Sigrid Weil

Klausur SS 2023
- Musterlösung -

Einführung in die Programmierung

Aufgabe 1 (10 P)
Im Anhang finden Sie den Code einer Klasse `Funktion`.

Erstellen Sie eine Tracetabelle mit einer Spalte für die Code-Zeilennummer 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.

Lösung 1
Tracetabelle:

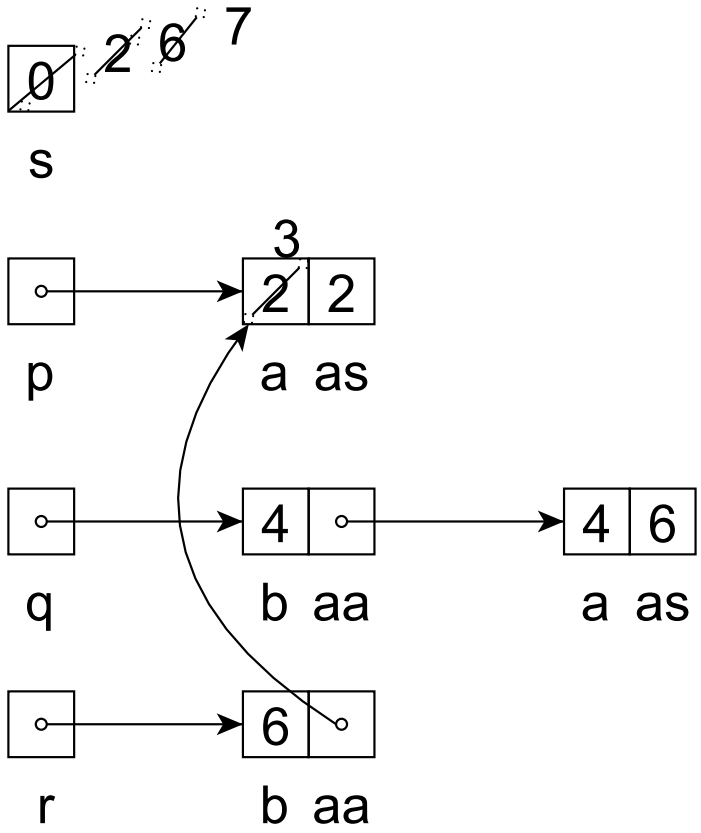
Z-Nr	a	b	erg	zw	b1	b2	Aufruf
11	3	5					
12			0				
14					true		
17						true	
19				0			g(3, 4)
25			0				
26	6						
27		7					
28					true		
17						false	
22				15			f(13)
23		6					
25			15				
26	12						
27		8					
28					false		
30							return 15

Bewertung:

nur „init“: 2 P
fehlende Aufrufparams von f, g: -2P, fehlende Aufrufstelle (ZNr) -2P
Schleife nicht erkannt: -4P
Ende-Bedg (ZNr 30 erreicht oder return-Wert angegben) nicht formuliert: - 1P
Rechenfehler: - 1P
Folgefehler (FF) keine zusätzlichen Abzüge, es sei denn, dadurch wird die Struktur deutlich einfacher

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?

Lösung 2
Speicherbild:



Ausgabe des Programms:

```
p: s: 2 a: 2 as: 2
q: b: 4 aa: s: 6 a: 4 as: 6
r: b: 6 aa: s: 6 a: 2 as: 2
p: s: 7 a: 3 as: 2
q: b: 4 aa: s: 7 a: 4 as: 6
r: b: 6 aa: s: 7 a: 3 as: 2
```

Bewertung:

s statisch: 1P
Referenz r.aa korrekt: 2P
Referenzvariablen enthalten Pfeile, simpleTypes enthalten Werte !
falsche Werte: -1

Ausgabe korrekt: 3P

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“)!

Lösung 3

```
public static int rank(int[] a, int pos) {
                if (pos < 0 || pos >= a.length)
                        return -1;
                int x = a[pos];
                int r = 0;
                for (int i = 0; i < a.length; i++) {
                        if(a[i] <= x)
                                r++;
                }
                return r;
        }
```

Bewertung:

Methodenkopf: 1 P
pos prüfen: 2 P
Wert a[pos] bestimmen: 1 P
Schleifenkopf: 2 P
Schleifenrumpf: 4 P

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“)!

Lösung 4

```
public static Sportler sieger(Sportler[] tnFeld) {
        Sportler s = null;
        double maxWeite = 0.0;
        double[] w;
        for (int i = 0; i < tnFeld.length; i++) {
                w = tnFeld[i].weite();
                for (int j = 0; j< w.length; j++) {
                        if (w[j] > maxWeite) {
                                s = tnFeld[i];
                                maxWeite = w[j];
                        }
                }
        }
        return s;
}
```

Bewertung:

Methodenkopf: 1 P
inits: 3 P
Schleifenkopf äußere Schleife: 2 P
weite() richtig aufgerufen: 1 P
Schleifenkopf innere Schleife: 2P
Schleifenrumpf innere: 5 P
return: 1 P

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.

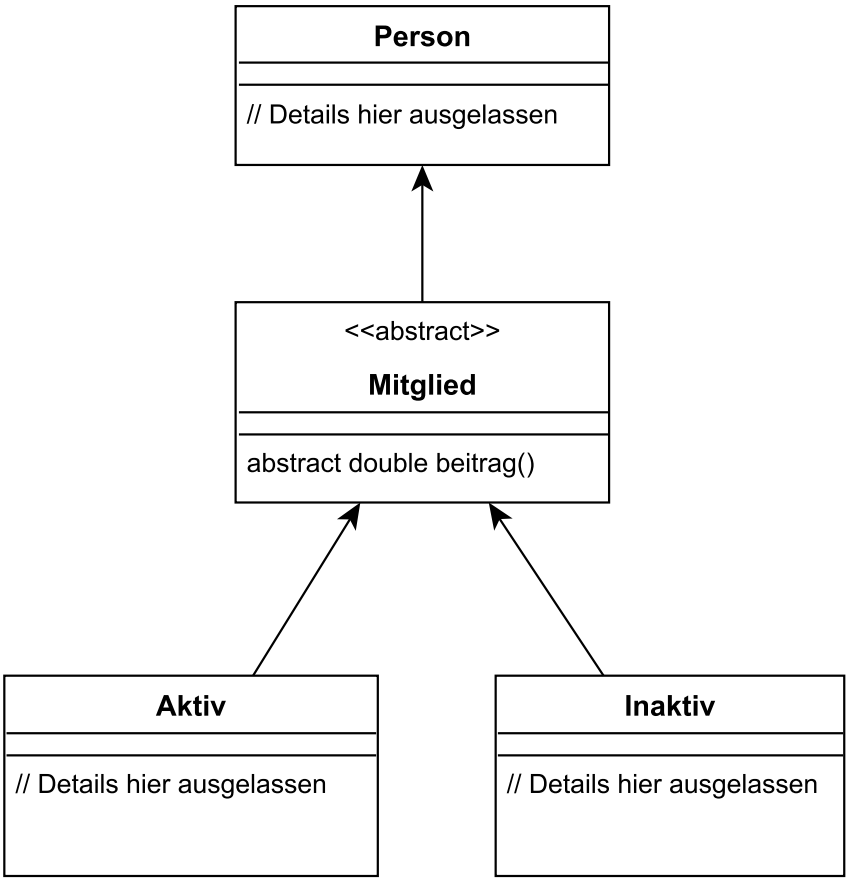
a) Erstellen Sie ein Diagramm in UML-ähnlicher Notation, um die Abhängigkeiten der vier Klassen `Person`, `Mitglied`, `Aktiv`, `Inaktiv` darzustellen.
   Nutzen Sie die Konzepte von **Vererbung** und **Abstraktion**, wo immer das möglich ist.
   (Eine Beschreibung der Attribute und Methoden ist nicht nötig - Klassennamen und -Modifier 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 weiters mal 3 Sozialstunden ableisten. Eines der inaktiven Mitglieder soll 100,00 Euro spenden. Lassen Sie für alle 4 Mitglieder den zu zahlenden Jahresbeitrag ausgeben.

Lösung 5

a) UML-Daigramm:



b) Code:

```
public abstract class Mitglied extends Person {

        private static int anzahl = 0;

        private int nr;

        public Mitglied(String n, int gj) {
                super(n, gj);
                nr = ++anzahl;
        }

        public static int anzahl() {
                return anzahl;
        }

        public abstract double beitrag();
}

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

public class Aktiv extends Mitglied {


        private static double beitrag = 75.0;
        private static double verguetung = 5.0;

        private int sozialstunden = 0;

        public Aktiv(String n, int gj) {
                super(n, gj);
        }

        public static void setBeitrag(double beitragNeu) {
                beitrag = beitragNeu;
        }

        public static void setVerguetung(double v) {
                verguetung = v;
        }

        public double beitrag() {
                double b = beitrag - this.sozialstunden*verguetung;
                if (b >= 0)
                        return b;
                else
                        return 0;
        }

        public void arbeiten(int std) {
                sozialstunden += std;
        }
}


//////////////////////////////////////////////////////////////////////
```

```
public class Inaktiv extends Mitglied {

        private static double beitrag = 20.0;

        private double spende = 0.0;

        public Inaktiv(String n, int gj) {
                super(n, gj);
        }

        public static void setBeitrag(double beitragNeu) {
                beitrag = beitragNeu;
        }

        public double beitrag() {
                return beitrag + this.spende;
        }

        public void spenden(double spende) {
                this.spende += spende;
        }
}
```

c) Testmethode:

```
public static void test() {
        Aktiv a = new Aktiv("Anna", 2000);
        Aktiv b = new Aktiv("Ben", 1999);
        Inaktiv c = new Inaktiv("Cem", 2002);
        Inaktiv d = new Inaktiv("Didi", 1990);

        a.arbeiten(5);
        c.spenden(100.0);
        a.arbeiten(4);
        c.spenden(80.0);

        System.out.println(a + " zahlt " + a.beitrag());
        System.out.println(b + " zahlt " + b.beitrag());
        System.out.println(c + " zahlt " + c.beitrag());
        System.out.println(d + " zahlt " + d.beitrag());
}
```

Bewertung:

a) 4 P Summe, dabei 2P Abzug für fehlendes „abstract“, kein Abzug für Pfeile in falsche Richtung ...

b) class Mitglied: 6 P, davon 2 für Mitgliedsnummer richtig erzeugt, 1P für abstrakte Methode deklariert
class Aktiv: 9 P
class Inaktiv: 7 P
jeweils 1 P für korrekte extends-Angaben
jeweils 1 für Konstruktoren mit super-Aufruf

c) 4 P

kein Abzug für „kleine“ Fehler (Syntax etc)

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 eine int-Wert als Parameter erhält.
...

c) Schreiben Sie eine main-Methode, in der die Methode `createPerson()` aufgerufen wird. Eventuell auftretende Exceptions sollen hier abgefangen und behandelt werden.

Lösung 6

```
public static void main(String[] args) {
        try {
                Person a = createPerson("Anna", 2000);
                System.out.println("Objekt angelegt");
        }
        catch (PersonException e) {
                System.out.println("kein Objekt angelegt");
                System.out.println(e.getMessage());
        }

}

public static void erfasseDaten(String n, int j) throws PersonException {
        if (n == null || n == "" )
                throw new PersonException("Name darf nicht leer sein");
        if (j > 2023)
                throw new PersonException("Ungueltiges Geburtsjahr");
}

public static Person createPerson(String n, int j) throws PersonException {
        erfasseDaten(n, j);
        return new Person(n, j);
}
```

Bewertung:

a) 4 P

b) 3 P

c) 3 P

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.

Lösung 7

a) Interface:

```
public interface MengeMitOps extends Menge {

        void merge(Menge m);

        MengeMitOps intersect(Menge m);

}
```

b) Implementierende Klasse:

```
public class MyMengeMitOps extends MyMenge implements MengeMitOps {

        public MyMengeMitOps() {
                super();
        }

        public void merge(Menge m) {
                int n;
                while (m.size() > 0) {
                        n = m.get();
                        m.remove(n);
                        this.insert(n);
                }
        }

        public MengeMitOps intersect(Menge m) {
                MengeMitOps cut = new MyMengeMitOps();
                int n;
                while(m.size() > 0) {
                        n = m.get();
                        m. remove(n);
                        if (this.contains(n))
                                cut.insert(n);
                }
                return cut;
        }

}
```

Bewertung:

a) 3 P

b) 12 P , davon:
2 P für korrekten Klassenkopf. je 4 bzw 6 P für Implementierung der beiden Methoden