Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 3 zur Vorlesung Programmieren (fällig bis 16.11.25)

Hinweis: Ab jetzt sollten Sie in einer Entwicklungsumgebung (IDE) (z.B. IntelliJ wie in der Vorlesung gesehen) programmieren.

(Die Kompilierung von Projekten mit mehreren Quelltextdateien auf der Kommandozeile, so wie es ansonsten in Aufgabe 4 nötig wäre, wäre etwas umständlich; die Entwicklungsumgebung erledigt das von alleine für Sie)

Legen Sie in Ihrer Entwicklungsumgebung für jede Aufgabe jeweils ein eigenes Java-Projekt an.

In Java-Projekten, die von einer IDE gemanaged werden, ist es nicht üblich, mehrere gleichrangige Klassen in einer gemeinsamen Quelltext-Datei abzulegen (so wie Sie es bei Blatt 1 getan hatten). Statt dessen wird pro neuer Klasse[1] auch stets eine neue `.java`-Datei angelegt, die den gleichen Namen trägt wie die jeweilige Klasse.

Aufgabe 1 (Quersummenberechnung):

Zur Erinnerung: die Quersumme ist die Summe aller Ziffern einer Zahl. Die Quersumme von 523 ist beispielsweise daher 10.

a) Schreiben Sie ein Programm, das die Quersumme einer Dezimalzahl berechnet. Wenn Sie möchten, dürfen Sie sich auf Dezimalzahlen mit höchstens drei Stellen beschränken.

- Zur Berechnung der Quersumme sollten Sie mathematisch vorgehen. Ihre Implementierung sollte nicht davon abhängig sein, dass z.B. in der ASCII-Tabelle bestimmte Symbole in einer bestimmten Anordnung bzw. Reihenfolge vorkommen. Sie finden eine mathematische Strategie, indem Sie die folgenden beiden Fragen lösen:
  - Eine natürliche Zahl im Dezimalsystem lässt sich schreiben als die Summe aus einem Vielfachen von 10 sowie der letzten (niedrigstwertigen) Ziffer. Wie können Sie diese letzte Ziffer mit einer Rechenoperation erhalten, die Ihnen in Java zur Verfügung steht?
  - Was geschieht mit den Ziffern bei einer ganzzahligen Division einer Zahl durch 10 in Java? (Dies hilft Ihnen, die eben betrachtete letzte Ziffer von der ursprünglichen Zahl abzutrennen)
- Erstellen Sie eine Klasse `Quersumme`.
- 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.
- Fügen Sie eine `main`-Methode hinzu, die (per Scanner) eine Zahl von der Konsole einliest, deren Quersumme berechnet und diese ausgibt.

[1] eine Ausnahme bilden *innere* Klassen; diese kommen in Blatt 3 jedoch nicht vor

b) Analog kann man Quersummen für andere Stellenwertsysteme definieren, z.B. für das Hexadezimalsystem. Die Quersummen von 157 bzw. von 45A sind z.B. (hexadezimal notiert) D bzw. 13.

- Machen Sie sich zunächst klar, wie die in den beiden Beispielen genannten Ergebnisse zustande kommen.
- Ihre (mathematische) Strategie aus a) lässt sich mit minimalen Anpassungen am Code auf andere Stellenwertsysteme übertragen (nicht nur für die Basis 16, sondern z.B. auch zur Basis 13 oder 7). Überlegen Sie sich, welche mathematische Anpassung gegenüber a) gemeint ist.
- Ergänzen Sie die Klasse `Quersumme` um eine Methode `quersummeHex`, die für (ggf. maximal dreistelllige) Hexadezimalzahlen deren Quersumme berechnet.

  Falls Sie die Zahl als `String` einlesen, finden Sie in der Klasse `Integer` eine geeignete Konvertierungsmethode, die das Stellenwertsystem berücksichtigt – oder Sie verwenden direkt eine passende Methode der Scanner-Klasse.
- Passen Sie die `main`-Methode aus a) 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. Alternativ finden Sie auch in der Klasse `Integer` einige Möglichkeiten, die gewünschte Zeichenkette für die Ausgabe zu erzeugen.

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 offizielle Dokumentation der Methode durch; Sie finden diese für Java 21 z.B. auf der Seite `https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/String.html`.

   Lesen Sie außerdem die Beschreibung des Parameters `format`. Von dort gelangen Sie per Hyperlink zu einer (umfangreichen, aber vollständigen) Beschreibung der verschiedenen Formatierungsmöglichkeiten. Diese sind nach Datentypen geordnet und dokumentiert. Versuchen Sie, die folgenden Teilaufgaben nur mit Hilfe dieser Beschreibung zu lösen.

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.

Hinweis: Hier ist ein schriftlicher Rechenweg erforderlich!

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.
- Gibt es Eingaben, für die im zehnten Schritt noch kein exaktes Ergebnis vorliegt?

b)
- 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.)

  Geben Sie pro Schritt jeweils n und x_n aus, sodass Sie erkennen können, wie viele Schritte bis zum Erreichen des Fixpunkts nötig sind. (b.w.)

- 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:

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).

  Hinweis: Enumerationen sind spezielle Klassen; für sie gelten also die gleichen Bemerkungen wie oben in der Einleitung des Blatts.
- Bedenken Sie außerdem, dass die `enum`-Werte (also hier die vier Farben) Konstanten sind. Beachten Sie die entsprechende Namenskonvention.
- Legen Sie auch 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).

b) 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. (Hinweis für Fortgeschrittenere: An Exceptions ist hier also nicht gedacht!)

c)
- Legen Sie eine Klasse `class Card` zur Repräsentation von Spielkarten an und versehen Sie die Klasse mit einem passenden Konstruktor.
- Fügen Sie der Klasse `Card` eine Methode `String name()` hinzu, um eine Zeichenkette mit dem Namen der Karte auf deutsch zu generieren.
- 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.

(b.w.)

d) 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.

Hinweise:

- Die Ausgabe des Namens ist offenbar ein Vorgang, der zehnmal in der gleichen Weise ausgeführt wird. Je nachdem, wie umfangreich sich dies gestaltet (siehe nächster Hinweis), kann hierfür eine eigene Methode sinnvoll sein, die in `main` dann zehnmal verwendet wird.
- Die Frage, welcher Text für welche Farbe (bzw. für welchen Wert) auszugeben ist, können Sie entweder komplett hier in `CardGame` behandeln – oder Sie reichern die `enum`-Klassen aus a) mit Hilfsmethoden (z.B. mit Klassenmethoden wie in b)) an, die einen Teil der Arbeit dort schon erledigen. Beide Ansätze haben Vor- und Nachteile. Überlegen Sie sich kurz, welche das sein können; danach wägen Sie ab und entscheiden sich[2].
- Für Fortgeschrittene (also hier nicht verlangt!): Sie können den einzelnen `enum`-Werten, da es sich dabei um Klassen-Objekte handelt, auch Konstruktor-Parameter mitgeben, die in Objektattributen gespeichert werden (diese müsste man dann ebenfalls anlegen). So lässt sich eine `if`-Kaskade oder eine `switch`-Anweisung für die Ausgabe vermeiden.

[2] Diese Situationen treten bei der Softwareentwicklung oft auf; in der Vorlesung Software-Engineering werden Sie sich damit noch im Detail auseinander setzen.