Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 2 zur Vorlesung Programmieren (fällig bis 09.11.25)

Hinweise (diese gelten auch für zukünftige Übungen):

- Ab diesem Aufgabenblatt ist *vor der* Abgabe in ILIAS auch eine Präsentation der Lösung bei einem Tutor in der Übungsgruppe erforderlich.

Aufgabe 1 (Datei-Signaturen): Datei-Signaturen, auch *magic numbers* genannt, dienen der Erkennung und Prüfung, welchen Typ der Inhalt einer Datei besitzt. Sie sind, falls vorhanden, die ersten Bytes einer Datei. Unter `https://en.wikipedia.org/wiki/List_of_file_signatures` finden Sie eine Liste von gängigen Dateisignaturen.

a) Geben Sie die Signaturen von drei von Ihnen frei gewählten Audio-Dateiformaten an, außerdem die von Java `.class`-Dateien und von UTF-8- und UTF-16-Dateien. Geben Sie bei den Audioformaten den Namen des Formats an und wozu es verwendet wird/wurde.

b) Ein Hex-Editor ist ein Programm, mit dem sich die Bytes einer beliebigen Datei als Folge von Hexadezimalwerten (und meist zusätzlich als ASCII-Werte) darstellen lassen. Hex-Editoren können eingesetzt werden, um Dateiformate zu verstehen, binär codierte Dateien zu modifizieren oder eigene solche Dateien zu erstellen. Unter `https://hexed.it` können Sie den Online-Hex-Editor `HexEd.it` finden.[1]

   Verwenden Sie `HexEd.it`, um sich den Inhalt einer von Ihnen erstellten `.class`-Datei (z.B. von Übungsblatt 1) anzeigen zu lassen.

   Für diese Teilaufgabe (1b) ist keine Abgabe, nur eine Präsentation beim Tutor erforderlich.

Aufgabe 2 (UTF-Codierung): Verwenden Sie den Online-Hex-Editor `HexEd.it`, um einen Unicode-Text in verschiedenen UTF-Codierungen darzustellen und in einer Datei zu speichern. Die betrachteten Codierungen sollen sein

- UTF-8,
- UTF-16LE und
- UTF-16BE.

Verwenden Sie bei allen drei Codierungen eine Byte-Order-Mark (BOM) am Beginn der Datei (2 Byte bei UTF-16, 3 Byte bei UTF-8).[2] Der zu codierende Text soll aus den vier Unicode-Codepoints U+0041 (“A”), U+00F6 (“ö”), U+4210 (Han-Chinesisches Schriftzeichen), U+1D11E (Violinschlüssel) in genau dieser Reihenfolge bestehen.

(b.w.)

[1] Die vollständige Funktionalität ist nicht in allen Browsern verfügbar. Verwenden Sie `Chrome` oder `Edge`.
[2] Die BOMs für UTF-16 wurden in der Vorlesung genannt, die für UTF-8 müssen Sie selbst herausfinden.

Die Abgabe besteht aus den drei Dateien mit den verschiedenen UTF-Codierungen. Bei der Abnahme in der Übungsgruppe sollten Sie erklären können, wie sich die Werte der Codierungen berechnen.

Der Rechenweg (also, wie Sie aus den Unicode Codepoints die jeweiligen UTF-Codierungen erhalten) gehört (schriftlich) zur Abgabe dazu! Die nötigen Informationen finden Sie im Foliensatz P1.

Aufgabe 3 (Daten-Layout im Speicher): Geben Sie Speicherabbilder an, wie in der Vorlesung in Kapitel 2 gezeigt, für drei verschiedene Punkte im Ablauf eines Programms. Das relevante Programmfragment soll wie folgt aussehen:

```
Vector2D p = new Vector2D();
Vector2D q = new Vector2D();
Vector2D r = p;
p.x = 1.0;
p.y = 30.9;
q.x = -7.1;
q.y = 10.5;
// <Situation 1>
r = q;
r.x = 22.2;
// <Situation 2>
r = new Vector2D();
r.x = 0.0;
r.y = 0.0;
// <Situation 3>
```

Dabei soll die Klasse `Vector2D` wie in der Vorlesung definiert sein. Sie können annehmen, dass die Speicheradressen von 1 an aufsteigend nummeriert sind und neue Objektidentitäten und Referenzen immer an der kleinsten noch freien Adresse abgelegt werden. Ferner können Sie, wie in der Vorlesung, annehmen, dass Referenzen und Objektattribute (z.B. vom Typ `float`) jeweils genau eine Speicherstelle belegen.

a) Geben Sie Speicherabbilder zu den Situationen 1, 2 und 3 an.

b) Welche Referenzen sind Aliase in den Situationen 1, 2 und 3?

Aufgabe 4 (Flächen-Berechnung): Im Königreich Babaku gibt es jedes Jahr im Oktober ein Fest, auf dem das große Halloumi-Grillen stattfindet. Die eingesetzten Grills sind kreisrund, haben aber unterschiedliche Durchmesser. Die Halloumi-Käse-Stücke sind alle rechteckig und haben dieselbe Größe. Sie können aber so zusammengeschnitten werden, dass die Grillfläche soptimal genutzt wird.

a) Schreiben Sie ein Programm `Babaku`, das als Eingaben auf der Kommandozeile entgegennimmt:

- Den Durchmesser d des Grills in cm.
- Die Länge l und Breite b der Halloumi-Stücke in cm.

Das Programm soll berechnen und ausgeben, wie viele Halloumi-Stücke maximal (ohne Stapeln) auf den Grill passen. Auch Bruchstücke sind erlaubt.

(b.w.)

b) Ein Kind isst auf dem Fest immer ein Stück Halloumi, ein Erwachsener zwei. Erweitern Sie das Programm aus Teil a) so, dass als zusätzliche Eingabe die Anzahl Personen (sowohl Kinder als auch Erwachsene) entgegengenommen wird, und ausgerechnet wird, wie oft der Grill nacheinander *vollständig* befüllt werden muss (als Ganzzahl), um alle Personen zu versorgen.

   Hinweis: Es kann nötig sein, den Grill im letzten Durchlauf nochmal teilweise zu befüllen – dieser letzte Durchlauf soll nicht mitgezählt werden.

Hinweis: Zum Einlesen von Werten im Terminal können Sie einen `Scanner` verwenden, wie auf Seite 8 des Foliensatzes P1 angegeben, Ausgaben erfolgen mittels `System.out.println(...)`.

Aufgabe 5 (Bremsweg-Berechnung): Fährt ein Auto mit der Geschwindigkeit v (in km/h), so wird als Faustformel zur Berechnung des Bremswegs b in Metern gerne die folgende Formel verwendet:

$$b = \frac{v}{10} \cdot \frac{v}{10} \ .$$

a) Schreiben Sie ein Programm `BrakingDistance`, das bei Eingabe einer Geschwindigkeit in km/h den Bremsweg in Metern anhand der angegebenen Faustformel berechnet.

b) Überlegen Sie sich, ob die Reihenfolge der Berechnungsschritte (Multiplikation, Divisionen) und/oder die Typen der Variablen einen Einfluss auf das Ergebnis haben kann. Beschreiben Sie Situationen, in denen dies der Fall ist.