Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 3 zur Vorlesung Programmieren

Ausgabe: Fr 31.03.2023 – Abgabe: Fr 21.04.2023

Hinweis: Am kommenden Freitag, den 07.04. (Karfreitag) wird es kein neues Aufgabenblatt geben. Daher verlängert sich der Abgabezeitraum für dieses Blatt auf drei Wochen.

Aufgabe 1 (Quersummenberechnung):

Schreiben Sie ein Programm, das die Quersumme einer maximal dreistelligen Dezimalzahl berechnet. Zur Erinnerung: die Quersumme ist die Summe aller Ziffern einer Zahl. Die Quersumme von 523 ist daher 10.

Gehen Sie dabei wie folgt vor:

a) Erstellen Sie eine Klasse `Quersumme`.

b) Erstellen Sie darin eine Methode `int quersumme(int x)`, die die Quersumme der Zahl x berechnet. Ist es sinnvoller, hierfür eine "gewöhnliche" Methode oder eine Klassenmethoden zu verwenden? Begründen Sie Ihre Antwort.

c) Fügen Sie eine `main`-Methode hinzu, die eine Zahl von der Konsole einliest, deren Quersumme berechnet und diese ausgibt.

Analog kann man Quersummen für andere Stellenwertsysteme definieren, z.B. für das Hexadezimalsystem. Hier wäre die Quersumme von 157 z.B. D oder von 45A die Hexadezimalzahl 13.

d) Ergänzen Sie die Klasse Quersumme um eine Methode `quersummeHex`, die für (maximal) dreistelllige Hexadezimalzahlen deren Quersumme berechnet.

e) Passen Sie die `main`-Methode aus c) so an, dass zusätzlich eine Hexadezimalzahl eingelesen, deren Quersumme berechnet und als Hexadezimalzahl wieder ausgegeben wird. Aufgabe 2 kann Hinweise geben, wie man diese Ausgabe bewerkstelligt. Für das Einlesen der Hexadezimalzahl verwenden Sie eine passende Methode aus der Klasse `java.util.Scanner`.

Aufgabe 2 (Formatierung von Zahlen):

In der Klasse `String` gibt es eine Methode mit der Signatur `static String format(String format, Object... args)`.

a) Lesen Sie sich die Dokumentation der Methode durch auf der Seite `https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html`. Lesen Sie auch die Beschreibung des Parameters `format`.

b) Legen Sie eine Klasse `FormatDemo` in einer Datei `FormatDemo.java` an. Neben einer `main`-Methode soll es in der Klasse `FormatDemo` zwei weitere Methoden geben, `static void intFormatDemo(int x)` und `static void doubleFormatDemo(double d)`.

c) Die Methode `intFormatDemo` soll den ganzzahligen Parameter x zuerst ohne und dann nacheinander mit den folgenden Formatierungen, jeweils auf einer Zeile, ausgeben:

- Als Dezimalzahl mit Vorzeichen (d.h. auch bei positiven Zahlen soll ein Vorzeichen ausgegeben werden).
- Als 8-stellige Hexadezimalzahl mit einem vorangestellten Präfix "`0x`". Die Ziffern A bis F sollen als Großbuchstaben ausgegeben werden.
- Als 32-stellige Zahl im Binärsystem mit einem Präfix "`0b`". **Hinweis:** Sie benötigen hier eine zusätzliche Funktion zur Konvertierung in einen Binärstring. Finden Sie heraus, welche dies ist. Auch das Auffüllen führender Nullen ist in diesem Fall nicht möglich. Sie können aber von `format` mit Leerzeichen auffüllen lassen, und diese dann mittels der Methode `replace` der Klasse `String` die Leerzeichen durch Nullen ersetzen.
- Als Dezimalzahl in eckigen Klammern, zwischen denen Platz für 10 Zeichen ist. Die Zahl soll einmal linksbündig und einmal rechtsbündig zwischen den eckigen Klammern ausgegeben werden.

d) Rufen Sie die Methode `intFormatDemo` nacheinander mit den folgenden Werten in der `main`-Methode auf: 100, 1023, 0xFFF1234, -3.

e) Die Methode `doubleFormatDemo` soll den Fließkommawert d zuerst ohne und dann nacheinander mit den folgenden Formatierungen, jeweils auf einer Zeile, ausgeben:
- Als Dezimalzahl mit zwei Stellen hinter dem Komma, formatiert auf eine Gesamtbreite von 25 Zeichen.
- Als Dezimalzahl in wissenschaftlicher Notation, mit großem E als Exponent-Symbol (z.B. 2.5E10), ebenfalls mit einer Gesamtbreite von 25 Zeichen.
- In hexadezimalem Exponentialformat, formatiert auf 25 Zeichen.

f) Rufen Sie die Methode `doubleFormatDemo` nacheinander mit den folgenden Werten in der `main`-Methode auf: 2.5, 3.141592653, 1e12 und -0.0001234.

g) Erklären Sie für den Wert 2.5 die Ausgabe in hexadezimalem Exponentialformat und geben Sie an, wie aus der formatierten Ausgabe der Dezimalwert berechnet werden kann.

Aufgabe 3 (Heron-Verfahren):

Das Heron-Verfahren ist ein Näherungsverfahren zur Berechnung der Quadratwurzel $\sqrt{a}$ einer positiven reellen Zahl a. Dabei werden schrittweise zunehmend bessere Näherungswerte $x_0, x_1, x_2, \ldots$ für den Wert $x = \sqrt{a}$ (für $a > 0$) berechnet.

Die Vorschrift für die Berechnung des nächsten Näherungswerts x_{n+1} aus seinem Vorgänger x_n ist dabei wie folgt:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \quad .$$

Als Startwert x_0 kann man a verwenden, also die Zahl, von der die Wurzel gezogen werden soll.

a) Schreiben Sie ein Programm `Heron10`, das für eine von der Konsole gelesene Eingabe a die zehn ersten Annäherungsschritte an $\sqrt{a}$ mit dem Heron-Verfahren berechnet. Verwenden Sie dabei $x_0 = a$ und geben Sie die mittels der angegebenen Rekursionsformel berechneten Werte $x_1, \ldots, x_{10}$ auf der Konsole aus.

b) Gibt es Eingaben, für die im zehnten Schritt noch kein exaktes Ergebnis vorliegt?

c) Schreiben Sie nun ein zweites Programm, `HeronFixpoint`, das Näherungswerte so lange

berechnet, bis keine Änderung mehr eintritt, d.h. bis $x_{n+1} = x_n$ gilt. (Dieser Wert wird dann auch Fixpunkt der Rekursionsgleichung genannt.)

d) Überlegen Sie sich, ob der Fall eintreten kann, dass Ihr Programm `HeronFixpoint` nicht terminiert, d.h., dass $x_{n+1} = x_n$ nie (d.h. für kein $n \in \mathbb{N}$) gilt.

Aufgabe 4 (Spielkarten):

Schreiben Sie eine Klasse zur Repräsentation von Spielkarten. Wir gehen dabei von anglo-amerikanischen Pokerkarten aus, d.h. es gibt die Farben Kreuz, Pik, Herz und Karo, sowie die dreizehn Werte 2 bis 10, Bube, Dame, König und Ass, insgesamt also 52 Karten.

Gehen Sie dabei wie folgt vor:[1]

a) Legen Sie eine Enumeration `enum Suit` für die vier Farben an (verwenden Sie die anglo-amerikanischen Namen der Farben für dir Werte der Enumeration, schreiben Sie die deutschen Namen als Kommentar dazu).

b) Legen Sie eine Enumeration `enum Rank` für die dreizehn Werte an (verwenden Sie auch hier die anglo-amerikanischen Namen und schreiben Sie wiederum die deutschen Namen in einen Kommentar).

c) Fügen Sie den beiden Enumerationen jeweils eine Klassen-Methode `fromInt` hinzu (`enum`s erlauben Methoden, wie Klassen), mit der sich eine Ganzzahl in einen Wert der Enumeration umrechnen lässt. Welche Ganzzahl in welchen `enum`-Wert konvertiert wird, bleibt Ihnen überlassen. Es sollte sich aber eine eindeutige Zuordnung ergeben. Sie können zur Umrechnung eine `switch`-Anweisung verwenden. Die Methode soll die Signatur `static Suit fromInt(int suitNr)` bzw. `static Rank fromInt(int rankNr)` besitzen. Beide Methoden sollen `null` zurückgeben, falls der Wert des Arguments keine Zuordnung besitzt.

d) Legen Sie eine Klasse `class Card` zur Repräsentation von Spielkarten an und versehen Sie die Klasse mit einem passenden Konstruktor.

e) Fügen Sie der Klasse `Card` eine Methode `String name()` hinzu, um eine Zeichenkette mit dem Namen der Karte auf deutsch zu generieren.

f) Fügen Sie der Klasse `Card` eine Klassenmethode der Signatur `static Card random()` hinzu, die eine neue, zufällige Karte generiert. Verwenden Sie den Zufallszahlengenerator aus der Klasse `java.util.Random` (d.h. Sie benötigen eine `import`-Anweisung für diese Klasse am Beginn Ihres Programms) und die `fromInt`-Methoden der Klassen `Suit` und `Rank`. Legen Sie in der Klasse `card` ein Attribut `static Random randomGenerator` an, das Sie bei der Deklaration direkt initialisieren, und verwenden Sie die Methode `int nextInt(int b)` zur Bestimmung der nächsten benötigten Zufallszahl.

g) Legen Sie eine Klasse `public class CardGame` an. Fügen Sie der Klasse eine `main`-Methode hinzu, die zehn zufällige Karten anlegt und deren Namen ausgibt.

[1] Für die nachfolgenden Klassen und Enumerationen verwenden Sie bitte jeweils eine eigene Datei, die so wie die Klasse oder Enumeration, mit der Endung `.java`, heißen soll.