Christian Bunse & Tilo Zülske
E-Mail: Vorname.Nachname@hochschule-stralsund.de
Büro: Haus 4/306b

Aufgaben zur Lehrveranstaltung

Laborpraktikum Software

SMSB, SMIB

Aufgabenblatt #2

1) *Hausaufgabe (Enum)*
2) *Hausaufgabe (Wert- und Referenztypen, Un-/Boxing)*
3) *Hausaufgabe (Lesen und Ausgeben einer Textdatei mit Streams)*
4) *Hausaufgabe (Exception)*
5) *Präsenzaufgabe (Debugging und Schreiben einer Textdatei)*
6) *Präsenzaufgabe (Encodings)*
7) *Hausaufgabe (Encodings)*
8) *Präsenzaufgabe (Encodings)*
9) *Hausaufgabe (Debugging, String, Parsing)*
10) *Hausaufgabe (String, Parsing, Streams)*

Die Präsenzaufgaben dieses Aufgabenblattes werden in den Laboren ab dem 06.10.2025 bearbeitet. Die Hausaufgaben dieses Aufgabenblattes sind bis **8:00 Uhr des Abgabetages (20.10.2025 - siehe Moodle)** abzugeben. Verspätete Abgaben werden nicht berücksichtigt.

Wichtig (1): Ab Aufgabenblatt 2 müssen alle Abgaben von Source-Code mit JavaDoc dokumentiert werden. Jedes Projektverzeichnis enthält hierzu ein Unterverzeichnis „doc“ in dem die generierte Dokumentation abgelegt wird.

Wichtig (2): Es werden nur Abgaben gewertet die im Source-Code (jede Klasse) deutlich den Namen des/der Autorin benennt.

Hinweis: Bitte beachten Sie, dass für alle Programmieraufgaben Testfälle existieren müssen. Hierzu reicht es aus, wenn die entsprechende Eigenschaft/Funktionalität mittels eines Methodenaufrufs in der Main-Routine angesprochen werden kann.

WICHTIG

Das Fehlen von JavaDoc Kommentaren, Autoren Name und/oder Testfällen führt zu Punktabzug und somit zu einer schlechteren Benotung oder ggf. Nichtbestehen des Testats.

Aufgabenblatt #2

1) Hausaufgabe (Enum)

[2 Punkte]

Ertsellen Sie ein vereinfachtes Modell eines Fahrkarten-Automaten. Die unterschiedlichen Ticketarten, Zonen und Preise sollen mithilfe eines Enums abgebildet werden. Dabei sollen zusätzliche Funktionalitäten wie Methoden, Felder, Polymorphie und Interaktion mit Collections umgesetzt werden.

Anforderungen

1. Enum TicketType

- Definieren Sie ein enum TicketType, das mindestens folgende Ticketarten enthält:
  - SINGLE (Einzelfahrkarte)
  - DAYPASS (Tageskarte)
  - WEEKLY (Wochenkarte)

- Jeder Enum-Wert soll:
  - Einen Preisfaktor besitzen (z. B. Einzelfahrkarte = 1.0, Tageskarte = 2.5, Wochenkarte = 10.0).
  - Eine Beschreibung (String).
  - Überschreiben Sie die Methode toString(), sodass eine menschenlesbare Ausgabe erfolgt.

2. Enum Zone

- Definieren Sie ein weiteres Enum Zone, das für unterschiedliche Tarifzonen steht, z. B.:
  - INNER_CITY, OUTER_CITY, REGIONAL.
    - Jede Zone besitzt: Eine Grundgebühr (z. B. 2.0€, 3.0€, 5.0€).

3. Berechnung des Ticketpreises
   - Schreiben Sie eine Methode calculatePrice(TicketType type, Zone zone), die den Preis berechnet: Preis = Zone.Gebühr * TicketType.Faktor

4. Interface Printable
   - Definieren Sie ein Interface Printable mit der Methode printDetails().
   - Lassen Sie beide Enums (TicketType und Zone) dieses Interface implementieren, sodass sie ihre Details (Name, Preisfaktor/Grundgebühr, Beschreibung) ausgeben können.
   -
5. Sammlung & Benutzerinteraktion
   - Implementieren Sie eine Klasse TicketMachine mit folgenden Methoden:
     - printAllTickets(): Gibt alle Ticketarten mit Preisen in allen Zonen aus (verschachtelte Schleifen oder Streams).
     - findCheapestTicket(): Findet das günstigste Ticket (Typ + Zone).
     - Nutzen Sie dafür Collections (List, Map, oder Stream-API).

Beispielausgabe:

Verfügbare Tickets:
SINGLE in INNER_CITY: 2.00 €

SINGLE in OUTER_CITY: 3.00 €
DAYPASS in INNER_CITY: 5.00 €
...

Günstigstes Ticket: SINGLE in INNER_CITY (2.00 €)

2) Hausaufgabe (Wert- und Referenztypen, Un-/Boxing)

[0,5 Punkte]

Beschäftigen Sie sich mit den Unterschieden zwischen Wert- und Verweistypen. Beschreiben Sie, worin der Unterschied zwischen Boxing und Unboxing besteht, und demonstrieren Sie diesen anhand eines Programmier-Beispiels.

3) Hausaufgabe (Lesen und Ausgeben einer Textdatei mit Streams)

[1 Punkt]

Schreiben Sie ein Programm, das eine Textdatei Byte für Byte einliest und auf der `Console` ausgibt. In den Vorlesungsfolien ist beschrieben, wie das geht. Der Benutzer soll den Dateinamen eingeben können. Das Programm soll prüfen, ob die Datei existiert (suchen Sie sich hierzu in der Klasse `File` eine geeignete Methode heraus), und ggf. eine Fehlermeldung ausgeben. Die Ausgabe soll nicht in Form von Bytes erfolgen, sondern in Gestalt von Zeichen. Dazu verwenden Sie die Methoden:

```
public static int decodeChar (char c) {
	return ((int) c);
}

public static char encodeChar (int i) {
	return ((char) i);
}
```

4) Hausaufgabe (Exception)

[1 Punkt]

Kopieren Sie Ihr Programm von Aufgabe 3) und ändern Sie es wie folgt ab. Das Programm soll nun nicht explizit prüfen, ob die Datei existiert, sondern stattdessen eine `Exception` abfangen, die beim Öffnen erzeugt wird, wenn die Datei nicht existiert. Geben Sie die in der `Exception` enthaltene Nachricht auf der `Console` aus und beenden Sie das Programm.

5) Präsenzaufgabe (Schreiben einer Textdatei)

Debuggen Sie das folgende Programm. Setzen Sie einen Breakpoint auf die markierte Zeile.

```
package ArbeitsblattB;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class B5 {

	public static void main(String[] args) {
		try {
```

```
                BufferedWriter out = new BufferedWriter(new FileWriter("test.txt"));
                out.write("Lorem ipsum dolor sit amet");
                out.flush();
                out.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

6) *Präsenzaufgabe (Encodings)*

Sie benötigen dieses Programm für die folgenden Hausaufgaben! Bitte sichern Sie Ihr Programm auf einem USB-Stick o.ä.

Schreiben Sie ein Programm, das die Zeichenfolge „Die Welt kostet 17 €“ mit einem anschließenden Zeilenumbruch in UTF-8-Kodierung in eine Datei schreibt. Lesen Sie die Datei anschließend byteweise ein und geben Sie die **Byte-Folge** auf dem Bildschirm aus. Geben Sie die Bytes in Dezimalschreibweise aus.

Wenn Sie alles richtig gemacht haben, gibt Ihr Programm folgendes aus:
68 105 101 32 87 101 108 116 32 107 111 115 116 101 116 32 49 55 32 226 130 172 13 10

Vergleichen Sie die Byte-Folge bitte ***genau*** mit der Ausgabe Ihres Programms.

7) *Hausaufgabe (Encodings)*

[1 Punkt]

Betrachten Sie die Ausgabe Ihres Programms aus Aufgabe 6).

1) Erläutern Sie das Ergebnis. Wieso sind gerade diese Bytes herausgekommen? Notieren Sie bitte detailliert in einer Tabelle der folgenden Form Ihre Antwort:

Byte	68	105	101	32	87	…		
Bedeutung	…	…	…	…	…	…		

2) Was ändert sich, wenn Sie da Encoding von UTF-8 auf UTF-16 ändern?

Unterstützung erhalten Sie hier:

- Java-API: http://docs.oracle.com/javase/7/docs/api/ `OutputStreamWriter`- bzw. `PrintWriter`-Konstruktor mit Encoding-Parameter
- http://de.wikipedia.org/wiki/UTF-8 (insb. bzgl. des Euro-Zeichens)
- http://en.wikipedia.org/wiki/Byte_Order_Mark

Sollten Sie Informationen über die Hexadezimal-Schreibweise benötigen, können Sie diese z. B. hier bekommen: http://de.wikipedia.org/wiki/Hexadezimal.

8) *Präsenzaufgabe (Debugging, String, Parsing)*

Ihre Aufgabe besteht darin, aus einer gegebenen Datei von Aktienfonds-Daten den Jahresanfangskurs (erster Tageskurs) des Jahres 2018 und den Jahresendkurs (letzter Tageskurs) des Jahres 2018 herauszufiltern. Die beiden Kurse sollen verglichen werden. Wenn der Kurs gestiegen ist, soll die Ausgabe „bull“, sonst „bear“ lauten.

Das Programm „DWS“, welches Sie von Ihrem Teamkollegen erhalten haben, versucht diese Aufgabe zu lösen, scheitert jedoch kläglich. Ihre Aufgabe ist nun, das Programm so weit zu korrigieren und zu vervollständigen, dass es das richtige tut. Außerdem sollen Sie das Programm robuster, wartbarer und benutzerfreundlicher machen (StyleGuide anwenden, Kommentare einfügen, Exceptions abfangen, ggf. Zwischen- oder Fehlermeldungen an den Benutzer, im Fehlerfall Zeilennummer ausgeben, Änderung des Suchjahres). Bedenken und dokumentieren Sie auch Ihre Annahmen und die Voraussetzungen, unter denen Ihr Programm funktioniert!

Tipp: Verwenden Sie bei Ihrer Arbeit die Möglichkeit, Haltepunkte zu setzen und sich Variablen-Inhalte anzeigen zu lassen. Vorgegebener Programmcode: DWS.zip (zum Download von der Kursseite im ILIAS). Darin ist auch die Datei mit den Fond-Daten enthalten.

9) Hausaufgabe (String, Parsing, Streams)

[3 Punkte]

Der Bibliothekar (vgl. Blatt 1, Aufgabe 11) wünscht sich eine Möglichkeit Medien, die in einem Bibtex ähnlichem Standard beschrieben sind, in den Zettelkasten importieren zu können. Ihre Aufgabe ist es eine Methode *parseBibTex* zu implementieren, die

- die Beschreibung eines Mediums als Eingabe in Form eines Strings erhält,
- die Eingabe auf Validität prüft (Klammerung korrekt, Felder vorhanden, etc.)
- den Typ des Mediums erkennt und ein entsprechendes Objekt erzeugt,
- die Attribute des Objekts mit den entsprechenden Daten füllt,
- und das erstellte Medium zurückgibt.

Die Methode soll dabei hinsichtlich der möglichen Eingaben robust ausgelegt sein und auf Probleme (Stichwort Exception) reagieren können. Überlegen Sie in welcher bzw. welchen Klassen die Methode am sinnvollsten realisiert werden kann.

Eingabeformate:

```
@book{author = {-}, title = {Duden 01. Die deutsche Rechtschreibung}, publisher = {Bibliogr
aphisches Institut, Mannheim}, year = 2004, isbn = {3-411-04013-0}}

@journal{title = {Der Spiegel}, issn = {0038-7452}, volume = 54, number = 6}

@cd{title = {1}, artist = {Die Beatles}, label = { Apple (Bea (EMI))}}

@elMed{title = {Hochschule Stralsund}, URL = {http://www.hochschule-stralsund.de}}
```

Hinweis: Bitte integrieren Sie Ihre Methode in Ihr Bibliotheksprogram (vgl. Blatt 1, Aufgabe 9).