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

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

Bachelor-Klausur im WiSe 2021 / 2022

im Fach

Grundlagen der Informatik

Angewandte Informatik / Medieninformatik / Medizininformatik

Datum: 9**. Februar 2022**

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]
/30	/25	/21	/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) Bitte stellen Sie eine beliebige for-Schleife mit Hilfe einer while-Schleife dar.Verwenden Sie dazu folgende allgemeine Darstellung:

```
for (A; B; C) {
   D;
}
```

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

(d) Erläutern Sie das **Single Responsibility Prinzip**! Erläutern Sie auch den entsprechenden Vorteil!

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

(f) In Java werden **8 primitive Daten-Typen** bereit gestellt. Bitte zählen diese auf!

(g) Erläutern Sie die beiden **Cast-Varianten, implizit** und **explizit,** an Hand von jeweils einem kleinen Beispiel (primitive Datentypen)!

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

1.)	a	b	c	d	e	f	g	h	Summe
Punkte	/4	/4	/4	/4	/3	/4	/4	/3	/30

2. Aufgabe: Zahlendarstellungen und Sprachen

Gegenstand dieser Aufgabe ist die **Darstellung** von **Zahlen** innerhalb eines Rechners, der Umgang mit den **Operatoren** auf der **Bit-Ebene** und die **Beschreibung von Sprachen** auf der Basis von **BNF** (Backus Naur Form) und **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 = " + 00000000101);

System.out.println("c = " + 0x00000101);

System.out.println("d = " + (byte) 0b11100011);
```

(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("e = " +  (byte) ~25);

System.out.println("f = " +  (14 & 13));

System.out.println("g = " +  (21| 17));
```

(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: Sie **müssen** und **dürfen** dazu **nur einen Buchstaben explizit** in der Methode verwenden.

```
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) Gegeben sei die folgende Methode **method**. Geben Sie bitte eine Java-**Variante an, die ohne Rekursion** auskommt:

```
public static int  method (int n) {
        if (n < 0) return - method (-n);
        if (n == 0) return n;
        return method (n-1) + 1;
} // methode
```

3.)	a	b	c	Summe
Punkte	/8	/8	/5	/21

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 = {3, 4, 5, 6};
                int [ ] [ ] m = new int [v.length] [ ];
                for (int i = 0; i < m.length; i++)
                        m [i] = new int [ v [i] ] ;

                m[0] = v;
                Arrays.fill (m[1], 10);
                Arrays.fill (m[2], 7);
                m[3][1] = m[1][1];
                m[3][2] = m[2][1];

                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) Entwerfen Sie eine Klasse **Vector**, welche zur Darstellung und Bearbeitung von **Vektoren der Form (v_1, v_2, ... ,v_n)** verwendet werden kann, entsprechend den folgenden Vorgaben an die Attribute, Konstruktoren und Methoden. Der Vektor **kann nur Werte vom Typ int** verwalten.

Als **Attribut** soll ein ein-dimensionales Array verwendet werden, welches den Vektor der Größe n darstellt und von Außen nicht sichtbar ist.

Es soll genau ein **Konstruktor** mit **einem Parameter** vom Typ int bereitgestellt werden, welcher die Größe des Vektors erwartet und entsprechend Speicherplatz für das interne Array allokiert.

Insgesamt sollen **4 Instanzmethoden** von der Klasse Vector (**öffentlich**) angeboten werden: Konkret sollen zwei **fill** Varianten, **print** und **add**, die im folgenden näher beschrieben werden, implementiert werden:

Die Methode **fill** erwartet einen Parameter vom Typ int und gibt keinen Wert zurück. Als Resultat sollen alle Werte des Vektors mit dem Wert des Parameters aufgefüllt werden.

Eine weitere Variante der Methode **fill** erwartet zwei Parameter vom Typ int und gibt keinen Wert zurück. Der erste Parameter gibt die Position an, die mit einem int-Wert (zweiter Parameter) gefüllt werden soll.

Die Methode **print** erwartet keinen Parameter und liefert auch keinen Wert zurück. Als Resultat soll der aktuelle Inhalt des Vektors auf der Standardausgabe ausgegeben werden, und zwar **genau in der Form (v_1, v_2, ... ,v_n)**.

Die Methode **add** erwartet einen Parameter v vom Typ Vector und liefert einen Rückgabewert vom Typ Vector. Als Resultat soll also ein neues Objekt vom Typ Vector zurückgeliefert werden. Der Inhalt des neuen Vektors ergibt sich aus der Vektor-Addition des (die Methode) aufrufenden Objekts und des Parameter-Objekts v. Für den Fall, dass beide Vektoren nicht kompatibel sind, soll der leere Zeiger zurückgegeben werden.

(b) 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);
    } //saldo

} // 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 **Test1-Programm,** falls in der Klasse **Konto** die Deklaration **static public int n** durch **static private int n** ersetzt wird?
Erläutern Sie Ihre Antwort!

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	Summe
Punkte	/12	/12	/24