Christian Bunse & Tilo Zülske
E-Mail: Vorname.Nachname@hochschule-stralsund.de
Büro: Haus 4/306c&331

Hochschule Stralsund

Aufgaben zur Lehrveranstaltung
Laborpraktikum Software
SMSB3300 / SMIB3300/ SKIB3300

Änderungs-datum	Änderung	Kommentar
09.09.2025	Initiale Fassung	
	•	
	•	

Aufgabenblatt #1

1. *Zum „Warmwerden“ (Online-Hilfe, Package)*
2. *Hausaufgabe (Zahlen, Arrays)*
3. *Hausaufgabe (Garbage Collector)*
4. *Präsenzaufgabe (Klassen, Objekte)*
5. *Präsenzaufgabe (Begriffe)*
6. *Präsenzaufgabe (Abstrakte Klassen, Polymorphie)*
7. *Hausaufgabe (Polymorphie)*
8. *Hausaufgabe (toString)*
9. *Hausaufgabe (Klassen, abstrakte Klassen, Properties, String-Operationen, Polymorphie)*

Präsenzaufgaben werden **im Labor** bearbeitet. Die Arbeit im Labor ist für Sie die Gelegenheit, Verständnisfragen so weit zu klären, dass Sie in der Lage sind, die nachfolgenden Hausaufgaben selbständig zu erledigen. Präsenzaufgaben werden nicht benotet, ihre Bearbeitung ist jedoch zum Verständnis wichtig. Präsenzaufgaben müssen von Ihnen bearbeitet werden, **bevor** Sie sich an die jeweils nachfolgenden Hausaufgaben machen.

Halten Sie sich also am besten einfach an die Reihenfolge der Aufgaben des Aufgabenblattes: zuerst A.1 bis A.5 zu Hause, dann im Labor A.6 bis A.8, anschließend A.9 bis A.11. Die Bearbeitung der Präsenzaufgaben weisen Sie direkt im Labor dem Betreuer nach (Vorführung). Falls Sie nicht fertig geworden sind, vereinbaren Sie mit dem Betreuer das weitere Vorgehen (z. B. Abgabe zusammen mit den Hausaufgaben).

Hausaufgaben werden (je nach individuellem Tempo) im Labor begonnen und zu Hause fortgeführt. Hausaufgaben müssen Sie zu einem vorgegebenen Termin (via MOODLE) abgeben. Abgaben erfolgen in elektronischer Form. Hausaufgaben sind Gegenstand des Testats und der Benotung. Die Hausaufgaben dieses Aufgabenblattes sind **bis 06.10.2025 (8:00 Uhr) via MOODLE abzugeben (siehe Zeitplan)**. Verspätete Abgaben oder Abgaben via E-Mail werden **nicht** berücksichtigt!

Hinweise bei Problemen mit der Ausführung von Programmen direkt vom USB-Stick.

Zum Arbeiten mit IntelliJ auf verschiedenen Rechnern empfiehlt es sich, den Workspace direkt auf einem USB-Stick abzuspeichern. Sie können einen lokal erstellten Workspace einfach über ihr Dateisystem (z.B. Microsoft Explorer oder Midnight Commander unter Linux) auf ihren USB-Stick und von dort auf einen zweiten Rechner kopieren. Sie können den Workspace auch direkt von Ihrem USB-Stick öffnen.

Sollten Sie Dateien aus einem anderen Workspace bzw. Projekt importieren wollen, so können Sie im Paket-Explorer mit der rechten Maustaste auf Ihr aktuelles Projekt klicken und dann den Menüpunkt „Importieren...“ wählen.

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

Aufgabenblatt #1

1. Zum „Warmwerden“ (Online-Hilfe, Package)

Machen Sie sich mit der Entwicklungsumgebung IntelliJ vertraut. Lesen Sie hierzu die unter Intellij Idea - Kurzanleitung (isolution.pro) verfügbare Kurzeinführung oder das Einführungsvideo unter https://youtu.be/VvMnlurITNY?feature=shared .

Wo können Sie Informationen nachschlagen? Wo finden Sie die Java-Sprachreferenz? Studieren Sie dort die Bedeutung des **package**-Schlüsselwortes, etc.

2. Hausaufgabe (Zahlen, Arrays)

[3 Punkte]

Szenario: Ein Team von Luft- und Raumfahrtingenieuren der Hochschule Stralsund entwickelt ein autonomes Drohnen-System, das für die Überwachung von landwirtschaftlichen Nutzflächen eingesetzt wird. Ihr Team ist für die Entwicklung der Software zur Missionsanalyse und -überwachung zuständig.

Sie erhalten kontinuierlich Daten von einem Schwarm von bis zu 50 Drohnen, die sich in einem 3D-Raum bewegen. Die gesammelten Daten umfassen Position, Geschwindigkeit, Batteriestatus und Missionsstatus jeder einzelnen Drohne zu jedem Zeitpunkt. Ihre Aufgabe ist es, eine Java-Anwendung zu entwickeln, die diese Daten verarbeitet, analysiert und kritische Zustände identifiziert.

Die Übung ist in drei Hauptteile unterteilt: Datenmodellierung, Datenanalyse und Performance-Optimierung.

Teilaufgabe 1: Datenmodellierung und -erzeugung [1 Punkt]

Entwickeln Sie eine Klasse namens DrohnenSchwarm, die die Daten des gesamten Schwarms verwalten kann.

1. **Datenstruktur:** Verwenden Sie ein zweidimensionales Array (`double[][]`) als Kern-Datenstruktur. Jede Zeile in diesem Array soll eine "Momentaufnahme" der Daten einer Drohne zu einem bestimmten Zeitpunkt repräsentieren. Die Spalten sollen folgende Daten enthalten:
    - `[0]`: Drohnen-ID (ganzzahlig, z.B. 1.0, 2.0, ...)
    - `[1]`: Zeitstempel (in Sekunden)
    - `[2]`: X-Koordinate
    - `[3]`: Y-Koordinate
    - `[4]`: Z-Koordinate
    - `[5]`: Batteriestand (Prozentsatz, 0.0 - 100.0)
    - `[6]`: Status-Code (z.B. 1.0 für "in Mission", 2.0 für "Rückflug zur Basis", 3.0 für "Problem erkannt")
2. **Datengenerierung:** Implementieren Sie eine Methode `generiereSimulationsdaten(int anzahlDrohnen, int simulationsdauer)`, die das `double[][]` Array dynamisch initialisiert und mit simulierten Daten füllt. Die Simulationsdauer wird in Zeitstempeln (z.B. 1000) gemessen. Die Methode soll:
    - Ein Array erstellen, das groß genug ist, um die Daten aller Drohnen über die gesamte Simulationsdauer zu speichern.
    - Für jede Drohne und jeden Zeitstempel realistische, zufällige Werte für Position, Batteriestand und Status generieren. Achten Sie darauf, dass sich die Werte (insbesondere die Position) über die Zeit fließend ändern.
    - Beispielwerte:
        - Positionen: innerhalb eines 1000x1000x1000 m³ Volumens.
        - Batteriestand: sinkt kontinuierlich, z.B. um 0.1% bis 0.5% pro Zeitstempel.

Teilaufgabe 2: Datenanalyse [1 Punkt]

Implementieren Sie in der Klasse `DrohnenSchwarm` Methoden, um die generierten Daten zu analysieren und folgende Erkenntnisse zu gewinnen:

1. **Kollisionserkennung:** Schreiben Sie eine Methode `findeKollisionen(double kritischerAbstand)`, die das gesamte Daten-Array durchläuft. Die Methode soll für jeden Zeitstempel Paare von Drohnen identifizieren, deren euklidischer Abstand ($\sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2}$) unter dem `kritischenAbstand` liegt. Die Methode soll eine Liste von Zeichenketten zurückgeben, die die kollidierenden Drohnenpaare und den Zeitpunkt protokollieren (z.B. "Kollision zwischen Drohne 5 und Drohne 12 bei Zeitstempel 450").
2. **Batterieanalyse:** Implementieren Sie eine Methode `meldeNiedrigeBatterie(double schwellenwert)`, die alle Drohnen identifiziert, deren Batteriestand unter dem angegebenen `schwellenwert` (z.B. 20%) liegt. Die Methode soll eine Liste der betroffenen Drohnen-IDs und ihrer letzten bekannten Position zurückgeben.
3. **Schwarm-Schwerpunkt:** Erstellen Sie eine Methode `berechneSchwarmSchwerpunkt()`, die für jeden Zeitstempel den durchschnittlichen geometrischen Mittelpunkt (Schwerpunkt) aller Drohnen berechnet und diese (X, Y, Z)-Koordinaten in einem separaten zweidimensionalen Array speichert.

Teilaufgabe 3: Performance-Optimierung und Berichterstattung [1 Punkt]

 Performance-Messung: Messen Sie die Ausführungszeit Ihrer `findeKollisionen`-Methode für einen Schwarm von 50 Drohnen über 1000 Zeitstempel. Protokollieren Sie die Laufzeit in Millisekunden.

 Berichtserstellung: Erstellen Sie eine Methode erstelleBericht(), die alle Ergebnisse der Analyse (Kollisionen, niedrige Batteriestände, den berechneten Schwarm-Schwerpunkt) in einem lesbaren Format auf der Konsole ausgibt.

 Bonus-Aufgabe (Optimierung [0.5 Punkte]): Die findeKollisionen-Methode aus Teil 2 hat eine Zeitkomplexität von $O(N^{2})$ pro Zeitstempel, was für große Schwärme ineffizient ist.

- Beschreiben Sie, warum diese Methode ineffizient ist.
- Skizzieren Sie einen alternativen, optimierten Algorithmus zur Kollisionserkennung. Erklären Sie, welche alternative Datenstruktur (z.B. ein räumliches Gitter oder ein Quadtree/Octree) anstelle des einfachen Arrays zur Beschleunigung des Prozesses genutzt werden könnte.

3. Hausaufgabe (Garbage Collector)

[1 Punkt]

Szenario: Sie sind Softwareentwickler in einem Unternehmen, das eine Echtzeit-Datenverarbeitungs-Engine für die Überwachung von IoT-Sensoren entwickelt. Die Engine ist für einen Dauerbetrieb von 24/7 ausgelegt und muss eine hohe Verfügbarkeit und Performance gewährleisten. Ein zentrales Problem ist die Speicherverwaltung, da die Sensordaten in hohem Tempo eingehen und verarbeitet werden, was zu einer schnellen Allokation und Deallokation von Objekten im Heap führt. Sie verwenden Java, das eine automatische Speicherverwaltung durch den Garbage Collector (GC) bietet.

Ihre Aufgabe ist es, eine kritische Code-Sequenz zu analysieren und die Auswirkungen auf die Speichernutzung sowie das Verhalten des Garbage Collectors zu erklären.

Aufgabe 1: Code-Analyse und Objekt-Lebenszyklus

Gegeben ist die folgende Methode:

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

public class SensorDataProcessor {

   private static final int NUM_SENSORS = 50000;
   private static final int DATA_POINTS_PER_RUN = 1000;

   public void processSensorData() {
      for (int i = 0; i < DATA_POINTS_PER_RUN; i++) {
         List<String> dataChunk = new ArrayList<>();
         for (int j = 0; j < NUM_SENSORS; j++) {
            dataChunk.add(new UUID(0L, j).toString());
         }
         // dataChunk wird in der nächsten Zeile verarbeitet
         analyzeAndStore(dataChunk);
      }
      // ... weiterer Code ...
   }

   private void analyzeAndStore(List<String> data) {
      // Diese Methode simuliert die Verarbeitung und Speicherung der Daten
      // In dieser Methode werden keine Referenzen auf die übergebenen Daten
      // gespeichert, die über den Methodenaufruf hinausgehen.
```

```
        System.out.println("Verarbeite Daten-Chunk mit " + data.size() + " Elementen.");
    }

    public static void main(String[] args) {
        SensorDataProcessor processor = new SensorDataProcessor();
        processor.processSensorData();

        // Was passiert mit den Objekten, die innerhalb von processSensorData erstellt wurden?
        System.gc(); // Freiwilliger Aufruf des Garbage Collectors
    }
}
```

Fragen zur Analyse:

1. Beschreiben Sie den Lebenszyklus der Objekte, die in der Methode processSensorData() erstellt werden.
2. Welche Objekte werden bei jedem Durchlauf der inneren for-Schleife erstellt? Wann werden diese Objekte für den Garbage Collector (GC) freigegeben?
3. Wann werden die List<String>-Objekte (dataChunk) für den GC freigegeben?
4. Erklären Sie, was System.gc() bewirkt. Ist dies in einer produktiven Anwendung empfehlenswert? Begründen Sie Ihre Antwort.

4. Präsenzaufgabe (Klassen, Objekte)

Implementieren Sie eine Klasse „Kraftfahrzeug“ mit einem Attribut „modell“ vom Typ Zeichenkette und einem weiteren Attribut „verbrauchProKilometer“ zur Speicherung einer Fließkommazahl.

Implementieren Sie folgenden Konstruktor:

```
public Kraftfahrzeug(String _modell, double _verbrauchProKilometer)
{
    // Ihr Code hier
}
```

Implementieren Sie sichtbare Zugriffsoperationen für den Zugriff auf das Feld „modell“.

Implementieren Sie eine Operation, die den Verbrauch für eine als Parameter übergebene Strecke berechnet.

```
public double verbrauch(int kilometer)
{
    // Ihr Code hier
}
```

Schreiben Sie folgendes Hauptprogramm:

```
public static void main(String[] args) {
        Kraftfahrzeug[] autoArr = new Kraftfahrzeug[2];
        //Der Focus verbraucht 6,5 Liter auf 100 km:
        autoArr[0] = new Kraftfahrzeug("Focus", 0.065);
        //Der Tesla verbraucht 0 Liter auf 100 km:
        autoArr[1] = new Kraftfahrzeug ("Tesla", 0.00);

        int km = 400;
        System.out.printf("Verbrauch auf %d km:%n", km);
```

```
        for (int i = 0; i < autoArr.length; i++) {
                System.out.printf("%s: %.0f Liter %n",
                                autoArr[i].getModell(),
                                autoArr[i].verbrauch(km));
        }
}
```

5. Präsenzaufgabe (Begriffe)

Verbinden Sie die Elemente Ihres Programms auf der linken Seite mit je einer Linie mit einem Begriff auf der rechten Seite. Achten Sie genau auf die Groß/Kleinschreibung der linksstehenden Elemente.

1	viereckArr	Klasse	**A**
2	Viereck	Sichtbarkeit	**B**
3	viereckArr[0]	Zugriffsoperation	**C**
4	flaeche	Kommentar	**D**
5	getLaengeA	Array-Index	**E**
6	i	Attribut	**F**
7	laengeA	Methode	**G**
8	// Fuellen des Array	Referenz auf ein Objekt	**H**
9	_laengeA	Parameter	**I**
10	public	lokale Variable	**J**
11	[1]	Array	**K**

6. *Präsenzaufgabe (Abstrakte Klassen, Polymorphie)*

Implementieren Sie in Ihrer Klasse „Kraftfahrzeug“ eine Methode „fahre“ ohne Parameter und ohne Rückgabewert. Diese Operation soll einfach „Gas geben“ auf der Konsole ausgeben.

Implementieren Sie eine Basisklasse „Fahrzeug“ von Kraftfahrzeug. Auch in der Basisklasse soll es eine Methode „fahre“ geben. Diesmal jedoch als abstrakte Methode.

Implementieren Sie eine Klasse „Motorrad“ als zweite Subklasse von Fahrzeug. Implementieren Sie in Motorrad ebenfalls eine Methode „fahre“. Diese soll den Text „Antreten“ auf der Konsole ausgeben. Erweitern Sie Ihr Hauptprogramm wie folgt:

```
Fahrzeug[] fahrzeugArr = new Fahrzeug[2];
fahrzeugArr[0] = new Kraftfahrzeug ("Golf", 0.065);
fahrzeugArr[1] = new Motorrad();
for (int j = 0; j < 2; j++)
{
   fahrzeugArr[j].fahre();
}
```

7. *Hausaufgabe (Polymorphie)*

[1 Punkt]

Implementieren Sie in der Klasse Motorrad ein Feld „längeInZoll“ und sichtbare Zugriffsoperationen (Getter/Setter). Ergänzen Sie ebenfalls einen geeigneten Konstruktor, der es Ihnen erlaubt, die Größe bei der Instanziierung eines Motorrades festzulegen.

Implementieren Sie in der Klasse Fahrzeug ein Feld „baujahr“ und entsprechende sichtbare Zugriffsoperationen. Ergänzen Sie ebenfalls geeignete Konstruktoren in den drei Klassen. Rufen Sie aus dem Konstruktor der Unterklasse den Konstruktor der Basisklasse auf (Schlüsselwort **super**). Folgender Programmcode soll möglich sein, um einen Focus mit Baujahr 1995 bzw. ein Motorrad der Länge 280“, Baujahr 1974 anzulegen:

```
Fahrzeug golf= new Kraftfahrzeug ("Golf", 0.065, 1995);
Motorrad gurke= new Motorrad(280, 1974);
```

8. *Hausaufgabe (toString)*

[1 Punkt]
Wenn Sie diese Aufgabe bearbeiten, können Sie sie in Gestalt eines Programms zusammen für beide Aufgaben 8 und 9 abgeben.

Wenn Sie diese Aufgabe bearbeiten, können Sie sie in Gestalt eines Programms zusammen für beide Aufgaben 0 und 0 abgeben.

Implementieren Sie die Methode „toString“ für alle drei Klassen Fahrzeug, Kraftfahrzeug und Motorrad. Diese Methode soll alle Eigenschaften in Form eines Strings zurückliefern, z. B. so:

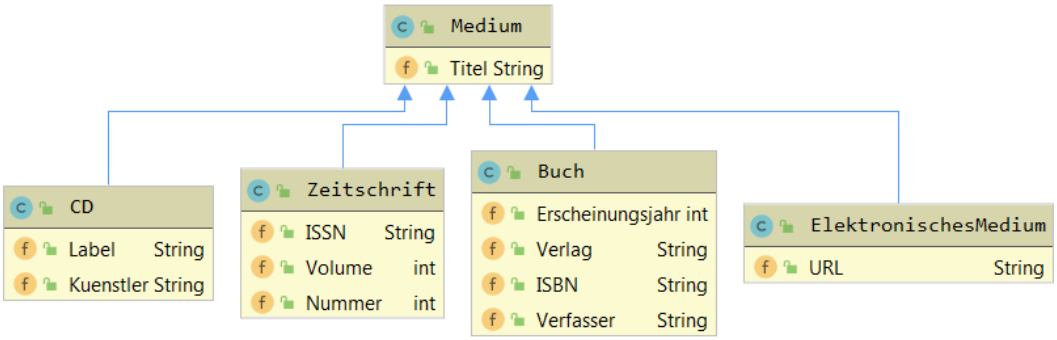
- für Fahrzeug: „Bj. 1975“.
- für Kraftfahrzeug: „Bj. 2012, Tesla, 0,00 l/km“,
- für Motorrad: „Bj. 1983, 280 Zoll“,

Verwenden Sie Operationen des **StringBuilder**. Informieren Sie sich über Möglichkeiten der Formatierung „Java ist auch eine Insel“ von Christian Ullenboom.

9. *Hausaufgabe (Klassen, abstrakte Klassen, Properties, String-Operationen, Polymorphie)*

Diese Aufgabe wird uns in den kommenden Wochen noch des Öfteren beschäftigen. Bitte bearbeiten Sie diese Aufgabe unbedingt.
[3 Punkte]

Der Archivar einer kleinen Hochschule möchte seinen Zettelkasten digitalisieren. In seinem Zettelkasten finden sich Zettel über diverse Medien: Bücher, Zeitschriften, CDs bzw. Online-Medien. Jedes Medium hat einen Titel. Unterschiede bestehen in weiteren Daten (Künstler, Verfasser, ISBN-Nummer, Verlag, etc.). Er modelliert seinen Anwendungsbereich wie folgt:



Medium ist eine abstrakte Klasse, von der die vier unteren Klassen das Attribut Titel erben.

Implementieren Sie die fünf Klassen. Schützen Sie den Zugriff auf die Attribute und realisieren Sie Getter- und Setter-Methoden.

Für Bücher kennt der Bibliothekar einen Algorithmus, mit dem er die Korrektheit der ISBN prüfen kann. Implementieren Sie in der Setter-Methode für das Attribut zur ISBN die Prüfroutine, die wir am Ende dieser Aufgabe abdrucken. Im Fehlerfall geben Sie eine Meldung auf der Konsole aus. Ähnlich soll auch mit der URL für Elektronische Medien verfahren werden. Nutzen Sie hierfür die ebenfalls angegebene Prüfroutine.

Der Bibliothekar will für jeden Zettel eine Zeichenkette erstellen. Wir nennen diese Zeichenkette eine „Repräsentation“ des Mediums. Die Repräsentation des Mediums „Duden“ soll z. B. wie folgt aussehen:

> Titel: Duden 01. Die deutsche Rechtschreibung
> Erscheinungsjahr: 2004
> Verlag: Bibliographisches Institut, Mannheim
> ISBN: 3-411-04013-0
> Verfasser: -

Implementieren Sie eine Methode namens „**calculateRepresentation**“, die als Rückgabewert die Repräsentation des Mediums in Gestalt einer Zeichenkette liefert. Implementieren Sie diese Methode in allen vier Subklassen. Verwenden Sie dabei den **StringBuilder**.

Schreiben Sie ein Hauptprogramm (Funktion „Main“) in einer neuen Klasse namens „Bibliothek“, dass die folgenden vier Medien instanziiert und in einem Array der Länge 4 speichert.

- „Duden 01. Die deutsche Rechtschreibung“. Erscheinungsjahr: 2004. Verlag: Bibliographisches Institut, Mannheim. ISBN: 3-411-04013-0. Verfasser: -
- “1”, Label: Apple (Bea (EMI)), Künstler: The Beatles
- „Der Spiegel“. ISSN 0038-7452. Volume 54. Nummer 6.
- „Hochschule Stralsund“: http://www.hochschule-stralsund.de

Anschließend implementieren Sie eine **for**-Schleife, die die einzelnen Repräsentationen auf der Konsole ausgibt

Erstellen Sie mit JavaDoc in einem neuen Unterverzeichnis „doc“ (neben dem „out“-oder „bin“ Verzeichnis des Projektes) eine HTML-Version der Programmkommentare und geben diese mit ab.

Und hier nun die versprochene Prüfroutinen für die ISBN-Nummer und die URL:

```
public static boolean checkISBN10(int[] isbn) {
   int sum = 0;
   for (int i = 1; i <= isbn.length; i++) {
      sum += i * isbn[i - 1];
   }
   if (sum % 11 == 0) {
      return true;
   } else {

      return false;
   }

}

public static boolean checkISBN13(int[] isbn) {
   int sum = 0;
   for (int i = 1; i < isbn.length; i++) {
      if (i % 2 == 0) {
         sum += isbn[i - 1] * 3;
      } else {
         sum += isbn[i - 1];
      }
   }

   int lastDigit = sum % 10;

   int check = (10 - lastDigit) % 10;

   if (isbn[isbn.length - 1] == check) {
      return true;
   } else {
      return false;
   }
}

public static void main(String[] args) {
   int[] isbn10 = new int[] { 3, 8, 6, 6, 8, 0, 1, 9, 2 };
   System.out.println(checkISBN10(isbn10));
   int[] isbn13 = new int[] { 9, 7, 8, 3, 7, 6, 5, 7, 2, 7, 8, 1, 8 };
   System.out.println(checkISBN13(isbn13));
}
```

```
public static boolean checkURL(String urlString)
{
   try
   {
      URL url = new URL(urlString);
      url.toURI();
      return true;
   } catch (Exception exception)
   {
      return false;
   }
}
```