

Bevor Sie mit der Bearbeitung der Klausur beginnen, lesen Sie bitte folgende Hinweise:

- Prüfen Sie die **Vollständigkeit** Ihres Exemplars. Jede Klausur umfasst
 - das Deckblatt
 - diese Hinweisseite
 - 7 Aufgaben mit Lösungsblättern
 - einen Anhang mit Quellcode, den Sie als gegeben annehmen und verwenden dürfen

Bei Unvollständigkeit Ihres Exemplars wenden Sie sich bitte an die Aufsicht.

- Tragen Sie auf **jedem Lösungsblatt** oben Ihre **Matrikelnummer** ein und **unterschreiben** Sie auf dem Deckblatt! Blätter ohne diese Angabe werden nicht bewertet.
- Hinter den Aufgaben ist jeweils hinreichend Platz für Ihre Lösungen. Bei Bedarf nutzen Sie auch die **Rückseiten** der Aufgabenblätter und der Deck- oder Hinweisblätter. Sollten Sie weiteres Papier benötigen, so wenden Sie sich bitte an die Aufsicht.
Verwenden Sie kein eigenes Papier!
- Stellen Sie sicher, dass die Zuordnung von Lösung zu Aufgabe auf jeden Fall deutlich erkennbar ist.

Es werden maximal 100 Punkte vergeben.

Sie haben bestanden, wenn Sie mindestens 40 Punkte erreicht haben.

Ergebnis (bitte nichts eintragen):

Aufgabe	1	2	3	4	5	6	7	Σ
Punkte	10	10	10	15	30	10	15	100
erreicht								

Aufgabe 1 (10 P)

Im Anhang finden Sie den Code einer Klasse `Funktion`.

Erstellen Sie eine Tracetabelle mit einer Spalte für die Code-Zeilenummer und je einer Spalte für jede vorkommende Variable. Tragen Sie die Werte der Variablen bei Aufruf der Methode `berechne(3, 5)` ein.

Geben Sie außerdem an, für welche Parameter die Funktionen f bzw g jeweils aufgerufen werden.

Aufgabe 2 (10 P)

Im Anhang finden Sie zwei Klassendefinitionen der Klassen A und B und eine main-Methode. Erläutern Sie durch ein Speicherbild die Situation bei Ausführung der main-Methode. Was ist die Ausgabe des Programms?

Aufgabe 3 (10 P)

Gegeben sei ein Array von int-Werten. Der *Rang* (engl. rank) eines Elementes e ist die Anzahl der Einträge, die höchstens gleich e sind.

Schreiben Sie eine (statische) Methode `rank()`, die ein int-Array und eine Positionsnummer (dh einen Index) übergeben bekommt.

Die Methode soll den Rang des Elementes **an der angegebenen Position** zurückliefern.

Falls der übergebene Index eine ungültige Position darstellt, soll die Methode den Wert -1 liefern.

Beispiel: Das Array habe die Einträge $\{6, -2, 4, 0, 7, 6, 1, -3\}$.

An Position 2 steht das Element 4, dieses hat den Rang 5, denn es gibt 5 Einträge, die ≤ 4 sind.

Nutzen Sie **keine** der API-Klassen `Arrays`, `ArrayList`, ... und auch nicht die erweiterte for-Schleife („for-each“)!

Aufgabe 4 (15 P)

Bei einem Weitsprung-Wettbewerb hat jeder teilnehmende Sportler 6 Versuche.

Wir nehmen an, es gebe eine Klasse `Sportler` mit folgenden Eigenschaften:

Jeder Sportler verfügt als Instanzattribut über ein `double`-Array der Länge 6, in das die Weiten seiner sechs Sprünge eingetragen werden. Auf dieses Array kann durch eine `get`-Methode `double[] weite()` zugegriffen werden.

Ein Array `tnFeld` („Teilnehmer-Feld“) von Sportlern sei mit Sportler-Objekten belegt, jeder Sportler habe bereits seine 6 Sprünge absolviert und die Weiten seien bereits in das Weiten-Array des betreffenden Sportlers eingetragen.

Implementieren Sie eine (statische) Methode `sieger`, die ein Array von Sportlern als Eingabeparameter erhält. Die Methode soll denjenigen Sportler ermitteln, der die größte Weite erzielt hat.

(Sie dürfen davon ausgehen, dass mindestens einer der Sportler eine positive Weite erreicht hat und dass es keine zwei Sprünge mit exakt der gleichen Weite gibt.)

Nutzen Sie **keine** der API-Klassen `Arrays`, `ArrayList`, ... und auch nicht die erweiterte `for`-Schleife („`for-each`“)!

Aufgabe 5 (30 P)

Sie sollen eine Software für die Mitgliederverwaltung eines Vereins entwickeln.

- Mitglieder sind spezielle Personen. (Eine Implementierung der Klasse `Person` finden Sie im Anhang.)
Jedes Mitglied ist entweder aktiv oder inaktiv. Nur solche Objekte sollen erzeugt werden können.
 - Jedes Mitglied erhält eine Mitgliedsnummer, die fortlaufend (beginnend bei 1) bei Erzeugung eines Objektes automatisch generiert wird.
Bei der Erzeugung werden außerdem der Name und das Geburtsjahr erfasst.
Die (Gesamt-)Anzahl aller Mitglieder soll abgefragt werden können.
 - Für jedes Mitglied soll der (individuell) zu zahlende Jahresbeitrag ermittelt werden können.
Wie der Beitrag berechnet wird, hängt allerdings von der Art der Mitgliedschaft (aktiv/inaktiv) und ggf weiteren Faktoren ab:
 - Aktive Mitglieder zahlen einen pauschalen Jahresbeitrag von 75,00 Euro, inaktive Mitglieder zahlen nur 20,00 Euro.
Diese pauschalen Beitragssätze für die aktiven und inaktiven Mitglieder sollen durch je eine `set`-Methode geändert werden können.
 - Aktive Mitglieder können zusätzlich „Sozialstunden“ leisten.
Durch eine Methode `arbeiten(int std)` wird die Anzahl der geleisteten Sozialstunden eines Aktiven gezählt. Für jede geleistete Sozialstunde sinkt der Jahresbeitrag eines Aktiven um 5,00 Euro (aber nicht auf weniger als 0 Euro). Der Betrag, mit dem eine Sozialstunde vergütet wird, kann (für alle aktiven Mitglieder gleichermassen) geändert werden.
 - Inaktive Mitglieder können ihren individuellen Jahresbeitrag durch freiwillige Spenden erhöhen. Eine Methode `spenden(double spende)` erhöht den Jahresbeitrag eines inaktiven Mitglieds entsprechend.
Durch die beiden oben genannten Möglichkeiten (`arbeiten()` bzw. `spenden()`) können die tatsächlich zu zahlenden Beiträge einzelner Mitglieder von den pauschalen Beitragssätzen abweichen.
- a) Erstellen Sie ein Diagramm in UML-ähnlicher Notation, um die Abhängigkeiten der vier Klassen `Person`, `Mitglied`, `Aktiv`, `Inaktiv` darzustellen.
Nutzen Sie die Konzepte von **Vererbung** und **Abstraktion**, wo immer das möglich ist.
(Eine Beschreibung der Attribute und Methoden ist nicht nötig - Klassennamen und -Modifizierer genügen.)
- b) Implementieren Sie die Klassen `Mitglied`, `Aktiv` und `Inaktiv`. Weitere Methoden (`get-`, `set-`, `toString`) als die oben geforderten sind nicht nötig.
- c) Schreiben Sie eine (statische) Test-Methode, in der zwei aktive und zwei inaktive Vereinsmitglieder erzeugt werden.
Eins der aktiven Mitglieder soll einmal 5 und ein weiteres mal 3 Sozialstunden ableisten. Eines der inaktiven Mitglieder soll 100,00 Euro spenden. Lassen Sie für alle 4 Mitglieder den zu zahlenden Jahresbeitrag ausgeben.

(Platz für Ihre Lösung)

Aufgabe 6 (10 P)

Von der Klasse `Exception` sei folgende Unterklasse abgeleitet:

```
public class PersonException extends Exception {  
    public PersonException(String txt) {  
        super(txt);  
    }  
}
```

- a) Schreiben Sie eine (statische) Methode `void erfasseDaten()`, die einen `String` und einen `int`-Wert als Parameter erhält.
Falls der übergebene `String` leer ist oder der übergebene `int`-Wert > 2023 ist, soll eine `PersonException` mit einem entsprechenden Fehlertext ausgelöst werden.
Ansonsten macht die Methode nichts.
- b) Schreiben Sie eine (statische) Methode `Person createPerson()`, die ebenfalls einen `String` und einen `int`-Wert als Parameter erhält.
Die Methode ruft die Methode `erfasseDaten()` mit diesen beiden Parametern auf.
Die Methode erzeugt ein Objekt der Klasse `Person` mit diesen beiden Parametern und gibt dieses `Person`-Objekt als `return`-Wert an die aufrufende Methode zurück.
Eine ggf. aufgetretene `PersonException` wird an die aufrufende Methode weitergeleitet.
Der Konstruktor der Klasse `Person` hat die Signatur (vgl. Anhang)
`public Person(String name, int gebJahr)`
- c) Schreiben Sie eine `main`-Methode, in der die Methode `createPerson()` aufgerufen wird.
Eventuell auftretende `Exceptions` sollen hier abgefangen und behandelt werden.

(Platz für Ihre Lösung)

Aufgabe 7 (15 P)

Das Interface `Menge` zur Darstellung von Mengen ganzer Zahlen sei wie folgt definiert:

```
public interface Menge {  
  
    int size();  
    // liefert die Anzahl der Elemente der Menge M  
    boolean contains(int n);  
    // gibt an, ob n in M enthalten ist  
    void insert(int n);  
    // fuegt n zu M hinzu (aendert M nicht, falls n bereits enthalten ist)  
    void remove(int n);  
    // entfernt n aus M (aendert M nicht, falls n nicht in M enthalten ist)  
    int get() throws NoSuchElementException;  
    // liefert ein (nicht naeher bestimmtes) Element der Menge,  
}  

```

a) Definieren Sie ein Unter-Interface `MengeMitOps` von `Menge`, das zusätzlich folgende Operationen bereitstellt:

- `merge(Menge m)`: fuegt der instanziierten Menge alle Element aus `m` hinzu
- `intersect(Menge m)`: erzeugt eine neue Menge und liefert diese zurück, die die Schnittmenge der instanziierten Menge mit `m` bildet.
- Sowohl die instanziierte Menge als auch die übergebene Menge dürfen dabei verändert werden.

b) Nehmen Sie an, das Interface `Menge` sei durch eine Klasse `MyMenge` implementiert.
(Die Implementierung dieser Klasse brauchen Sie nicht zu kennen.)

Implementieren Sie eine Klasse `MyMengeMitOps` als Unterklasse von `MyMenge`, die das Interface `MengeMitOps` implementiert.

Bemerkung:

`NoSuchElementException` ist eine ungeprüfte Exception; Sie brauchen also in Ihrem Code keine Fehlerbehandlung vorzunehmen.

(Platz für Ihre Lösung)

Anhang

Code zu Aufgabe 1:

```
1  public class Funktion {
2
3      public static int f(int x) {
4          return x + 2;
5      }
6
7      public static int g(int x, int y) {
8          return 4*x-3*y;
9      }
10
11     public static int berechne(int a, int b) {
12         int erg = 0;
13         int zw;
14         boolean b1 = true;
15         boolean b2;
16         while(b1) {
17             b2 = b < 6;
18             if (b2) {
19                 zw = g(a, b-1);
20             }
21             else {
22                 zw = f(a+b);
23                 b = b-1;
24             }
25             erg = erg + zw;
26             a = 2*a;
27             b = b+2;
28             b1 = a < 10;
29         }
30         return erg;
31     }
32 }
```

Code zu Aufgabe 2:

```
public class A {

    public static int s = 0;
    public int a;
    public int as;

    public A(int n) {
        a = n;
        s = s+n;
        as = s;
    }

    public void incr() {
        s++;
        a++;
    }

    public String toString() {
        return "s: " + s + " a: " + a + " as: " + as;
    }
}

////////////////////////////////////

public class B {

    public int b;
    public A aa;

    public B(int n) {
        b = n;
        aa = new A(b);
    }

    public B(int n, A x) {
        b = n;
        aa = x;
    }

    public String toString() {
        return "b: " + b + " aa: " + aa;
    }
}

////////////////////////////////////

public static void main(String[] args) {
    A p = new A(2);
    System.out.println("p: " + p);
    B q = new B(4);
    System.out.println("q: " + q);
    B r = new B(6, p);
    System.out.println("r: " + r);
    p.incr();
    System.out.println("p: " + p);
    System.out.println("q: " + q);
    System.out.println("r: " + r);
}
```

Code zu Aufgabe 5 und 6:

```
public class Person {  
    private String name;  
    private int gebJahr;  
  
    public Person(String name, int gebJahr) {  
        this.name = name;  
        this.gebJahr = gebJahr;  
    }  
  
    public String name() {  
        return name;  
    }  
  
    public int gebJahr() {  
        return gebJahr;  
    }  
}
```