Klausur über den Stoff der Vorlesung
Programmieren I

Name: _____ Vorname: _____

Matnr.: _____ Studiengang: _____

Semester: WiSe 24/25

Ich bestätige, dass ich die folgenden Anmerkungen gelesen habe, prüfungsfähig bin und mich von der Vollständigkeit dieses Klausurexemplares (Seiten 1 – 26) überzeugt habe. Ich bestätige, dass ich mich nicht im letzten Wiederholungsversuch der Prüfung befinde oder bin mit der Durchführung des letzten Wiederholungsversuchs als schriftliche Prüfung einverstanden.

_____
Unterschrift der o. g. Klausurteilnehmer:in

Anmerkungen:

1. Legen Sie bitte Ihren Studierendenausweis bereit.
2. Bitte tragen Sie *Name*, *Vorname, Studiengang* und *Matrikelnummer* deutlich lesbar in Druckbuchstaben in den Feldern oben ein.
3. Dieser Test beinhaltet **10 Aufgaben** mit insgesamt **90 Punkten**.
4. Folgende Hilfsmittel sind zugelassen: **keine.**
5. Täuschungsversuche führen zum Nichtbestehen der Modulprüfung.
6. Unleserliche oder mit Bleistift geschriebene Lösungen werden nicht gewertet.
7. Die Bearbeitungszeit beträgt 90 Minuten. Sie erhalten 10 Minuten Einlesezeit.
8. Die Aufgaben sind entweder auf Englisch oder auf Deutsch, aber pro Aufgabe durchgehend in genau einer Sprache, zu beantworten.

Nur für Prüfende:

1:13	**2**:13	**3**:10	**4**:6	**5**:8	**6**:12	**7**:4	**8**:11	**9**:7	**10**:6	**Σ90**

Aufgabe 1 – Grundlagen **(5 + 4 + 4 = 13 Punkte)**

a) Das folgende Programm beinhaltet 5 (Laufzeit-)Fehler. Geben Sie in der Ergebnistabelle jeweils die Zeilennummer des Fehlers sowie eine (sehr kurze) Begründung an, bspw. den Namen einer Exception oder eine ähnlich kurze (aber eindeutige) Information. Pro richtig identifiziertem Fehler mit Begründung erhalten Sie einen Punkt.

```
// Importanweisungen ausgelassen
1  public class Fehlersuche {

2     private Map<Integer, String> myMap = new Map<>();

3     public Fehlersuche(int start, String... values) {
4        int i = 0;
5        do {
6           myMap.put(start++, values[i++]);
7        } while (i < values.length);
8     }


9     public void printMap() {
10       for (Object o : myMap.keySet())
11          System.out.println(o.toString() + " -> " + myMap.get(o));
12    }

13    public static String getValue(Map map, int key) {
14       if (map.containsKey(key))
15          return map.get(key);
16       return "";
17    }

18    public static void main(String[] args) {
19       Fehlersuche f = new Fehlersuche(5, "AB", "CD", "EFG");
20       Fehlersuche f2 = new Fehlersuche(1);
21       Fehlersuche f3 = new Fehlersuche(1, 2, 3);
22       Fehlersuche.printMap();
23       Fehlersuche.getValue(f.myMap,5);

24    }
   }
```

Zeilennummer	Fehlerursache

b) Implementieren Sie eine Methode `highestValue`, die ein eindimensionales Number-Array übergeben bekommt und das größte Element in diesem Array zurückgibt. Im Fehlerfall, z. B. bei ungültigen Parametern, soll `null` zurückgegeben werden. *Hinweis: Mit der Instanzmethode `doubleValue()` erhalten Sie den Wert eines `Number`-Objektes als `double`.*

c) Implementieren Sie eine Methode `create`, die ein quadratisches `boolean`-Array (2D) zurückgibt. Die Seitenlänge des Quadrats wird als ganzzahliger Parameter übergeben. Eine Zelle soll mit dem Wert `true` befüllt werden, wenn die Position im inneren Array ohne Rest durch drei teilbar ist. Sonst soll die Zelle mit `false` gefüllt werden. Im Fehlerfall, z. B. bei ungültigem Parameter, soll `null` zurückgegeben werden.

Aufgabe 2 – Datenstrukturen **(7 + 6 = 13 Punkte)**

a) Gegeben sind die untenstehenden Binärbaumknotenklassen für Integer- bzw. String-Werte. Ihre Aufgabe ist es, eine generische Klasse `Node` zu schreiben, die die beiden gegebenen Klassen ersetzt. Die neue Klasse `Node` wird beispielhaft in der `main`-Methode weiter unten verwendet. Implementieren Sie die `Node` Klasse so, dass Sie die gleiche Anzahl von Attributen und Methoden (und die gleiche Funktionalität) hat wie die Klassen StringNode und `IntNode`.

```java
public class StringNode {
    final String value;
    StringNode left;
    StringNode right;

    public StringNode(String value) {
        this.value = value;
    }

    public void add(String value) {
        if(this.value.compareTo(value) < 0) {
            left = set(left,value);
        } else {
            right = set(right,value);
        }
    }

    private StringNode set(
            StringNode node,
            String value
        ) {
        if(node == null) {
            return new StringNode(value);
        } else {
            node.add(value);
            return node;
        }
    }
}
```

```java
public class IntNode {
    final Integer value;
    IntNode left;
    IntNode right;

    public IntNode(Integer value) {
        this.value = value;
    }

    public void add(Integer value) {
        if(this.value.compareTo(value) < 0) {
            left = set(left,value);
        } else {
            right = set(right,value);
        }
    }

    private IntNode set(
            IntNode node,
            Integer value
        ) {
        if(node == null) {
            return new IntNode(value);
        } else {
            node.add(value);
            return node;
        }
    }
}
```

```java
...
public static void main(String[] args) {
    Node<String> a = new Node<>("Tatü");
    a.add("Tata");
    Node<Integer> b = new Node<>(7);
    b.add(9);
}
...
```

b) Gegeben ist die Klasse `MyStackElement`. Vervollständigen Sie die Methoden `push()` zum Hinzufügen und `pop()` zum Entfernen von Elementen in der Klasse `MyStack`.

```
public class MyStackElement {

    public Integer value;
    public MyStackElement next;

    MyStackElement(Integer value, MyStackElement next) {
        this.value = value;
        this.next = next;
    }
}
```

```
public class MyStack {

    private MyStackElement top = null;

    public void push(Integer value) {









    }
```

```
    public Integer pop() {

    }
}
```

Aufgabe 3 – Objekte **(10 Punkte)**

Gegeben ist das untenstehende Programm. Geben Sie in der Tabelle unten für die jeweilig genannte Variable den Wert an, den diese an der im Code mit `//hier` markierten Stelle hat. Für jede richtige Zeile erhalten Sie einen Punkt. *Tipp: Skizzieren Sie, welche Objekte wie erstellt und verändert werden.*

```
public class Foo {

   protected int a, b;
   private static int c = 6;

   static Foo buildFoo(int a, int b) {
      Foo f = new Foo();
      f.a = a;
      f.b = b;
      return f;
   }

   void invert() {
      int temp = this.a;
      this.a = this.b;
      this.b = temp;
   }

   Foo redo(Foo other) {
      Foo f = new Foo();
      f.a = other.a - this.a;
      f.b = other.b + this.b;
      return f;
   }

   void replace(int a, int b) {
      a = b;
   }

   void overwrite() {
      a = 42;
      b = ++c;
   }

   public static void main(String[] args) {
      Foo f = buildFoo(3, 2);
      Foo f2 = buildFoo(5, 1);
      Foo f3 = Bar.cloneFoo(f);
      f.invert();
      Foo f4 = f.redo(f2);
      f.overwrite();
      f2.overwrite();
      f4.replace(9, 3);
      f3.overwrite();
      f3.invert();
      //hier
   }

}
```

```
class Bar extends Foo {
   void invert() {
      this.a = -a;
      this.b = -b;
   }

   static Bar cloneFoo(Foo foo) {
      Bar b = new Bar();
      b.a = foo.a;
      b.b = foo.b;
      return b;
   }
}
```

Variable	Wert
f.a	
f.b	
f.c	
f2.a	
f2.b	
f2.c	
f3.a	
f3.b	
f4.a	
f4.b	

Aufgabe 4 – Vererbung **(6 Punkte)**

Gegeben ist untenstehendes Programm. Kreuzen Sie darunter an, was das Programm beim Ausführen der Klasse `Caesar` ausgibt.

```
public class Caesar {
   public static void main(String[] args) {
      Veni a = new Veni();
   }
}

class Vici {
   static Printme mp = new Printme("Tres!");
}

class Vidi extends Vici {
   static Printme mp1 = new Printme("In!");
   Printme mp2 = new Printme("Partes!");

   public Vidi() {
      System.out.print("Divisa!");
   }
}

class Veni extends Vidi {
   static Printme mp = new Printme("Est!");
   Printme mp2 = new Printme("Omnis!");

   public Veni() {
      System.out.print("Gallia!");
   }
}

class Printme {
   public Printme(String msg) {
      System.out.print(msg);
   }
}
```

O Gallia!Est!Omnis!Divisa!In!Partes!Tres!

O Tres!In!Partes!Divisa!Est!Omnis!Gallia!

O Divisa!Gallia!Tres!Partes!In!Omnis!Est!

O Tres!In!Est!Partes!Divisa!Omnis!Gallia!

O Tres!In!Est!Partes!Omnis!Divisa!Gallia!

O Est!In!Tres!Omnis!Partes!Gallia!Divisa!

Aufgabe 5 – Algorithmen und Datenstrukturen **(1 + 4 + 1 + 2 = 8 Punkte)**

a) Die Effizienz von Algorithmen wird durch die Zuordnung zu Komplexitätsklasssen beschrieben. Nennen Sie die Komplexitätsklasse, in der die folgende Methode liegt.

```
public int method(int x, int ... ints){
    int sum = 0;
    for (int i = 0; i < ints.length; i++) {
        int a = x;
        int res = 0;
        if (ints[i] > 0){
            if (ints[i] % 2 != 0){
                res += a;
            }
            a = a*2;
            ints[i] = ints[i]/2;
        }
        sum += res;
    }
    return sum;
}
```

c) Gegeben sind die untenstehenden Use-Cases, die Datenhaltung mithilfe der Collections realisieren. Nennen Sie jeweils die für den beschriebenen Anwendungsfall am besten geeignete Datenstruktur (konkrete Klasse, keine abstrakten Klassen oder Interfaces). Nennen Sie genau eine Datenstruktur!

Eine Anwendung speichert für eine fixe Menge an Feuerwehrautos jeweils den aktuellen Standort. Es soll schnell anhand des Kennzeichens auf den Standort eines bestimmten Autos zugegriffen werden können.

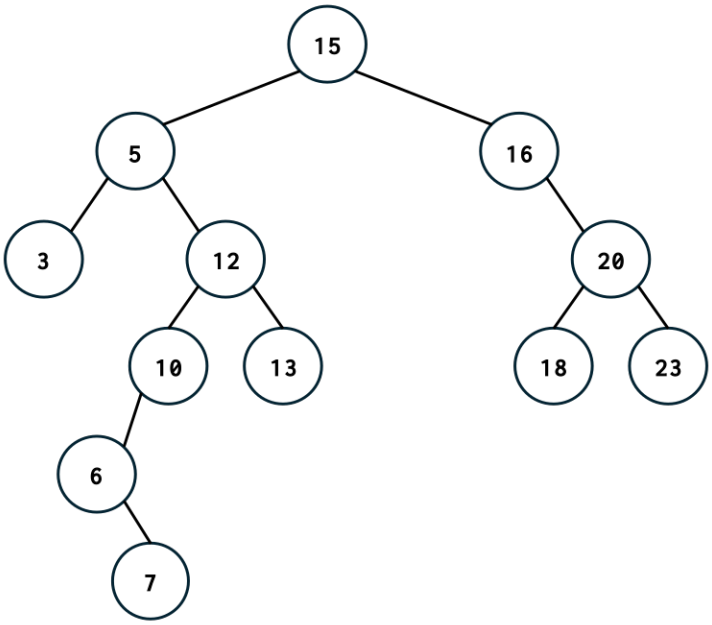
Eine Anwendung speichert Informationen zu allen Produkten des Ausstattungskatalogs. Der Zugriff darauf erfolgt stets über deren ganzzahlige, fortlaufende Produkt-ID. Diese IDs beginnen immer bei 0. Das Auslesen soll möglichst schnell sein und die Reihenfolge der Produkte ist wichtig.

In einer Stadt brennen aktuell sehr viele Feuer. Die Feuerwehr muss entscheiden, welches Feuer zuerst gelöscht werden soll. Dabei sollen die Intensität des Feuers und die Höhe der „Spende“ an die Feuerwehrkaffeekasse eine Rolle spielen.

Es soll der Speicherverbrauch und die Anzahl der geöffneten Dateien von jedem auf einem Computer laufenden Prozesse gespeichert werden. Jeder Prozess wird über einen zufälligen String identifiziert.

d) Erläutern oder skizzieren Sie kurz, warum das Einfügen einer aufsteigend oder absteigend sortierten Wertemenge in den Ihnen aus der Vorlesung bekannten sortierten Binärbaum problematisch ist.

e) Gegeben ist ein sortierter Binärbaum. Geben Sie diesen in Pre-Order aus, trennen Sie die Werte dabei durch Bindestriche ab. Zeichnen Sie *danach* ein, wie der Baum aussieht, wenn man den Wert 14 hinzufügt und der Baum danach noch sortiert ist.



Aufgabe 6 – Streams & Sockets **(12 Punkte)**

Implementieren Sie einen einfachen Server. Stellen Sie sicher, dass Ihr Server über eine `main`-Methode gestartet wird und auf Port 9090 auf Anfragen wartet. Nutzen Sie eine Instanz der Klasse `BufferedReader`, um sämtliche vom Client gesendeten Informationen zu lesen und anschließend auf der Konsole auszugeben. Die Verbindung zum Client soll vom Server geschlossen werden, sobald der Client den String „end“ übersendet oder die Verbindung durch den Client geschlossen wird. Achten Sie darauf, dass Ihre Serverimplementierung (theoretisch) unbegrenzt viele parallele Anfragen bedienen kann und dass Sie beim Lesen entstehende Exceptions abfangen. Tipp: Nutzen Sie die Klassen `ServerSocket`, `Socket`, `Thread` und `InputStreamReader`. Importanweisungen können Sie auslassen.

Aufgabe 7 – Concurrency **(3 + 1 = 4 Punkte)**

a) Sie sollen im Auftrag der Verwaltung des kinderreichen Bezirks Prenzlzwerg einen konfliktfreien Kinderspielplatz simulieren. Untenstehender Code wurde bereits von Ihrem Kollegen vorbereitet, der allerdings im Gegensatz zu Ihnen nicht verstanden hat, wie man in Java korrekt mit Threads programmiert. Ergänzen Sie in der Methode `backeKuchen()` die notwendigen Codezeilen, die sicherstellen, dass immer nur ein Kind gleichzeitig mithilfe der Förmchen- und Schaufelobjekte einen Sandkuchen backen kann. D. h., für den Vorgang des Sandkuchenbackens benötigt ein Kind alleinigen Zugriff auf sowohl die Schaufel als auch das Förmchen. Achten Sie darauf, möglichst effizienten Code zu schreiben.

```
public class Sandkasten {

    private final Foermchen foermchen = new Foermchen();
    private final Schaufel schaufel = new Schaufel();
    private final static Sandkasten sandkasten = new Sandkasten();



    private Sandkasten() {


    }



    public static Sandkasten getSandkasten() {
        return sandkasten;
    }

    public void backeKuchen() {



        foermchen.backeKuchen();



        schaufel.backeKuchen();



    }


    // weitere Methoden, die exklusiven Zugriff auf foermchen oder schaufel benötigen
    // ...

    public static void main(String[] args) {
        for (int i = 0; i < 50; i++)
            new Kind("Kind-"+i).start();
    }
}
```

```
class Foermchen {
    public void backeKuchen() {
        System.out.println(Thread.currentThread().getName()
                + " bäckt Kuchen mit Förmchen!");
    }
}

class Schaufel {
    public void backeKuchen() {
        System.out.println(Thread.currentThread().getName()
                + " bäckt Kuchen mit Schaufel!");
    }
}

class Kind extends Thread {

    public Kind(String name){
        super.setName(name);
    }

    public void run() {
        for (int i = 0; i < 100; i++) {
            Sandkasten.getSandkasten().backeKuchen();
            try {
                Thread.sleep((long) (Math.random() * 100));
            } catch (InterruptedException e) {
            }
        }
    }
}
```

b) Stellen Sie sich vor, dass in einem Programm mehrere Instanzen von Sandkasten parallel benutzt werden. Welche Auswirkung hätte hier ein `synchronized`-Block, der eine Klassenvariable als Lock benutzt? Antworten Sie in einem Satz.

Aufgabe 8 – Programmentwurf **(6 + 5 = 11 Punkte)**

a) Für ein Programmierprojekt sollen Sie die Klassen/Interfaces/Enums `Fahrzeug`, `Bus`, `Fahrrad`, `Haltestelle`, `Personenbefördernd` und `FahrzeugFarbe` (mögliche Varianten: `Braun`, `Blau`, `Schwarz`, `Silber`) implementieren. Ergänzen Sie die untenstehende Tabelle, so dass diese auf sinnvolle Art und Weise eine Vererbungshierarchie beschreibt. Geben Sie jeweils an, ob Sie den Typ über eine Klasse (abstrakt oder konkret), ein Interface oder ein Enum realisieren würden und zu welchen anderen Typen diese über extends oder implements in einer Vererbungshierachie stehen. Möchten Sie angeben, dass kein extends oder implements notwendig ist, wählen Sie dort bitte „keine“. Für jede richtige Zeile erhalten Sie einen Punkt.

Name	Art	extends	implements
Fahrzeug	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine
Bus	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine
Fahrrad	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine
FahrzeugFarbe	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine
Haltestelle	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine
Personenbefördernd	○ abstract class ○ class ○ interface ○ enum	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine	○ Fahrzeug ○ Bus ○ Fahrrad ○ FahrzeugFarbe ○ Haltestelle ○ Personenbefördernd ○ keine

b) Sie sollen im Auftrag eines Unternehmens, das Busse und Fahrräder vermietet, das Fahrzeugbestandssystem überarbeiten. Ergänzen Sie hierfür die untenstehende Klasse `BestandVerwaltung`, sodass folgende Aspekte erfüllt sind. *Hinweis: Imports können ignoriert werden.*

- Die Variable `bestand` soll für jede Fahrzeugnummer vom Typ `int` den Verweis auf den entsprechenden Bus oder das entsprechende Fahrrad beinhalten. Wählen Sie einen geeigneten Datentyp. Fügen Sie keine weiteren Instanz-/Klassenvariablen ein.
- Die Methode `neueLieferung` soll allen Fahrzeugen, die neu in den Bestand aufgenommen werden sollen, mit Hilfe der Variable `nextId` eine eindeutige ID zuweisen und beide in der Variable `bestand` hinterlegen. Da Busse und Fahrräder von unterschiedlichen Herstellern produziert werden, muss die Methode sowohl Buslisten als auch Fahrradlisten verarbeiten können.

```
public class BestandVerwaltung {

                    bestand = new

    static int nextId = 0;

    public void neueLieferung(List<?                          > neu) {

    }
}
```

Aufgabe 9 – Ausblick **(4 + 3 = 7 Punkte)**

a) Welche Ausgabe liefert folgendes Programm?

```
interface Print{
    void print(String text);
}

public class CustomPrinter implements Print {
    String lastPrint = "";

    @Override
    public void print(String text) {
        System.out.println(text);
        lastPrint = text;
    }

    public class CustomInnerPrinter implements Print {
        @Override
        public void print(String text) {
            System.out.println(text + lastPrint);
            lastPrint = text;
        }
    }
}

class Write {
    public static void main(String[] args) {
        new CustomPrinter().new CustomInnerPrinter().print("ABC");
        Print p = new CustomPrinter();
        p.print("DEF");
        Print p2 = ((CustomPrinter) p).new CustomInnerPrinter();
        p2.print("GHI");
    }
}
```

b) Gegeben ist folgendes Programm in der Programmiersprache Kotlin. Welche Ausgabe liefert es?

```
fun main() {
    println(listOf(1,2,3,4,5).mystic())
    println(listOf(5,4,3,2,1).mystic())
}

fun List<Int>.mystic(): Int {
    val v = this.first() ?: return -1
    this.forEach {
        if (it < v) return it
    }
    return -1
}
```

Aufgabe 10 – Rekursion **(6 Punkte)**

Gegeben ist folgender Code, der iterativ funktioniert. Schreiben Sie die noch fehlende Überladung der Methode `mysteryRecursive` so, dass die gleiche Funktionalität durch Rekursion erfüllt wird.

```
public static void main(String[] args) {
    System.out.println(mysteryIterative(4, 5));
    System.out.println(mysteryRecursive(4, 5));
}
```

```
public static double mysteryIterative(double base, int exponent) {
    double result = 1;
    for (int i = 0; i < exponent; i++) {
        result *= base;
    }
    return result;
}

public static double mysteryRecursive(double base, int exponent) {
    double currentResult = 1;
    return mysteryRecursive(base, exponent, currentResult);
}
```

```
// Your implementation here:
```