch in diesem Fall die Methoden der Klasse (java.awt.)Point vererbt bekommt und ermöglicht, jedem Punkt einen Namen zu geben.

Die Klasse NamedPoint soll alle Konstruktoren implementieren, die auch für die Klasse (java.awt.)Point zur Verfügung stehen.

7.2 VERERBUNG 2

Erstellen Sie eine Klasse Figur mit den Methoden:

```
void zeichne()
double getFlaeche()
```

Die Klassen Kreis und Rechteck sind von Figur abgeleitet und implementieren die beiden Methoden. Schreiben Sie ein Testprogramm, das die beiden Methoden für verschiedene Figuren testet.

Hinweis: Die Methode zeichne soll auf der Konsole nur die Eigenschaften des Objekts (also Breite und Höhe bzw. Radius) und den Flächeninhalt ausgeben.

7.3 VERERBUNG 3

Ein Obstlager kann verschiedene Obstsorten (Apfel, Birne, Orange) aufnehmen. Die Klasse Obst soll die folgenden Methoden enthalten:

```
String getName()
String getFarbe()
```

Die Klassen Apfel, Birne und Orange sind von Obst abgeleitet. Die Klasse Obstlager enthält ein Array des Typen Obst. Die Methode print() dieser Klasse soll für alle Obstsorten im Lager die Methoden getName() und getFarbe() aufrufen und das Ergebnis ausgeben.

7.4 ABSTRAKTE KLASSEN

Zeigen Sie wie Aufgabe 7.2 und 7.3 umgesetzt wird, wenn es sich bei den Klassen `Figur` (7.2) und `Obst` (7.3) um abstrakte Klassen handelt. Die angegebenen Methoden (zeichne(), getFlaeche(), getName(), getFarbe()) sind ebenfalls abstrakte Methoden.

7.5 INTERFACES 1

Eine Liste ganzer Zahlen größer oder gleich 0 soll als Interface `IntegerList` implementiert werden. Dazu sollen die folgenden abstrakten Methoden verwendet werden:

```
int getLength()
// liefert die Länge der gesamten Liste

void insertLast(int value)
// fügt value am Ende der Liste an

int getFirst()
// liefert das erste Element der Liste

void deleteFirst()
// löscht das erste Element der Liste

boolean search(int value)
// prüft, ob value in der Liste vorkommt.
```

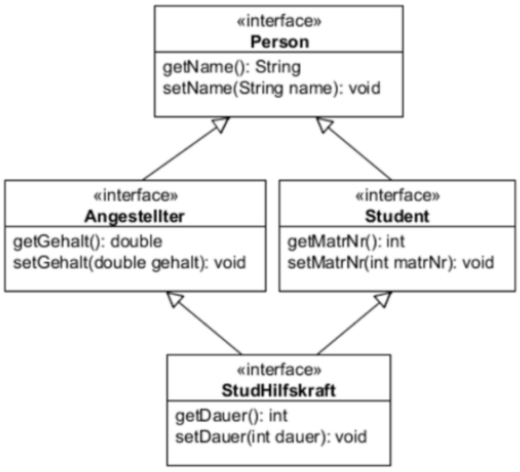
Implementieren Sie dieses Interface mithilfe eines Arrays in der Klasse ArrayIntegerList und testen Sie alle Methoden.

7.6 INTERFACES 2

Erweitern Sie die Klasse Figur aus Aufgabe 7.2 um die Schnittstelle `Comparable<Figur>` so dass die Objekte nach Größe der Fläche sortiert werden können. Testen Sie.

7.7 INTERFACES 3

Implementieren Sie das folgende UML-Diagramm:



8. ÜBUNGSBLATT

8.1 EXCEPTIONS 1

Welche Ausnahmen müssen abgefangen werden, wenn der folgende Code-Schnipsel übersetzt werden soll. Nutzen Sie dazu die Javadoc-Dokumentation.

```
Clip clip = AudioSystem.getClip();
clip.open(AudioSystem.getAudioInputStream (new
File("audio.wav"))); clip.start();
TimeUnit.MICROSECONDS.sleep(clip.getMicrosecondLeng
th()+50);
clip.close();
```

Kann/Sollte man Ausnahmen zusammenfassen?

8.2 EXCEPTIONS 2

Eine Datei lässt sich in Java mit folgendem Quellcode einlesen:

```
String filename = ....
List<String> lines =
Files.readAllLines(Paths.get(filename));
```

Erstellen Sie in Ihrem Projektordner eine eigene Datei, die aus mehreren Zeilen mit jeweils einem Wort besteht. Schreiben Sie ein Programm, das ermittelt, welches Wort in der Datei das längste ist. Welche Ausnahmen müssen Sie dabei berücksichtigen?

8.3 EXCEPTIONS 3

Schreiben Sie ein Programm, das analog zu Aufgabe 3 eine Datei ein eine String-Liste einliest. Die Datei soll aber in diesem Fall pro Zeile eine Zahl beinhalten.
Konvertieren Sie die eingelesenen Zeilen so, dass die Zahlen in ein Double-Array ein- gelesen werden. Achten Sie auf Ausnahmen derart, dass bei fehlerhaften Einträgen das Programm nicht abbricht, sondern die entsprechende Zeile ignoriert und die fehlerhafte Zeile auf der Konsole ausgegeben wird.

8.4 EXCEPTIONS 4

a) Schreiben Sie eine Klasse Person, die die Attribute `vorname`, `nachname` und `alter` umfasst. Sofern ein Objekt dieser Klasse erzeugt wird, soll eine `IllegalArgumentException` ausgelöst werden, falls der Nachname leer ist oder das Alter kleiner als 0 oder größer als 150 gesetzt wird. Testen Sie mit einem entsprechenden Hauptprogramm.
b) Erstellen Sie eine eigene Ausnahmeklasse `PersonException` und passen Sie die Klasse so an, dass in den oben genannten Fällen eine `PersonException` ausgelöst wird.

9. ÜBUNGSBLATT

9.1 WRAPPER 1

Implementieren Sie ein Programm, bei dem der Nutzer eine beliebige natürliche Zahl eingeben kann. Diese Zahl soll vom Programm dann in binärer, in oktaler und in hexadezimaler Darstellung ausgegeben werden.

Hinweis: Schauen Sie sich die Dokumentation einer bestimmten Wrapper-Klasse genauer an.

Erweitern Sie Ihr Programm derart, dass der Nutzer sowohl dezimale, binäre, oktale und hexadezimale Zahlen eingeben kann.

9.2 WRAPPER 2

Addieren Sie in einer Schleife die Zahlen von 1 bis `Integer.MAX_VALUE`. Worauf müssen Sie beim Vergleich der Schleifenvariable mit `Integer.MAX_VALUE` achten?

Speichern Sie das Ergebnis in einer Variablen des elementaren Datentyps `long` und messen Sie die Laufzeit.

Ändern Sie den elementaren Datentyp long in die Wrapper-Klasse Long und messen Sie erneut.

Hinweis: Überlegen Sie sich, wie Sie die Laufzeit eines Programmes messen könnten. Die Klasse System kann Ihnen dabei helfen. Wenn das Programm nach wenigen Sekunden noch nicht beendet ist, dann haben Sie einen Fehler in Ihrer Schleife.

9.3 VECTOR

Erstellen Sie eine Klasse Artikel, die die Instanzvariablen `id`, `preis` und `menge` enthalten soll. Hierzu sind ein Konstruktor und die jeweiligen `get/set`-Methoden zu definieren. Erstellen Sie eine weitere Klasse Warenkorb, die mit Hilfe eines `Vector`-Objekts Artikel-Objekte aufnehmen können soll.

Implementieren Sie die folgenden Methoden:

```
public void add(Artikel artikel)
/* fugt ein Artikel-Objekt in den Warenkorb ein. */

public double bestellwert()
/* liefert den Bestellwert (preis * menge) aller
Artikel im Warenkorb. */
```

10. ÜBUNGSBLATT

10.1 KÜHLSCHRANK

Erstellen Sie eine Klasse `Kuehlschrank`, die den Inhalt eines Kühlschranks verwalten soll, also die Lebensmittel und deren vorrätige Quantität. Um dies zu erreichen, nutzen Sie eine `HashMap` mit einem Key für das Lebensmittel (Datentyp String) und einem geeigneten Datentypen für die Quantität.

Schreiben Sie eine Methode, die das Befüllen des Kühlschranks ermöglicht. Nutzen Sie dazu Eingaben/Ausgaben auf der Konsole. Vermeiden Sie doppelte Einträge (also, dass der Nutzer mehrfach dasselbe Lebensmittel hinzufügt). Die Methode soll eine eigene *KuehlschrankException* auslösen, wenn der Nutzer anstelle einer Menge eine falsche Eingabe macht.

Schreiben Sie anschließend eine Methode, die die Menge eines Lebensmittels ausgibt, das der Nutzer über die Konsole eingeben kann.

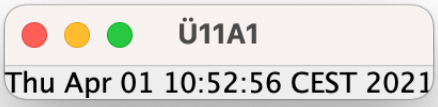
Danach erstellen Sie noch eine Methode, die den gesamten Inhalt des Kühlschranks auflistet.

Erstellen Sie eine Testklasse, die die Klasse *Kuehlschrank* testet.

11. ÜBUNGSBLATT

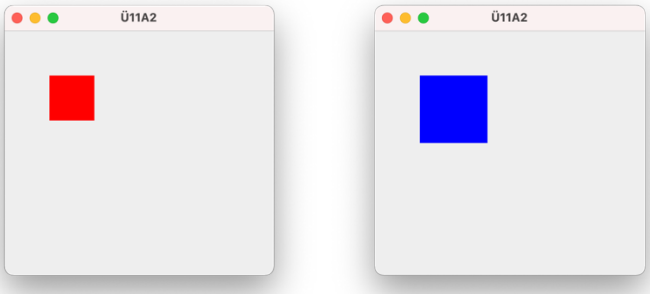
11.1 DATUM

Implementieren Sie die in der folgenden Abbildung dargestellten Java-Anwendung. Diese soll per Timer gesteuert in einem `JLabel` die aktuelle Uhrzeit anzeigen.



11.2 FARBIGE PANEL

Erstellen Sie eine Java-Anwendung, wie in der folgenden Abbildung dargestellt. Das rote Panel soll sich vergrößern und die Farbe auf blau ändern, sobald der Benutzer die Maus in das Panel bewegt. Wenn die Maus das Panel wieder verlässt, dann soll die ursprüngliche Größe und Farbe wiederhergestellt werden.

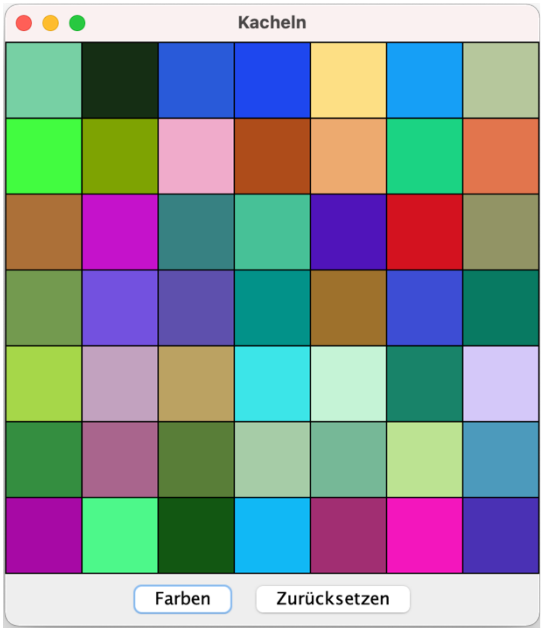


12. ÜBUNGSBLATT

12.1 MEHR FARBIGE PANEL

In einem Fenster sollen 7x7 weiße Kacheln angezeigt werden, wie unten dargestellt. Wird der Button Farben gedrückt, so sollen alle Kacheln mit einer zufälligen Farbe gefüllt werden. Wird nur ein einzelnes Panel angeklickt, so wird die Farbe für dieses Panel zufällig gesetzt. Ein Klick auf Zurücksetzen setzt alle Kacheln wieder auf Weiß.

Hinweis: Schauen Sie sich die Konstruktoren der Klasse `Color` *an.*



II. VERTIEFUNGSÜBUNGEN

1. ANWEISUNGEN UND OPERATOREN

1.1 OPERATOREN 1

Betrachten Sie die Deklarationen

```
int a = 76, b = 37, c;
```

Nehmen Sie an, dass die folgenden Anweisungen der Reihe nach ausgeführt werden. Welche Werte haben `a`, `b` und `c` nach jeder Anweisung?

```
c = b % a * 9;
a++;
b--;
c += a + b;
c %= 2 * 5;
c++;
```

1.2 TYPKONVERTIERUNGEN

Welche impliziten Typkonversionen werden in den folgenden Ausdrücken vorgenommen? Welche Werte haben die Variablen `y`, `i` und `x` nach der Zuweisung? Gehen Sie von folgenden Deklarationen aus:

```
double x = 23.57, y;
float z = 4.5F;
int i;
short s = 5;
```

a) y=2*z;
b) i=s*3;
c) x = z/s;

1.3 OPERATOREN 2

Betrachten Sie das folgende Programm. Tragen Sie die Werte ein, welche die Variablen `a` bzw. `b` nach Ausführung der jeweiligen Anweisung haben.

```
public static void main(String[] args)
{
  int a, b;
  a = 20;
  a-=4;              // a = ________

  a = 20;
  b = 4;
  a+=b;              // a = ________
  b *= a;            // b = ________
  a %= b + 6;        // a = ________

  a = 35;
  b = 88;
  a++;               // a = ________
  b--;               // b = ________
}
```

1.4 OPERATOREN 3

Wodurch unterscheiden sich die Ausdrücke

```
a=5 a == 5
```

1.5 PRIMITIVE 1

Die Variable `ch` sei vom Typ `char`.

a) Geben Sie einen logischen Ausdruck an, mit dem Sie testen können, ob `ch` eine Dezimalziffer ist.
b) Stellen Sie fest, ob `ch` ein großes `Y` oder ein kleines `y` ist.
c) Stellen Sie fest, ob `ch` ein Vokal (Selbstlaut; a, e, i, o oder u) ist.

1.6 OPERATOREN 4

Die `int`-Variable `x` habe den Wert 5. Die folgende Anweisung führt dennoch nicht zu einer Division durch 0. Warum?

```
if(x!=5 && 1.0/(x–5)>0) y = 1;
```

1.7 BEDINGTE AUSDRÜCKE 1

Geben Sie für die folgenden bedingten Ausdrücke jeweils eine gleichwertige `if`- Anweisung an.

```
boolean odd;
int x, y, value;
```

a) odd = value % 2==1 ? true:false;
b) value = x>y?x:y;

1.8 BEDINGTE AUSDRÜCKE 2

Wandeln Sie die folgende if-Anweisung in einen bedingten Ausdruck um, welcher der Variablen c einen Wert zuweist.

```
if (a < 2 * b)
    c = a;
else
    c = b;
```

1.9 BEDINGTE AUSDRÜCKE 3

Die Kosten für das Mieten eines Kleinwagens betragen das Maximum aus entweder 45,- € pauschal oder aber 10,- € plus 0,35 € pro gefahrenem Kilometer. Der Mietpreis beträgt also mindestens 45,- €. Geben Sie einen bedingten Ausdruck an, welcher der Variablen `mietpreis` den entsprechenden Betrag zuweist. Die folgenden Deklarationen seien gegeben:

```
double gefahreneKm, mietpreis;
```

1.10 ABFRAGEN 1

Nehmen Sie an, dass `x`, `y` und `val` vom Typ `int` sind. Geben Sie für die folgende Wertzuweisung eine gleichwertige `if`- Anweisung an.

```
val = x < y && y != 0 ? 1 : 0;
```

1.11 GLEITKOMMA ARITHMETIK

Die Zahlendarstellung für float und double verwendet eine feste Zahl von Bits. Daher ist die Repräsentation von Gleitkommazahlen in einem Rechner „grobkörnig“, d. h. es gibt „Löcher“, in denen keine Gleitkommazahlen liegen.

Geben Sie das folgende Programm ein und lassen Sie es ablaufen. Sie werden feststellen, dass irgendwann 1+d == 1 gilt. Was sagt dies über die Gleitkommazahlen in der Nähe des Wertes 1 aus?

```
public class fpNumbers
{
  public static void main(String args[])
  {
    int i;
    double d = 0.1;
    for(i = 1; i <= 20; i++)
    {
      System.out.println(1.0 + d);
      d = d / 10;
    }
  }
}
```

2. ABFRAGEN

2.1 ABFRAGEN 1

Welche Ausgabe erzeugt das folgende Programmstück?

```
int x = 80, y = 60;
if (x < 100)
  if (y > 50)
    System.out.println("eins");
  else
    System.out.println("zwei");
else
  System.out.println("drei");
```

2.2 ABFRAGEN 2

Welche Ausgabe erzeugen die folgenden Anweisungen?

```
int x = 110, y = 40;
if (x < 100)
{
  If (y > 50)
    System.out.println("eins");
}
else
{
  System.out.println("zwei");
  System.out.println("drei");
}
```

2.3 ABFRAGEN 3

Führen Sie die folgende Anweisung „von Hand“ aus. Welche Ausgabe wird für jeden der Werte von t erzeugt?

```
if (t < 15)
  if (t > 7)
    System.out.println("eins");
  else
```

```
    System.out.println("zwei");
else
  if(t < 18)
    System.out.println("drei");
System.out.println("Ende");
```

a) t=9
b) t=20
c) t=4
d) t=7
e) t=16

2.4 ABFRAGEN 4

Führen Sie die folgende Anweisung „von Hand“ aus und beantworten Sie die angegebenen Fragen.

```
if (a < b)
  if (m > n)
    System.out.println("schwarz");
  else
    System.out.println("rot");
 else
   if(m > n)
     System.out.println("gruen");
   else
     System.out.println("blau");
```

a) Was wird ausgegeben für a= 3, b= 4, m= 1 und n= 5?
b) Was wird ausgegeben für a= 5, b= 3, m= 7 und n= 4?
c) Was wird ausgegeben, wenn a<b und m>n gilt?

2.5 ABFRAGEN 5

Führen Sie die folgende bedingte Anweisung „von Hand“ aus. Welche Ausgabe wird für jeden der Werte von n erzeugt?

```
if (n <= 10)
  if (n/3 < 2)
    System.out.println("eins");
  else if(n/3 >= 3)
    System.out.println("zwei");
  else
    System.out.println("drei");
else if(n > 20)
  System.out.println("vier");
else
  System.out.println("fuenf");
```

a) n=17
b) n=25
c) n=7
d) n=0
e) n=9

2.6 SWITCH-CASE 1

Schreiben Sie eine `switch`-Anweisung, die äquivalent zur folgenden `if`-Anweisung ist. a (vom Typ int) soll als Auswahlausdruck verwendet werden.

```
if(a == 3)
  result = a;
else if(a == 6)
  result = a + 10;
else if(a == 10)
  result = a + 20;
else
  result = a + 30;
```

2.7 SWITCH-CASE 2

Schreiben Sie eine `if`-Anweisung, die zur folgenden `switch`-Anweisung äquivalent ist.

```
switch(n) {
  case 39:
  case 15: System.out.println("gewonnen");
           break;
  default: System.out.println("verloren");
           break;
}
```

2.8 SWITCH-CASE 3

Betrachten Sie die folgende Fallunterscheidung:

```
switch(n) {
  case 1:
  case 3:
  case 4:  System.out.println(n);
           break;
  case 2:
  case 5:
  case 6:  System.out.println(n * 10);
           break;
  default: System.out.println(n * 100);
           break;

}
```

Was wird ausgegeben für

a) n=5
b) n=4
c) n=8

Nehmen Sie an, dass die break-Anweisungen aus der `switch`-Anweisung entfernt wurden. Was würde ausgegeben für

d) n=5
e) n=4
f) n=8

2.9 SWITCH-CASE 4

Geben Sie zur folgenden `if`-Anweisung eine gleichwertige `switch`-Anweisung an.

```
if(ch == 'a' || ch == 'b' || ch == 'c')
  System.out.println("im Bereich \"abc\"");
else if(ch >= 'g' && ch < 'i')
  System.out.println("in der Mitte des Alphabets");
```

```
else if(ch == 'z')
  System.out.println("ein z, ein z, ...");
else
  System.out.println("andere Buchstaben");
```

2.10 SWITCH-CASE 5

Schreiben Sie eine `switch`-Anweisung, die äquivalent zur folgenden `if`-Anweisung ist. Wie beurteilen Sie Ihre Lösung im Vergleich zur vorliegenden `if`-Anweisung?

```
if(n < 5 || n > 8)
  System.out.println("Bereich 1");
else if(n >= 5 && n <= 7)
  System.out.println("Bereich 2");
else
  System.out.println("Bereich 3");
```

2.11 ABFRAGEN 6

Nehmen Sie an, dass `x`, `y`, `z` und `val` vom Typ `double` sind. Geben Sie für die folgende Wertzuweisung eine gleichwertige `if`-Anweisung an.

```
val = x >= y && x <= z ? z-y : x < y ? 0 : 1.0E25;
```

3. SCHLEIFEN

3.1 WHILE-SCHLEIFE 1

Die folgende while-Schleife soll die Ausgabe 1 4 9 16 25 erzeugen. Ergänzen Sie den Programmcode entsprechend.

```
int i = 1;
while (_____________)
{
  System.out.print(_________+ " ");
  _________;
}
```

3.2 WHILE-SCHLEIFE 2

Die folgende while-Schleife soll die Ausgabe 100 64 36 16 4 erzeugen. Ergänzen Sie den Programmcode entsprechend.

```
int i = _________;
while (_________)
{
 System.out.print_________+ " ");
 _________;
}
```

3.3 WHILE-SCHLEIFE 3

Führen Sie die folgende while-Anweisung „von Hand" aus. Welchen Wert hat die Variable n nach dem Abarbeiten der Schleife?

```
int n = 0, i = 1;
while (i < 8)
{
  n += i;
  i += 2;
}
```

3.4 WHILE-SCHLEIFE 4

Führen Sie die folgende while-Anweisung „von Hand“ aus. Welchen Wert hat die Variable n nach dem Abarbeiten der Schleife?

```
int n = 0, jahr = 1980;
while (jahr <= 1989)
{
  if (jahr % 4 == 0)
    n++;
  jahr++;
}
```

3.5 DO-WHILE-SCHLEIFE 1

Führen Sie die folgende do-while-Anweisung „von Hand“ aus. Was wird ausgegeben?

```
int i = 5;
do
{
  System.out.print(i*i + " ");
  i--;
}
while (i>0);
```

3.6 DO-WHILE-SCHLEIFE 2

Führen Sie die folgende do-while-Anweisung „von Hand“ aus. Was wird ausgegeben?

```
int i = 5;

do
{
  i--;
  System.out.print(i*i + " ");
}
while (i>0);
```

3.7 WHILE-SCHLEIFE 5

Welche Ausgabe erzeugt der folgende Java-Code?

```
int sum = 0, i = 1;
while (i <= 10)
{
  if (i % 2 == 0)
    sum += i;
  i++;
}
System.out.println(sum);
```

3.8 DO-WHILE-SCHLEIFE 3

Welche Ausgabe erzeugt der folgende Java-Code?

```
int fact = 1, i = 1;

do
{
  fact *= i;
  i++;
```

```
} while(i<6);
System.out.println(fact);
```

3.9 WHILE-SCHLEIFE 6

Schreiben Sie eine while-Schleife, welche die ganzen Zahlen von 10 bis 100 ausgibt, jede Zahl in einer eigenen Zeile.

3.10 DO-WHILE-SCHLEIFE 4

Geben Sie für die folgende while-Schleife eine gleichwertige do-while-Schleife an.

```
int wert = 100, i = 1;
while (wert > 0)
{
  wert -= i*i;
  i--;
}
```

3.11 WHILE-SCHLEIFE 7

Führen Sie die folgende while-Anweisung „von Hand“ aus. Welchen Wert hat die Variable n nach dem Abarbeiten der Schleife?

```
int n = 0, i = 5;
while (i>0)
{
  n += 2 * i;
  i--;
}
```

3.12 DO-WHILE-SCHLEIFE 5

Führen Sie die folgende do-while-Anweisung „von Hand“ aus. Welchen Wert hat die Variable n nach dem Abarbeiten der Schleife?

```
int n = 0, i = 1;
do
{
  n += 10;
  i++;
} while (i<=5);
```

3.13 SCHLEIFEN 1

Schreiben Sie Schleifen, welche die folgenden Algorithmen implementieren.

a) Addieren Sie mit einer while-Schleife alle geraden ganzen Zahlen von 2 bis 20.
b) Addieren Sie mit einer do-while-Schleife die ganzen Zahlen 1, 2, 3, ... bis die Summe größer als 500 geworden ist. Geben Sie die Summe aus
c) Eine while-Schleife, startet mit n=10 und gibt 10^2, 8^2, ... 2^2 aus.

3.14 FOR-SCHLEIFE 1

Geben Sie für die folgende while-Schleife eine gleichwerte for-Schleife an.

```
int i = 10;
while (i>=1)
{
  System.out.print(i + " ");
  i--;
}
```

3.15 FOR-SCHLEIFE 2

Geben Sie für die folgende for-Anweisung eine äquivalente while-Schleife an.

```
for (int i=1; i<10; i++)
  System.out.println(i+5);
```

3.16 FOR-SCHLEIFE 3

Geben Sie für die folgende while-Schleife eine äquivalten for-Schleife an.

```
int i = 1, sum = 0;
while (i<= 15)
{
  sum += i;
  i += 2;
}
```

3.17 FOR-SCHLEIFE 4

Führen Sie die folgende Anweisungen „von Hand" aus. Was wird ausgegeben?

a)

```
for (int i=1; i<5; i++)
  System.out.println(i*10);
```

b)

```
for (int i=5; i>0; i--)
  System.out.println(i*i);
```

3.18 FOR-SCHLEIFE 5

Die Variable count führt Buch über die Gesamtzahl der Iterationen (Schleifendurchläufe). Welchen Wert hat count nach dem Abarbeiten der folgenden Anweisungen?

a)

```
int count = 0;
for (int i=1; i<=5; i++)
  for (int j=1; j<=3; j++)
    count++;
```

b)

```
int count = 0;
for (int i=1; i<=3; j++)
  for (int j=0; j<i; j++)
    count++;
```

3.19 FOR-SCHLEIFE 6

Welche Ausgabe erzeugt der folgende Programmcode?

```
int i, j;

for (i=1, j=5; i+2*j > 9; i++, j--)
  System.out.println(2*i + 3*j);
```

3.20 FOR-SCHLEIFE 7

Führen Sie die folgende for-Anweisung „von Hand“ aus. Welches Problem tritt hier auf?

```
for (int n=5; n<25; n++)
{
  System.out.println(n);
  n = Einlesen.LiesInt();
}
```

Welche Ausgabe wird erzeugt bei den folgenden Eingabewerten:

- 10
- 22
- 6
- 23
- 24

3.21 FOR-SCHLEIFE 8

Ergänzen Sie jeweils den Schleifentest derart, dass 12 Sternchen ausgegeben werden.

a)

```
for (int i=1; i<=3; i++)
  for (int j=1; _______; j++)
    System.out.print(“*”);
```

b)

```
for (int i=3; i<5; i++)
  for (int j=6; _______; j--)
    System.out.print(“*”);
```

c)

```
for (int i=1; i<8; i+= 2)
  for (int j=2; _______; j+= 3)
    System.out.print(“*”);
```

3.22 SCHLEIFEN 2

Wieviele Sternchen geben die folgenden geschachtelten Schleifen aus? Die Deklarationen: int i, j, k; seien gegeben.

a)

```
int i = 0;
while (i<=10)
{
  j=1;
  while(j<i)
  {
    System.out.print(“*”);
    j++;
```

```
    }
    i++;
  }
```

b)

```
for (i=0; i<3; i++)
  for (j=0; j<3; j++)
    for (k=0; k<2; k++)
      System.out.print("*");
```

3.23 SCHLEIFEN 3

Schreiben Sie eine Schleife, die bis zu 10 ganze Zahlen einliest und jede dieser Zahlen nach den folgenden Regeln verarbeitet.

1) Die Schleife wird beendet, wenn der Eingabewert 6 ist.
2) Wenn der Eingabewert durch 3 teilbar ist, wird der nächste Schleifendurchlauf begonnen.
3) Wenn der Eingabewert kleiner als 8 ist, wird das Quadrat der Zahl ausgegeben; andernfalls wird die Zahl selbst ausgegeben.

4. ARRAYS

4.1 ARRAYS 1

Schreiben Sie ein Programm, welches in ein Array mit 10 Feldern die Zahlen 0^2 bis 9^2 belegt. Geben Sie anschließend das Array über eine entsprechende Schleife aus.

4.2 ARRAYS 2

Schreiben Sie ein Programm, welches 10 Integer-Werte vom Benutzer erfragt und in einem Array speichert. In einem Verarbeitungsschritt sollen alle Werte verdoppelt werden. Geben Sie anschließend das Array über eine entsprechende Schleife aus.

4.3 ARRAYS 3

Erzeugen Sie ein Array mit 100 Werten und füllen Sie dieses mit den Zahlen 1 bis 100. Setzen Sie das erste Feld des Arrays auf 0. Es sollen nun schrittweise die Zahlen entfernt werden, die Vielfache einer bereits vorhandenen Zahl sind. Beginnen Sie mit der 2 und entfernen Sie die 4, 6, 8, 10, usw. Entfernen Sie anschließend die Vielfachen der 3 und somit die 6, 9, 12, 15. Zum Entfernen können Sie den Wert auf 0 setzen. Geben Sie anschließend eine Liste der verbleibenden Zahlen (alle ungleich 0) aus.

4.4 ARRAYS 4

Definieren Sie ein Array mit beliebigem Inhalt. Realisieren Sie über ein Schleifenkonstrukt, dass die Reihenfolge des Arrays umgedreht wird. Versuchen Sie Ihre Implementierung dahingehend zu optimieren, dass kein zweites Array benötigt wird.

4.5 ARRAYS 5

Definieren Sie ein 3x3-Array welches stellvertretend für ein Soduku-Feld steht. Dies bedeutet, dass in jeder Zeile und Spalte die Zahlen 1 bis 9 nur 1-Mal vorkommen dürfen. Realisieren Sie eine solche Gültigkeitsprüfung. Beispielhaft können Sie das folgende Array zur Prüfung nehmen.

```
int[][] array = {
    { 1, 2, 3 },
    { 4, 5, 6 },
    { 7, 8, 9 }
};
```

5. OBJEKTE UND KLASSEN

5.1 ATTRIBUTE UND OPERATIONEN 1

Welche Attribute und Operationen beschreiben die realen Objekte

a) Digitalwecker,
b) Computermonitor?

5.2 ATTRIBUTE UND OPERATIONEN 2

Welche Attribute und Operationen beschreiben ein Sparkonto? Welche Aktionen muss der Konstruktor ausführen?

5.3 ATTRIBUTE UND OPERATIONEN 3

Ein Radio ist ein Beispiel für eine Klasse.

a) Welche Attribute und Operationen beschreiben die Klasse Radio?
b) Ein Objekt ist ein Exemplar einer Klasse. Geben Sie Beispiele für verschiedene Radio-Objekte an.

5.4 ATTRIBUTE UND OPERATIONEN 4

Die Klasse Rechteck habe die Attribute laenge und breite. Als Operationen stehen z. B. flaeche() und umfang() zur Verfügung, welche die Fläche bzw. den Umfang des Rechtecks berechnen und als Rückgabewert liefern.

a) Welche weiteren Operationen benötigt die Klasse Rechteck, welche könnten nützlich sein?
b) Geben Sie eine Beschreibung der Klasse Rechteck an, die auch die Argumente, Aktionen und Rückgabewerte der Operationen umfasst.

5.5 ATTRIBUTE UND OPERATIONEN 5

Überlegen Sie sich, welche Attribute und Operationen die Klasse Kreis benötigt.

Geben Sie eine Beschreibung der Klasse Kreis an, die auch die Argumente, Aktionen und Rückgabewerte der Operationen umfasst.

5.6 KLASSENIMPLEMENTIERUNG 1

Implementieren Sie die Klassen `Rechteck` und `Kreis` entsprechend Ihrer angefertigten Klassenbeschreibungen.

Schreiben Sie eine Klasse `FigurenTest`, die eine Funktion `main()` besitzt, mit der Sie Ihre Klassen testen können. Verwenden Sie zum Test auch den folgenden Programmcode:

```
Rechteck box = new Rechteck(4, 5);
box.setKanten(box.getLaenge() * 2, box.getBreite()
* 2);
System.out.println("Flaeche: " + box.flaeche() +
                   " Umfang: " + box.umfang() +
                   "\n");

double len, wid, randFlaeche, randUmfang;
len = Einlesen.LiesDouble();
wid = Einlesen.LiesDouble();
Rechteck smBox = new Rechteck(len, wid);
Rechteck lgBox = new Rechteck(len + 1.0, wid +
1.0);
```

5.7 KLASSENIMPLEMENTIERUNG 2

Die Klasse `RegistrierKasse` ermöglicht es den Mitarbeitern eines Kaufhauses, den Preis einer Ware einzugeben, zieht einen prozentualen Rabatt ab, den das Kaufhaus auf alle Waren gewährt, und zeigt den zu zahlenden Betrag an. Sie gibt einen Kassenbon aus, der den an der Ware befindlichen Preis, den Rabatt in % und den zu zahlenden Preis ausgibt.

a) Überlegen Sie sich, welche Attribute und Operationen die Klasse `RegistrierKasse` benötigt.
b) Geben Sie eine Beschreibung der Klasse an, die auch die Argumente, Aktionen und Rückgabewerte der Operationen umfasst.

5.8 KLASSENINITIALISIERUNG

Erstellen Sie eine neue Klasse `MeineKlasse` und einen Klasseninitalisierer:

```
public MeineKlasse()
{
 System.out.println("A");
}

static {
 System.out.println("B");
}

public static void main(String[] args)
{
 System.out.println("C");
 new MeineKlasse();
 System.out.println("D");
}
```

In welcher Reihenfolge erscheint die Ausgabe auf der Konsole?

5.9 OBJEKTINITIALISIERUNG

Ergänzen Sie den Quelltext aus Aufgabe 5.8 um einen Instanzintialisierer (hier mit Ausgabe „E“):

```
public MeineKlasse()
{
 System.out.println("A");
}

static {
 System.out.println("B");
}

public static void main(String[] args)
{
 System.out.println("C");
 new MeineKlasse();
 System.out.println("D");
}

{
 System.out.println("E");
}
```

In welcher Reihenfolge erscheint die Ausgabe auf der Konsole?

5.10 KLASSEN- UND OBJEKTINITIALISIERUNG

Ergänzen Sie den Quelltext aus Aufgabe 5.9 um weitere Klassen- und Objektinitialisierer. In welcher Reihenfolge werden diese auf der Konsole ausgegeben?

LITERATUR

- Christian Ullenbohm – Java ist auch eine Insel (2020), siehe auch http://openbook.rheinwerk-verlag.de/javainsel (Achtung: alte Auflage!)
- Dietmar Abts – Grundkurs JAVA (2020)
- Kathrin Passig, Johannes Jander – Weniger schlecht programmieren (2013)
- Helmut Balzert – Lehrbuch der Softwaretechnik (2011)
- http://openbook.rheinwerk-verlag.de/oop
- http://scrumguides.org