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

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

Bachelor-Klausur im WiSe 2023 / 2024

im Fach

Grundlagen der Informatik

Angewandte Informatik / Angewandte Informatik-D / Medizininformatik

Datum: **14. Februar 2024**

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

Achtung: Bitte schreiben Sie die **Lösung 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.

1	2	3	4	5	Summe	Note[a]
/28	/25	/23	/20	/24	/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) Beschreiben Sie den Unterschied zwischen einer **Klassenvariablen** und einer **Instanzvariablen**!

(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) Die deutsche Sprache besitzt (von Umlauten einmal abgesehen) **26 Buchstaben**. **Wieviele Bits** sind notwendig, um diese als **Binär-Muster** darstellen zu können?

(e) Erläutern Sie die **Cast-Operation** (**implizit** und **explizit**) an Hand von jeweils einem kleinen Beispiel!

(f) Im Zusammenhang mit Objekten wird nun auch zwischen Up- und Down-Cast unterschieden. Erläutern Sie kurz beide Begriffe insbesondere in Bezug auf Polymorphie.

(g) Mit Hilfe der beiden folgenden Java-Befehle werden zwei String-Objekte erzeugt. Wie kann überprüft werden, ob der Inhalt beider **Objekte s1 und s2 identisch ist?**

```
String s1 = new String ("Klausur");
String s2 = new String ("Klausur");
```

1.)	a	b	c	d	e	f	g	Summe
Punkte	/4	/4	/4	/3	/4	/6	/3	/28

2. Aufgabe: Zahlendarstellungen und Sprachen

Gegenstand dieser Aufgabe ist die **Darstellung** von **Zahlen** innerhalb eines Rechners und die **Beschreibung von Sprachen** mit Hilfe von Grammatiken und der **EBNFs (**Extended Backus Naur Form**)**.

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 = " + 0b00001001);

System.out.println("b = " + 0x00000104);

System.out.println("c = " + (short) 0b10100011);

System.out.println("d = " + (byte) 0b10100011);
```

(b) Gegeben sei die folgende Grammatik G = (N, T, P, S) mit N = {S, A}, T = {0, 1} und P = {(S,0), (S,1A0), (A,ε), (A,0A), (A,1A)}. Wie sieht die durch L(G) erzeugte Sprache aus, welche Elemente sind darin enthalten? **Beschreiben** Sie bitte diese **Sprache anschaulich mit eigenen Worten in einem einzigen Satz**!

(c) Überprüfen Sie, ob die folgenden **Worte** mit den vorgegebenen **EBNFs erzeugt** werden können und markieren die entsprechenden Stellen mit OK!.

	{a} b [a]	{a} | {b}	{a | b}	[a] {(ab)} [b] {a}
ε				
aaab				
aaba				
aabba				

2.)	a	b	c	Summe
Punkte	/8	/9	/8	/25

3. Aufgabe: Java-Programmierung

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

(a) Wenn man von Umlauten einmal absieht, besteht das **deutsche Alphabet** aus **26 Buchstaben**. In der ASCII-Tabelle sind diese **26 Buchstaben konsekutiv** abgelegt und zwar jeweils in verschiedenen Bereichen für die kleinen bzw. großen Buchstaben.

Implementieren Sie die **Java-Methode alphabet**, welche **alle großen Buchstaben** in einer Zeile in der korrekten Reihenfolge hintereinander ausdruckt, also:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Hinweis: Innerhalb der Methode alphabet darf **explizit nur ein Buchstabe als Literal** verwendet werden. Die Methode alphabet erwartet **keine Eingabe** und liefert auch **keinen Wert zurück**.

```
static void alphabet ( ) {



}// alphabet
```

(b) Gegeben sei die folgende Klasse WasWohl. Was wird beim Aufruf **java WasWohl** auf dem Bildschirm ausgegeben?

```
public class WasWohl {

    public static void main (String [] args) {
        for (int i=0; i<=8; i++)
            System.out.println ("value = " + methode (i));
    } // main

    static int methode (int value) {
        if (value <= 2 ) return value;
        return methode (value -1) + methode (value - 2) + methode (value - 3);
    } // methode

}// WasWohl
```

(c) Die **Fibonacci-Zahlen** bzw. die entsprechende Folge F(n) ist folgt formuliert:

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

Als naheliegende Lösung bietet sich sofort die **rekursive Variante** an. Wie sieht in diesem Fall die **Rechen-Komplexität** aus? (Kein Java Source Code gefragt!)

(d) 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	d	Summe
Punkte	/6	/8	/3	/6	/23

4. Aufgabe: Java-Programmierung

Gegenstand dieser Aufgabe ist die Programmierung mit Hilfe der Programmiersprache Java, insbesondere die Verwendung interner Datenstrukturen.

(a) **Welche Ausgabe** wird durch das **folgende Java-Programm Aufgabe4a erzeugt**?
Hinweis: Aus der online Java Dokumentation kann entnommen werden, dass die Methode **static void fill (int[] a, int val)** aus der **Klasse Arrays** die folgende Funktionalität besitzt: **Assigns the specified int value to each element of the specified array of ints**.

```
import java.util.Arrays;

class Aufgabe4a {

        static void print (int [] vektor) {
                for (int i = 0; i < vektor.length; i++)
                        System.out.print ( vektor [i] + " ");
                System.out.println ();
        } // print

        public static void main (String [] args) {
                int [ ] v = {2, 4, 3, 3};
                int [ ] [ ] m = new int [v.length] [ ];
                for (int i = 0; i < m.length; i++)
                        m [i] = new int [ v [i] ] ;

                m[1] = v;
                Arrays.fill (m[0], 7);
                m[2][0] = m[0][0];
                m[3][1] = m[1][1];
                m[3][2] = m[2][0];

                System.out.print ("a: ");
                print (m [0]);

                System.out.print ("b: ");
                print (m [1]);

                System.out.print ("c: ");
                print (m [2]);

                System.out.print ("d: ");
                print (m [3]);
        } // main

} // Aufgabe4a
```

(b) Eine natürliche **Zahl n** heißt **perfekt**, wenn sie gleich der Summe aller ihrer echten Teiler ist.
Als Beispiel betrachten Sie die Zahl **n = 6**. Ihre echten Teiler sind 1, 2 und 3. Da die Summe **1 + 2 + 3 = 6** gilt, ist die Zahl **6 eine perfekte Zahl**.
Die Zahl **n = 8** ist nicht perfekt, da die Summe ihrer Teiler: $1 + 2 + 4 \neq 8$.

Implementieren Sie eine **statische Methode** mit dem **Namen perfekteZahl**, welche als **Eingabe** eine **int-Zahl** erwartet und als **Ausgabe** einen **boolean-Wert** zurück liefert mit der Bedeutung true (perfekte Zahl) und sonst false.
Hinweis: Implementieren Sie nur die statische Methode **perfekteZahl**. Benutzen Sie dabei den **Modulo-Operator n % t**.

4.)	a	b	Summe
Punkte	/10	/10	/20

5. Aufgabe: Objektorientierung

(a) Ein Vektor im dreidimensionalen Raum kann durch drei reelle Zahlen dargestellt werden. Implementieren Sie eine **Klasse Vektor**, welche **Instanz-Methoden** (**und nur diese**) für die **Addition** von zwei Vektoren, die **Multiplikation** eines Vektors mit einem Skalar und die **Ausgabe** eines Vektors anbietet. Stellen Sie einen geeigneten **Konstruktor** bereit und vergeben die **Zugriffsrechte sparsam** (Information Hiding Prinzip).

(b) Implementieren Sie eine weitere Klasse Test zum interaktiven Testen der Operationen der Klasse Vektor. Führen Sie dazu die (**und genau die**) folgenden **Schritte** aus:

Generieren Sie zunächst die beiden **Vektoren v = (3, 4, 6)** und **w = (2, 5, 7).**

Anschließend **addieren Sie beide Vektoren**, das Ergebnis soll über den **Vektor u erreichbar** sein. Verwenden Sie dann Ihre **Ausgabe**-Methode, um das **Ergebnis der Addition** (Vektor u) auf der Standardausgabe sichtbar zu machen.

Danach **multiplizieren Sie den Vektor v mit dem Skalar 5**, das Ergebnis soll über den **Vektor z erreichbar** sein. Verwenden Sie Ihre **Ausgabe**-Methode, um das **Ergebnis der skalaren Multiplikation** (Vektor z) auf der Standardausgabe sichtbar zu machen!

(c) Im folgenden seien die Java-Klassen **Konto**, **Girokonto** und verschiedene **Test-Klassen** vorgegeben. Beantworten Sie hierzu die Fragen auf den nächsten Seiten:

```
package Bank;
public class Konto {


    public int k;
    public double s;
    static public int n;

    public Konto ( ){
        k = ++n;
    } // Konto

    public Konto (double value){
        this ();
        s += value;
        System.out.println ("Konto " + k + " generiert mit Kontostand: " + s);
    } // Konto

    public void auszahlen (double value){
        if (value <= s)  s -= value;
    } //auszahlen

    public void saldo ( ){
        System.out.println ("Kontostand Konto " + k + " beträgt: " + s);
    } //auszahlen

} // Konto


package Bank;
public class Girokonto extends Konto {

    public double  d = 1000;

    public Girokonto () {
        super ();
        System.out.println ("Girokonto " + k + " generiert mit Kontostand: " + s);
    } // Girokonto

    public void auszahlen (double value){
    if (value <= s + d)  s -= value;
    } //auszahlen

    public void set (double value){
    d += value;
    } //set

} // Girokonto
```

Welche **Ausgabe** liefert das folgende **Test-Programm** auf dem **Bildschirm**?

```
import Bank.Konto;
import Bank.Girokonto;
class Test1 {
    public static void main (String [] args) {

            Konto k1 = new Konto (0);
            k1.auszahlen (500);
            k1.saldo ();

            Konto k2 = new Girokonto ();
            k2.auszahlen (500);
            k2.saldo ();

    } // main
} // Test1
```

Kompiliert das folgende **Test-Programm**? **Erläutern** Sie bitte Ihre Antwort!

```
import Bank.Konto;
import Bank.Girokonto;
class Test2 {
    public static void main (String [] args) {
            Girokonto k = new Konto (0);
            k.auszahlen (500);
            k.saldo ();
    } // main
} // Test2
```

Kompiliert das folgende **Test-Programm**? **Erläutern** Sie bitte Ihre Antwort und **beschreiben** Sie eine mögliche Lösung des Problems!

```
import Bank.Konto;
import Bank.Girokonto;
class Test3 {
    public static void main (String [] args) {
            Konto k = new Girokonto ( );
            k.auszahlen (500);
            k.set (300);
    } // main
} // Test3
```

5.)	a	b	c	Summe
Punkte	/10	/5	/9	/24