Name: **Vorname:** **Matr.-Nr.:**

Hochschule Kaiserslautern
FB Informatik und Mikrosystemtechnik
Prof. Dr. M. **Duque-Antón**

Bachelor-Klausur im WiSe 2022 / 2023

im Fach

Grundlagen der Informatik

Angewandte Informatik / Medieninformatik / Medizininformatik

Datum: 1**. Februar 2023**

Es sind **keinerlei Hilfsmittel** zur Klausur zugelassen.

Achtung: Bitte schreiben Sie die **Lösung in / unter die Aufgabe in das Aufgabenblatt**. Wenn Sie mehr Platz benötigen, legen Sie **von der Aufsicht** zu erhaltende **zusätzliche Blätter** dazwischen.

Die schriftliche Prüfung besteht aus 5 Aufgaben. Schreiben Sie bitte auf **jedes Blatt Ihren Namen und Matrikelnummer**. Es werden nur Blätter mit Namen und Matrikelnummer korrigiert oder bewertet. Unleserliche Lösungen, Lösungen mit Bleistift oder Rotstift werden nicht korrigiert oder bewertet. **Java-Lösungen**, die höhere Programmierkonstrukte **wie Collections oder Generics verwenden**, werden mit **0 Punkten** bewertet.

1	2	3	4	5	Summe	Note[a]
/23	/19	/21	/34	/23	/120	

a.Eine 1.0 gibt es ab 100 Punkte, eine 5.0 unter 50 Punkte.

1. Aufgabe: Allgemeine Grundlagen

Gegenstand dieser Aufgabe sind allgemeine Themen aus den Bereichen Rechnerarchitektur, Betriebssysteme, Programmierung und Design. Konkret sollen die folgenden Fragen kurz beantwortet werden:

(a) Erläutern Sie die Verwendung und Bedeutung der Modifier **static** und **final**!

(b) Welchen Wert liefert der arithmetische Ausdruck (**n**++ + ++**n**) für eine beliebige Integer Variable n, welcher Wert wird von dem Ausdruck (**--n - n--**) zurückgeliefert?

(c) Methoden können **überschrieben** werden, was ist damit gemeint? Was wird mit **Überladen** von Methoden ausgedrückt?

(d) Erläutern Sie das **Interface-Segregation-Prinzip** und wie es in Java umgesetzt werden kann!

(e) In Java wird der Basis-Datentyp **byte** angeboten. Wie sieht der entsprechende **Zahlenbereich** aus?

(f) Seien die beiden Klassen B (Basis) und A (abgeleitet von B) gegeben. Wie sehen die **entsprechenden Standard-Konstruktoren** der Klassen A und B aus?

(g) In einem Ordner befindet sich aktuell **nur** die Datei MyClass.java. Welche Befehle sind notwendig, um das entsprechende **Java-Programm** zur **Ausführung** zu bringen? In welcher Reihenfolge müssen diese erfolgen, wie sieht der Inhalt des entsprechenden Ordners anschließend aus?

1.)	a	b	c	d	e	f	g	Summe
Punkte	/4	/2	/4	/3	/3	/4	/3	/23

2. Aufgabe: Zahlendarstellungen und Sprachen

Gegenstand dieser Aufgabe ist die **Darstellung** von **Zahlen** innerhalb eines Rechners und der Umgang mit den **Operatoren** auf der **Bit-Ebene**.

Hinweis: In **Java** können **Literale** auf der Basis verschiedener **Basissysteme** bearbeitet werden. Mit den Präfixen **0**, **0b** und **0x** sind jeweils die Zahlensysteme **Oktal, Binär** bzw. **Hexadezimal** gemeint. Bitte beachten Sie auch, dass in Java **Ganzzahlen Literale** implizit vom Typ int sind.

(a) Gegeben seien die folgenden Java-Anweisungen. Wie sieht die entsprechende Ausgabe auf dem Bildschirm aus?

```
System.out.println("a = " + 0b00000111);

System.out.println("b = " + 0110);

System.out.println("c = " + 0x00000111);

System.out.println("d = " + 0b10000110);

System.out.println("e = " + (byte) 0b10000110);
```

(b) In dieser Teilaufgabe wird die Anwendung der Operatoren auf der Bit-Ebene überprüft. Auch hier wieder die Frage, wie sieht die entsprechende Ausgabe auf dem Bildschirm aus?

```
System.out.println("f = " +  (byte) ~63);

System.out.println("g = " +  (9 & 13));

System.out.println("h = " +  (13 | 25));
```

2.)	a	b	Summe
Punkte	/10	/9	/19

3. Aufgabe: Java-Programmierung

Gegenstand dieser Aufgabe ist die Programmierung mit Hilfe der **Programmiersprache Java**, insbesondere die Verwendung von **Rekursion** und **Iteration.**

(a) Gegeben sei die folgende **Klasse Aufgabe3a**, welche bereits die Methode iterativ enthält. Fügen Sie bitte der Klasse eine **weitere Methode** mit dem Namen **rekursiv hinzu**, welche **dieselbe Funktionalität** liefert und das Prinzip der **Rekursion** verwendet.

```
class Aufgabe3a {

    public static void main (String [] args) {
            int eingabe = Integer.parseInt (args[0]);
            System.out.println ("Das iterative Ergebnis lautet: " + iterativ (eingabe));
            System.out.println ("Das rekursive Ergebnis lautet: " + rekursiv (eingabe));
    } // main

    static int iterativ (int n) {
            int result = 0;
            for (int i = 1; i <= n; i++)      result += i;
            return result;
    } // iterativ
```

```
} // Aufgabe3a
```

(b) Ein funktionierendes Java-Programm ist durcheinander geraten. Es soll die folgende Ausgabe erzeugen:

java Aufgabe3b
0 4
0 3
1 4
1 3
3 4
3 3

Die einzelnen Codeteile des Programms sind in folgender Liste ungeordnet angegeben.

x++
if (x == 1)
System.out.println (x + " " + y)
class Aufgabe3b
for (int y=4; y>2; y--)
for (int x=0; x<4; x++)
public static void main (String[] args)

In der Unordnung sind leider auch alle geschweiften Klammern und Semikolons verloren gegangen. Bauen Sie aus den Bausteinen, Semikolons und geschweiften Klammern wieder ein korrektes Programm, das die obige Ausgabe erzeugt! Achten Sie dabei auf korrekte Einrückungen.

(c) Aus der Vorlesung kennen Sie die **Fibonacci-Zahlen** bzw. die entsprechende Folge F(n):

$$F(n) = \begin{cases} n & \text{für } n <= 1 \\ F(n-1) + F(n-2) & \text{für } n > 1 \end{cases}$$

Die **schnellste** iterative Implementierung der **Fibonacci-Folge** ist von der **Ordnung O (n)**. Bitte geben Sie die entsprechende **Java-Implementierung** an!

3.)	a	b	c	Summe
Punkte	/7	/7	/7	/21

4. Aufgabe: Java-Programmierung

Gegenstand dieser Aufgabe ist die Programmierung mit Hilfe der Programmiersprache Java, insbesondere die Verwendung interner Datenstrukturen und der Umgang mit Hilfs-Methoden (Problem-Reduktion) in Java.

(a) Ein **magisches Quadrat** ist eine quadratische Matrix, für die die Summe jeder Zeile und jeder Spalte denselben Wert ergibt. **Bitte entscheiden Sie ob eine vorgegebene quadratische Matrix ein magisches Quadrat darstellt**!

Hinweis: Zur Lösung implementieren Sie eine **Klasse MagischesQuadrat**, welches keine Attribute enthält, genau eine von **Außen benutzbare Klassenmethode istMagisch** und zwei **private Hilfs-Klassenmethoden summeZeile und summeSpalte** enthält.
Die Methode **istMagisch** erwartet als **Eingabe ein quadratisches zwei-dimensionales Array von int-Werten** und **liefert** ein **boolean-Wert** zurück.
Die **beiden Hilfsmethoden** sollen jeweils **einen int-Wert zurückgeben** und genau zwei **Eingabe-Werte erwarten**.

(b) Implementieren Sie eine statische Methode mit dem Namen **gegenDiagonale**, welche als **Eingabe** ein **2-dimensionales quadratisches Array von int-Werten** erwartet und als Ausgabe die **Gegen-Diagonale** in Form eines eindimensionalen Arrays zurück liefert.
Hinweis: Implementieren Sie nur die Methode gegenDiagonale. Die Gegen-Diagonale liefert, salopp gesprochen, im Gegensatz zur Diagonalen die Werte von „oben rechts abwärts bis unten links."

(c) Im Default Directory sind aktuell die beiden Dateien **Aufgabe4c.java**, **Aufgabe4c.class** und **eingabe.txt** enthalten. Wie sieht die **Ausgabe in der Konsole** aus, wenn das entsprechende Programm aufgerufen wird, also nach dem Befehl **java Aufgab4c**?

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Aufgabe4c {
    public static int [][] methode1 (String datei)  throws FileNotFoundException {
        Scanner eingabe = new Scanner (new File  (datei));
        int [][] matrix;
        matrix = new int [eingabe.nextInt()][];
        for (int i=0; i<matrix.length; i++) {
            matrix[i] = new int [eingabe.nextInt()];
            for (int j=0; j < matrix[i].length; j++)
                matrix[i][j] = eingabe.nextInt();
        }
        eingabe.close ();
        return matrix;
    } // methode1

    public static int [] methode2 (int [][] m)   {
        int [] v = new int [m.length];
        for (int i=0; i < v.length; i++)
            for (int j=0; j < m[i].length; j++)
                v[i] += m[i][j];
        return v;
    } // methode2

    public static void methode3 (int [] v)   {
        for (int i=0; i < v.length; i++)
            System.out.println (v[i]);
    } // methode3

    public static void main (String [] args)  throws FileNotFoundException {
        methode3 (methode2 (methode1 ("eingabe.txt")));
    } // main

} // Aufgabe 4c
```

Inhalt der Datei **eingabe.txt**: **4 2 1 2 3 4 2 5 1 3 2 8 9**

Wie sieht die Ausgabe auf dem Bildschirm aus?

4.)	a	b	c	Summe
Punkte	/15	/7	/12	/34

5. Aufgabe: Objektorientierung

(a) Implementieren Sie eine **Klasse Time** zur Speicherung einer **Zeit (-Dauer)**. Ein Time-Objekt enthält zwei von Außen nicht einsehbare **Integer - Attribute Stunden** und **Minuten**. Die **Minuten** liegen wie üblich im **Bereich von 0 bis 59**. Der Wertebereich für **Stunden** ist nach oben nicht begrenzt, kann also **beliebige Werte beginnend mit dem Wert 0** annehmen. Der Zugriff von Außen auf das Objekt soll maximal eingeschränkt sein. Es sollen folgende Methoden implementiert werden:

Der **Konstruktor** soll bei der Generierung des Objekts die entsprechende Zeit aufnehmen.

Mit Hilfe der **Instanz-Methode add** sollen die Zeiten zweier Time-Objekte addiert werden. Die Methode add erwartet genau ein Time-Objekt als Eingabe und addiert das eigene Time-Objekt dazu. Als Ergebnis wird ein korrektes neues Time-Objekt zurückgeliefert.

Mit Hilfe der **Methode sub** soll der Betrag der Differenz zwischen zwei Time-Objekten in Minuten berechnet werden. Die Methode sub erwartet genau ein Time-Objekt als Eingabe und subtrahiert das eigene Time-Objekt davon. Als Ergebnis wird der Betrag der entsprechenden Differenz in Minuten zurückgegeben.

Des Weiteren soll eine sinnvolle **Methode print** implementiert werden, welche für ein Time-Objekt die Zeit in Stunden und Minuten auf dem Bildschirm ausgibt.

Hinweis: **public static int abs (int a)** returns the absolute value of an int value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Method abs is part of the package **java.lang.Math**.

(b) Betrachten Sie die folgenden Klassen **Beleg** und **TestBeleg**. Was wird auf dem Bildschirm ausgegeben, wenn Sie **java TestBeleg** in die Kommandozeile eingeben?

```
class Beleg {

    private static int anzahlBelege = 100;
    private int belegNummer = 0;

    Beleg ( ) {
        belegNummer  = ++anzahlBelege;
    } // Beleg

    int nummer ( ) {
        return belegNummer;
    } // nummer

    static int anzahl ( ) {
        return anzahlBelege;
    } // anzahl

} // Beleg

public class TestBeleg {
    public static void main (String [] args) {
        System.out.println ("Belege fangen mit Nummer " + Beleg.anzahl ( ) + "an");
        Beleg [] beleg = new Beleg [2];
        for (int i = 0; i < beleg.length; i++) {
            beleg [i] = new Beleg ( );
        } // for
        for (int i = 0; i < beleg.length; i++) {
            System.out.println (" Beleg mit Nummer " + beleg[i].nummer ( ) );
        } // for

    } // main

} // TestBeleg
```

5.)	a	b	Summe
Punkte	/16	/7	/23