HHU Düsseldorf
Institut für Informatik
Prof. Dr. Michael Schöttner

Daten des Prüflings

Matrikelnummer: _____

Nachname: _____

Vorname: _____

2. Klausur

Programmierung WS 24/25
1. April 2025

- Prüfen Sie bitte zuerst, ob die Klausur alle Seiten enthält.
- Sie erhalten von uns leere Blätter. Sollten Sie auf diesen Blättern für die Korrektur relevante Teile Ihrer Lösung notieren, markieren Sie dies deutlich an der entsprechenden Aufgabe und auf dem zusätzlichen Blatt. Schreiben Sie auf alle zusätzlichen Blätter Ihren Namen. Geben Sie unten auf dem Deckblatt an, wie viele zusätzliche Blätter Sie zur Korrektur abgeben.
- Fachbegriffe werden wie in der Vorlesung definiert verwendet. Die Aufgaben beziehen sich auf die in der Vorlesung vorgestellte Java-Version 21. Programmcode muss in Java geschrieben werden.
- Antworten dürfen auf Deutsch oder Englisch gegeben werden. Sofern keine ausformulierten Sätze verlangt sind, reichen nachvollziehbare Stichworte als Antwort.
- Erlaubte Hilfsmittel: eine beidseitig beschriebene oder bedruckte A4-Seite, Wörterbuch (Wörterbuch muss vor Klausurbeginn der Aufsicht zur Kontrolle vorgelegt werden.)
- Schalten Sie technische Geräte aus. Täuschungsversuche führen zum sofortigen Ausschluss von der Klausur. Die Klausur wird dann als nicht bestanden gewertet.
- Schreiben Sie **nicht** mit radierbaren Stiften und auch **nicht** mit rot!

Mit meiner Unterschrift bestätigen ich, die Klausur auf Vollständigkeit überprüft und die Hinweise gelesen zu haben: _____

☐ Falls ich diese Klausur bestehe, möchte ich **nicht** automatisch zum Programmierpraktikum 1 im Sommersemester angemeldet werden. (nur Studiengang Informatik)

zur Bewertung abgegebene, lose Blätter: _____ **Viel Erfolg!**

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	Σ
Punkte	10	26	3	6	11	34	90
Erreicht							

Aufgabe 1

_____ / 10 Punkte

__/6 (a) Gegeben seien die folgenden java-Dateien:

IntFunction.java

```java
public interface IntFunction<T> {
    T calc(int x);
}
```

Square.java

```java
public class Square implements IntFunction<Integer> {
    Integer calc(int x) {
        return x * x;
    }
}
```

Test.java

```java
public class Test {
    public static void main(String[] args) {
        if(args.length == 0) {
            System.out.println("arg missing");
            return 1;
        }
        int x = Integer.parseInt(args(0));
        IntFunction<Integer> sq = new Square();
        System.out.println(sq.calc(x));
    }
}
```

Beim Compilieren der Klassen gibt es folgende Fehlermeldungen:

```
% javac Test.java
Test.java:5: error: incompatible types: unexpected return value
            return 1;
                   ^
Test.java:7: error: cannot find symbol
        int x = Integer.parseInt(args(0));
                                      ^
  symbol:   method args(int)
  location: class Test
./Square.java:2: error: calc(int) in Square cannot implement calc(int) in
   IntFunction
    Integer calc(int x) {
            ^
  attempting to assign weaker access privileges; was public
  where T is a type-variable:
    T extends Object declared in interface IntFunction
3 errors
```

Geben Sie auf der nächsten Seite an, in welchen Dateien und Zeilen die **Ursachen** für die Fehler sind, beschreiben Sie die Fehlerursachen jeweils kurz (max. 1 Satz) und geben Sie die korrigierte Codezeile **vollständig** an, sodass der Code das tut, was bei der Programmierung wahrscheinlich vorgesehen war; falls eine Zeile entfernt werden muss, tragen Sie `-leer-` als Korrektur ein. Geben Sie keine Folgefehler an, die durch Korrektur eines vorherigen Fehlers behoben werden.

Datei, Zeilennummer: _____

Fehlerursache: _____

korrigierte Codezeile: _____

Datei, Zeilennummer: _____

Fehlerursache: _____

korrigierte Codezeile: _____

Datei, Zeilennummer: _____

Fehlerursache: _____

korrigierte Codezeile: _____

Datei, Zeilennummer: _____

Fehlerursache: _____

korrigierte Codezeile: _____

__/4 (b) Im Terminal ist die Ordnerstruktur im aktuellen Verzeichnis dargestellt:

```
/home/jan/projects % tree
.
|-- src
    |-- main
        |-- progra
            |-- Hello.java
            |-- Greeter.java
/home/jan/projects %
```

Sie wollen die Klasse mit dem voll-qualifizierten Namen `progra.Hello` im Ordner `/home/jan/projects/src/main/progra` kompilieren. Die Klasse `progra.Hello` benutzt die Klasse `progra.Greeter`, deren Code sich im selben Ordner befindet wie der Code von `progra.Hello`.
Geben Sie den Classpath an: _____

Vervollständigen Sie die Befehle, mit denen `progra.Hello` kompiliert und die enthaltene `main`-Methode ausgeführt werden kann. `X` steht dabei für den Classpath, den Sie oben angegeben haben, sodass sie ihn nicht nochmal abschreiben müssen. Wechseln Sie **nicht** das aktuelle Verzeichnis.

`javac` _____ `X` _____

`java` _____ `X` _____

Aufgabe 2

_____ / 26 Punkte

Ein regulärer Kettenbruch ist ein Bruch der Form $a + \cfrac{1}{b+\cfrac{1}{c+\cfrac{1}{\ddots}}}$,

wobei a, b usw. ganze Zahlen sind.

Reguläre Kettenbrüche werden abgekürzt auch als $[a;b,c,\dots]$ geschrieben.

Hinweise:

- `"1=28=-4".split("=")` gibt ein String-Array mit den drei Elementen `"1"`, `"28"` und `"-4"` zurück.
- `String` besitzt die Objektmethoden `startsWith(String)`, `endsWith(String)`, `contains(String)` und `length()`.
- `"abcdef".substring(2,4)` gibt den neuen String `"cd"` zurück; die Parameter sind Indices.
- `"abcb".replace("b", "x")` gibt den neuen String `"axcx"` zurück.
- Die Parse-Methode der Klasse `Integer` wirft im Fehlerfall eine `NumberFormatException`.

___/1 (a) Kreuzen Sie den Ausdruck an, der denselben Wert wie der reguläre Kettenbruch [2;3,4] hat:

☐ $2 + \frac{1}{3+\frac{1}{4}}$ ☐ $0 + \frac{1}{3+\frac{1}{4}}$ ☐ $4 + \frac{1}{3+\frac{1}{2}}$

___/13 (b) Schreiben Sie eine **private Klassenmethode** `int[] parse(String)`, die einen beliebig langen Kettenbruch-String der Form `[2;3,4]` entgegennimmt und die enthaltenen Zahlen als Array zurückgibt (hier also 2, 3, 4). Bei folgenden falschen Eingabe-Strings soll eine `java.lang.IllegalArgumentException` geworfen werden:

- Das erste oder letzte Zeichen sind nicht `[` bzw. `]`.
- Der übergebene String hat weniger als 3 Zeichen.
- Die Zeichen zwischen den `[`, `;`, `,` und `]` lassen sich nicht als ganze Zahl verarbeiten.

Abgesehen von diesen Fehlerfällen dürfen Sie davon ausgehen, dass der Eingabe-String korrekt ist, also z. B. höchstens ein `;` enthält.

___/5 (c) Schreiben Sie eine **private Klassenmethode** `double evaluate(int[])`, die die Zahlen aus der Kettenbruchdarstellung (z. B. 2, 3, 4) übergeben bekommt und den Wert des regulären Kettenbruchs zurückgibt. Sie dürfen davon ausgehen, dass immer ein Array-Objekt übergeben wird, das mindestens die Länge 1 hat.

___/7 (d) Schreiben Sie eine `main`-Methode. Das erste Argument soll als Kettenbruch-String interpretiert werden und der Wert des Kettenbruchs auf der Standardausgabe ausgegeben werden. Falls nicht **genau ein** Argument übergeben wird oder das Argument **nicht** als **gültiger** Kettenbruch (gemäß Aufgabenteil (b)) interpretiert werden kann, soll sich das Programm mit der Ausgabe `err7999` beenden, ohne dass eine Exception-Meldung ausgegeben wird.

```
% java Kettenbruch "[2;zwei]"
err7999
% java Kettenbruch "21"
err7999
% java Kettenbruch "[2;1]"
3.0
% java Kettenbruch "[1;2,2,2,2,2]"
1.4142857142857144
```

Kettenbruch.java Java

```
public class Kettenbruch {
```

Kettenbruch.java (Fortsetzung) Java

```
}
```

Aufgabe 3

_____ / 3 Punkte

Geben Sie für die beiden folgenden Codeausschnitte jeweils an, ob die idiomatische Kontrollstruktur verwendet wird. Falls die Kontrollstruktur nicht idiomatisch ist, geben Sie idiomatischen Code mit gleicher Semantik an.

Vorgabe — Java

```java
public static void printEveryOtherWord(String[] words) {
    int i = 0;
    for(String word: words) {
        if(i % 2 == 0) {
            System.out.println(word);
        }
        i++;
    }
}
```

☐ ist idiomatisch

☐ ist nicht idiomatisch; folgender Code wäre idiomatisch:

ggf. idiomatischen Code angeben — Java

```java
public static void printEveryOtherWord(String[] words) {









}
```

Vorgabe — Java

```java
Scanner stdin = new Scanner(System.in);
while(stdin.hasNext()) {
    System.out.println(stdin.next().toUpperCase());
}
```

☐ ist idiomatisch

☐ ist nicht idiomatisch; folgender Code wäre idiomatisch:

ggf. idiomatischen Code angeben — Java

```java
Scanner stdin = new Scanner(System.in);
```

Aufgabe 4

_____ / 6 Punkte

Gegeben sei die folgende Klasse `LinkedList`, die eine einfach verkettete Liste implementiert, in der Strings gespeichert werden können. Es gibt die folgenden fertigen Methoden, die Sie in Ihrem Code aufrufen können:

LinkedList.java — Java

```
public class LinkedList {
    private class Node {
        private String data;
        private Node next;

        private Node(String data, Node next) {
            this.data = data;
            this.next = next;
        }
    }

    private Node head = null;

    public void add(String element) {
        // fügt den String element am Ende der Liste ein
        // ... (Code nicht abgedruckt)
    }

    public int length() {
        // gibt die Anzahl der Strings in der Liste zurück
        // ... (Code nicht abgedruckt)
    }

    public String get(int index) {
        // gibt den String am gegebenen Index zurück
        // ... (Code nicht abgedruckt)
    }
```

Implementieren Sie für `LinkedList` eine öffentliche Methode `toArray`, die die Elemente der Liste als String-Array zurückgibt; die Reihenfolge im Array muss der Reihenfolge in der Liste entsprechen. Folgendes Beispiel zeigt, wie `toArray` funktionieren soll:

Beispiel — Java

```
LinkedList daten = new LinkedList();
daten.add("a");
daten.add("foo");
daten.add("42");
// "daten" enthält jetzt die Werte "a", "foo" und "42" (in dieser Reihenfolge)

String[] datenArray = daten.toArray();
System.out.println(datenArray[0]);     // "a"
System.out.println(datenArray.length); // 3
```

LinkedList.java (Fortsetzung) — Java

LinkedList.java (Fortsetzung) Java

```
}
```

Aufgabe 5

_____ / 11 Punkte

Hinweis: Für diese Aufgabe müssen Sie nicht *wissen, wie Matrizen multipliziert werden.*

Wir haben angefangen eine Klasse `Matrix` zu schreiben. Den bereits fertigen Code finden Sie **auf dem letzten Blatt** der Klausur. Dieses Blatt dürfen Sie abtrennen.

Die Klasse besitzt eine Methode `multiply` zum Multiplizieren zweier Matrizen. Zwei Matrizen können nur dann miteinander multipliziert werden, wenn die Spaltenzahl der ersten Matrix gleich der Zeilenzahl der zweiten Matrix ist – anders ausgedrückt: Eine $n \times m$-Matrix kann nur mit einer $m \times l$-Matrix multipliziert werden.

(a) Erweitern Sie die Klasse `Matrix` so, dass der folgende Beispielcode die angegebene Ausgabe erzeugt:

Beispielcode — Java

```
int[][] werte = {{1,2,3},{4,5,6}};
Matrix m = new Matrix(werte);
System.out.println(m);
```

```
1 2 3
4 5 6
```

Zur Erinnerung: Der Character `'\n'` *steht für einen Zeilenumbruch.*

___/1 i. Welche **Methode** welcher **Klasse** müssen Sie überschreiben?

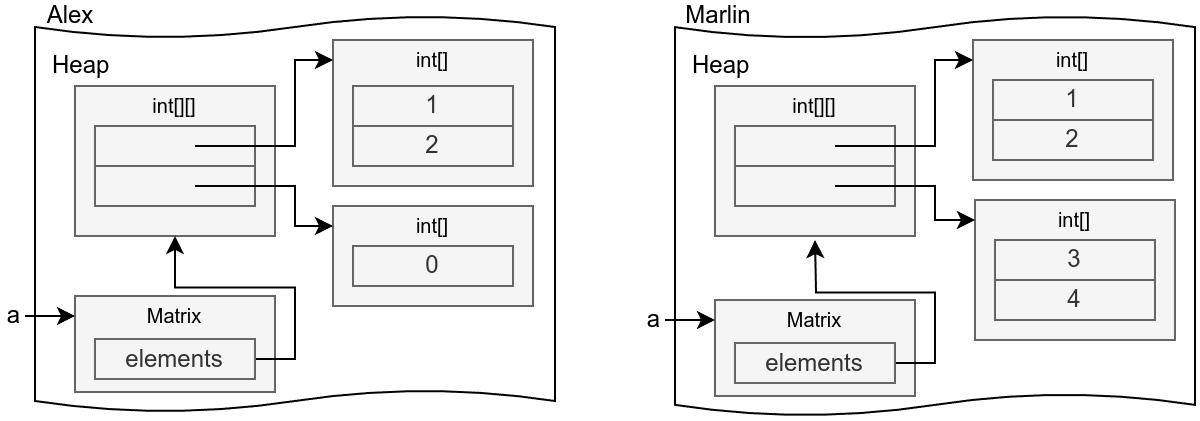
___/5 ii. Implementieren Sie diese Methode. Die Methode soll für beliebige Matrixen korrekt funktionieren, also nicht nur mit den oben angegebenen Beispielzahlen.

Matrix.java (Fortsetzung) — Java

___/5 (b) Nehmen Sie an, dass Aufgabenteil (a) korrekt umgesetzt ist. Betrachten Sie folgendes, compilierendes Codefragment:

```
1 int[][] values = {{1,2},{3,4}};
2 Matrix a = new Matrix(values);
3 Matrix b = new Matrix(new int[][]{{5},{6}});
4 Matrix c = a.multiply(b); // 2x2-Matrix mit 2x1-Matrix multiplizieren
5 values[1] = new int[]{0};
6 Matrix d = a.multiply(b);
7 System.out.println(d);
```

Alex und Marlin sind sich uneinig, wie das Matrix-Objekt (a) nach der Ausführung dieses Codes aussieht. Sie haben folgende Bilder gemalt:



Kreuzen Sie an, welches Bild korrekt ist: ☐ Alex (links) ☐ Marlin (rechts)

Begründen Sie Ihre Antwort; gehen Sie darauf ein, warum `a.elements` die dargestellten Werte hat und beziehen Sie sich auf relevante **Referenzen** und **Heap**-Werte:

Kreuzen Sie an, was passiert, wenn das Codefragment ausgeführt wird:

- ☐ In Zeile 4 gibt es eine Exception. – **Begründen** Sie, **welche** Exception es gibt und in welcher **Codezeile** in `Matrix.java` sie ausgelöst wird.
- ☐ In Zeile 6 gibt es eine Exception. – **Begründen** Sie, **welche** Exception es gibt und in welcher **Codezeile** in `Matrix.java` sie ausgelöst wird.
- ☐ In Zeile 7 wird Text auf der Standardausgabe ausgegeben. – Geben Sie an, **was** berechnet wird (sinngemäß, konkrete Ergebnisse müssen nicht angegeben werden) und welche **Methode** in `Matrix.java` für die Ausgabe aufgerufen wird.

Aufgabe 6

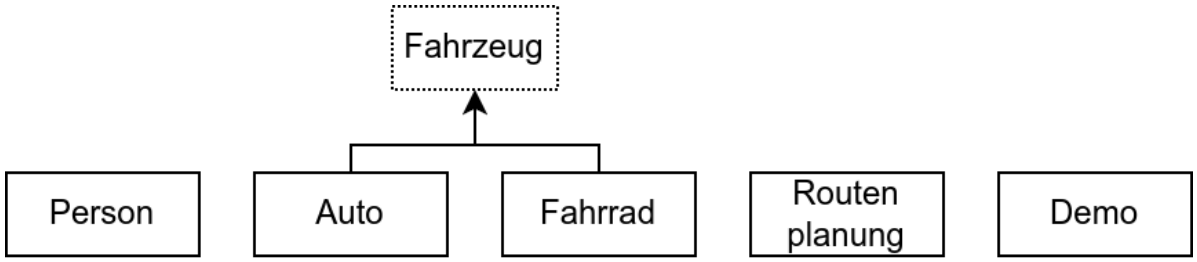
_____ / 34 Punkte

Wir arbeiten für einen Dienstleister, der Software für eine Firma entwickeln soll, die verschiedene Fahrzeuge vermietet. In dieser Aufgabe sollen Sie Interfaces, Klassen und Methoden für diese Vermietungs-Firma programmieren.

Hinweise:

- Sie dürfen alle Variablennamen frei wählen.
- Wählen Sie sinnvolle Datentypen für Variablen und Rückgabetypen, wenn es keine genaue Vorgabe gibt.
- Alle Instanz- und Klassenvariablen müssen **privat** sein.
- Wenn kein Konstruktorverhalten vorgeschrieben ist, reicht der Default-Konstruktor.
- Das genaue Format von Textausgaben ist Ihnen überlassen.
- Innerhalb dieser Aufgabe müssen Sie keine Exceptions abfangen. Sie müssen nur dann Parameter validieren, wenn dies verlangt ist.
- Lesen Sie sich die Aufgabenstellung vor Beginn der Implementierung **vollständig** durch, um einen besseren Überblick über das Gesamtbild zu erhalten.
- Gehen Sie davon aus, dass alle in dieser Aufgabe von Ihnen zu schreibenden Klassen im Default-Package liegen.

Innerhalb dieser Aufgabe sollen Sie die Klassen `LocalDate` und `Period` benutzen, die im JDK enthalten sind. Informationen zu diesen Klassen finden **auf dem letzten Blatt** der Klausur; dieses Blatt dürfen Sie abtrennen.



__/$9\frac{1}{2}$ (a) Schreiben Sie (auf der nächsten Seite) eine nicht abstrakte, **öffentliche** Klasse `Person`, die abspeichert, wann eine Person Geburtstag hat und ob sie einen Führerschein hat. Die Signatur des öffentlichen Konstruktors soll `Person(LocalDate, boolean)` lauten.

Schreiben Sie eine öffentliche Methode `int getAlter()`, die mit Hilfe von `Period` das aktuelle Alter der Person in Jahren ausrechnet und zurückgibt.

Schreiben Sie eine öffentliche Methode `boolean hatFuehrerschein()`, die zurückgibt, ob die Person einen Führerschein hat.

Hinweis: Bedenken Sie, dass `Person` **nicht** im Package `java.time` liegt.

Person.java Java

___/$2\frac{1}{2}$ (b) Schreiben Sie ein **öffentliches** Interface `Fahrzeug`, das zwei Methoden vorschreibt:

- `kannMieten(Person)`, die später angibt, ob eine Person ein bestimmtes Fahrzeug mieten darf
- `geschwindigkeit()`, die später die Durchschnitts-Geschwindigkeit des Fahrzeugs im Stadt-Verkehr in km/h zurückgibt

Fahrzeug.java — Java

___/9 (c) Schreiben Sie zwei nicht abstrakte, **öffentliche** Klassen `Auto` und (auf der nächsten Seite) `Fahrrad`, die das Interface `Fahrzeug` implementieren.

Eine Person kann nur dann ein Auto mieten, wenn sie mindestens 18 Jahre alt ist und einen Führerschein hat. Die Durchschnittsgeschwindigkeit eines Autos ist 40 km/h.

Eine Person kann nur dann ein Fahrrad mieten, wenn sie mindestens 14 Jahre alt ist. Die Durchschnittsgeschwindigkeit eines Fahrrads ist 20 km/h.

Auto.java — Java

Fahrrad.java Java

```
```

___/$3\frac{1}{2}$ (d) Fügen Sie der Klasse `Routenplanung` eine **öffentliche Klassenmethode** `zeit` hinzu, die ein beliebiges `Fahrzeug` und eine Streckenlänge (in km) übergeben bekommt, die benötigte Zeit (in Stunden) berechnet und zurückgibt. Die Zeit ist der Quotient $\frac{s}{v}$ aus Streckenlänge s und Durchschnittsgeschwindigkeit v. Auch Rückgabewerte zwischen 0 und 1 sollen möglich sein. **(Rückseite beachten)**

Routenplanung.java Java

```
public class Routenplanung {









}
```

___/$7\frac{1}{2}$ (e) Ergänzen Sie die Klasse `Demo`, sodass in der `main`-Methode folgendes passiert:

1. Eine Person, die irgendwann im Jahr 2009 geboren ist und einen Führerschein hat, wird erstellt.
2. Eine `Auto`-Instanz wird erstellt.
3. Es wird mithilfe dieser Instanz auf der Standardausgabe ausgegeben, ob die Person das Auto mieten kann.
4. Mithilfe der Klasse `Routenplanung` wird die benötigte Zeit für eine 10 km lange Strecke berechnet und das Ergebnis auf der Standardausgabe ausgegeben.

Hinweis: Bedenken Sie, dass `Demo` **nicht** im Package `java.time` liegt.

Demo.java Java

```
public class Demo {

    public static void main(String[] args) {









    }

}
```

___/2 (f) Angenommen, wir als Dienstleister packen alle Klassen aus dieser Aufgabe in einer jar-Datei zusammen und geben der Vermietungs-Firma nur die jar-Datei; in der jar-Datei befinden sich dann nur die class-Dateien der Klassen. Sowohl Dienstleister als auch Firma verwenden das in der Vorlesung vorgestellte JDK 21.

Die Vermietungs-Firma möchte die Software nun in einem Projekt selbständig erweitern, hat aber keinen Zugriff auf die Quellcode-Dateien. Die jar-Datei ist im Classpath zum Projekt.

Kreuzen Sie die korrekten Aussagen an:

- ☐ Es ist für die Vermietungs-Firma möglich, eine weitere Implementierung `EBike` des Interfaces `Fahrzeug` zu erstellen.
- ☐ Es ist möglich, der Methode `Routenplanung.zeit` ein Objekt der Klasse `EBike` zu übergeben, obwohl zum Zeitpunk, als `Routenplanung` compiliert wurde, die Klasse `EBike` noch nicht existiert hat.
- ☐ Es ist **nicht** möglich, der Methode `Routenplanung.zeit` ein `EBike`-Objekt zu übergeben, weil die Klasse `EBike` nicht in der jar-Datei enthalten ist.

Für Aufgabe 5

Matrix.java Java

```java
public class Matrix {
    private final int[][] elements;

    public Matrix(int[][] elements) {
        if(elements == null || !isMatrix(elements)) {
            throw new IllegalArgumentException("no matrix given");
        }
        this.elements = elements;
    }

    private static boolean isMatrix(int[][] array) {
        for (int[] row: array) {
            if (row == null || array[0].length != row.length) {
                return false;
            }
        }
        return true;
    }

    public Matrix multiply(Matrix that) {
        if (that == null || this.elements[0].length != that.elements.length) {
            throw new IllegalArgumentException();
        }
        int[][] product = new int[this.elements.length][that.elements[0].length];
        for (int i = 0; i < product.length; i++) {
            for (int j = 0; j < product[0].length; j++) {
                for (int k = 0; k < this.elements[0].length; k++) {
                    product[i][j] += this.elements[i][k] * that.elements[k][j];
                }
            }
        }
        return new Matrix(product);
    }
```

Für Aufgabe 6

Im Package `java.time` sind die Klassen `LocalDate` und `Period` enthalten.

Eine Instanz der Klasse `LocalDate` repräsentiert ein Datum. Die für diese Aufgabe relevanten Methoden in der Klasse sind:

- `static LocalDate of(int year, int month, int dayOfMonth)`: Erzeugt eine neue Instanz mit dem angegebenen Jahr, Monat (1–12) und Tag (1–31).
- `static LocalDate now()`: Gibt das aktuelle Datum (heute) zurück.

Eine Instanz der Klasse `Period` beschreibt den Abstand zweier Datums-Angaben:

- `static Period between(LocalDate startDateInclusive, LocalDate endDateExclusive)`: Berechnet den Abstand zwischen den beiden Datumsangaben.
- `int getYears()`: Gibt die Anzahl (vollständiger) Jahre des `Period`-Objekts zurück.

Sie dürfen dieses Blatt abtrennen und als Schmierzettel benutzen. Dieses Blatt muss nicht, darf aber abgegeben werden. Wenn das Blatt abgetrennt ist und bewertet werden soll, muss es als loses Blatt auf dem Deckblatt angegeben werden.