Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 2 SoSe 2024
Fakultät für Angewandte Informatik
Institut für Informatik
Prof R. Lorenz, L. Petrak, J. Linne, V. Le Claire, P. Schalk 03.05.2024

Lösungsvorschlag zu Übungsblatt 3

Abgabe: 20.05.2024, 10:00 Uhr (im Digicampus via VIPS: .java-Dateien für Code, .uxf für UML, .pdf für alles andere)

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe

Aufgabe 9 *(Wissensfragen (API, Klassen, Schnittstellen), 20 Minuten)*

Beantworten Sie möglichst knapp und genau in einem Satz die folgenden Fragen:

a) *(*, Die Klasse `Currency`, 8 Minuten)*

1. Wie lautet der vollständige Prototyp der Klasse?
   ```
   public final class Currency extends Object
   implements Serializable
   ```
2. Beschreiben Sie in einem Satz die Objekte der Klasse.
   Jedes Objekt stellt eine Währung dar und beinhaltet neben dem alphabetischen sowie dem numerischen Code die Anzahl der Standard-Nachkommastellen (z.B. 2 für die Währung Euro, da ein Euro äquivalent zu 100 Cent ist).
3. Ist die Klasse final?
   ja
4. Ist die Klasse abstrakt?
   nein
5. Welche Schnittstellen implementiert die Klasse (direkt und indirekt)?
   `Serializable`
6. Welche Oberklasse hat die Klasse?
   `Object`
7. Zu welchem Paket und Modul gehört die Klasse?
   Paket: `java.util`**, Modul:** `java.base`
8. Welche Unterklassen hat die Klasse?
   keine
9. Hat die Klasse öffentliche Klassenmethoden?
   ja (z.B. getAvailableCurrencies())
10. Hat die Klasse öffentliche Objektmethoden?
    ja (z.B. getDefaultFractionDigits())

11. Hat die Klasse öffentliche Konstruktoren?
    nein
12. Hat die Klasse öffentliche Klassenattribute?
    nein
13. Hat die Klasse öffentliche Objektattribute?
    nein
14. Mit welcher Anweisung kann man mithilfe der Methoden `getInstance()` und `getDisplayName()` den deutschen Namen der Währung mit dem Währungscode `"CZK"` erhalten?
    `Currency.getInstance("CZK").getDisplayName()`
15. Nennen Sie eine Methode der Klasse `Object`, die in `Currency` überschrieben wird. Inwiefern unterscheidet sich die `Currency`-Implementierung von der `Object`-Implementierung?
    `toString`: Statt Klassenname und Speicheradresse wird der ISO 4216 Währungscode als Zeichenkette zurückgegeben.

b) *(*, Die Klasse `AbstractSet<E>`, 4 Minuten)*

1. Wie lautet der vollständige Kopf der Klasse?
   `public abstract class AbstractSet<E> extends AbstractCollection<E> implements Set<E>`
2. Ist die Klasse final?
   nein
3. Ist die Klasse abstrakt?
   ja
4. Welche API-Unterklassen hat die Klasse?
   `ConcurrentSkipListSet, CopyOnWriteArraySet, EnumSet, HashSet, TreeSet`
5. Nennen Sie eine Methode der Klasse (vollen Methodenkopf angeben), die in einer Oberklasse deklariert wird.
   `public boolean add(E e)`
6. Ist die Methode `hashCode()` abstrakt?
   nein
7. Hat die Klasse finale Methoden?
   nein
8. Wie ist die `toString`-Methode der Klasse implementiert?
   Die String-Darstellung eines `AbstractSet<E>`-Objekts ist in der Klasse `AbstractCollection<E>` definiert und besteht aus den String-Darstellungen aller in ihm enthaltenen Elemente, getrennt durch Kommata und Leerzeichen (“ , ”) und umschlossen mit eckigen Klammern.

c) *(*, Die Klasse `AbstractList<E>`, 5 Minuten)*

1. Wie lautet der vollständige Prototyp der Klasse?
   `public abstract class AbstractList<E> extends AbstractCollection<E> implements List<E>`
2. Ist die Klasse final?
   nein
3. Ist die Klasse abstrakt?
   ja
4. Welche API-Unterklassen hat die Klasse?
   `AbstractSequentialList, ArrayList, Vector`
5. Nennen Sie eine abstrakte Methode der Klasse (vollen Prototyp angeben).
   `public abstract E get(int index)`
6. Ist die Methode `public boolean add(E e)` abstrakt?
   nein
7. Hat die Klasse finale Methoden?
   nein
8. Überschreibt die Klasse die Methode `toString`? Falls nicht: Aus welcher ihrer Oberklassen erbt sie die Implementierung?
   Die Klasse überschreibt `toString` nicht, sondern erbt die Implementierung von `AbstractCollection<E>`.
9. Wie ist die `equals`-Methode der Klasse implementiert?
   Ein `AbstractList<E>`-Objekt ist gleich einem Objekt `o`, falls `o` eine Liste ist (also vom Typ `List<E>` ist) und die gleichen Elemente in derselben Reihenfolge enthält.

d) *(*, Die Schnittstelle `Iterable<T>`, 3 Minuten)*

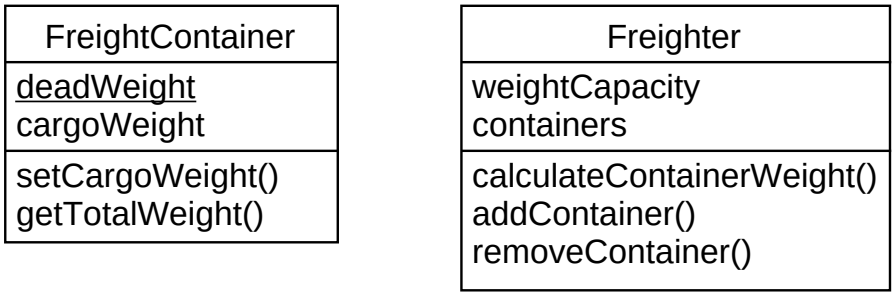
1. Wie lautet der vollständige Prototyp der Schnittstelle?
   `public interface Iterable<T>`
2. Gibt es Unterschnittstellen der Schnittstelle?
   ja
3. Gibt es Oberschnittstellen der Schnittstelle?
   nein
4. Nennen Sie eine abstrakte Methode der Schnittstelle (vollen Prototyp angeben).
   `Iterator<T> iterator()`
   (einzige richtige Antwort)
5. Hat die Schnittstelle statische Methoden?
   nein
6. Wie viele Default-Methoden hat die Schnittstelle?
   2

Aufgabe 10 *(Eigene Klasse & die Klasse `ArrayList`, 25 Minuten)*

In dieser Aufgabe lernen Sie die Klasse `ArrayList` kennen. Arrays wie z.B. `new String[5]` haben immer eine feste Länge, während `ArrayList`-Objekte ihre Größe dynamisch während der Programmlaufzeit anpassen. `ArrayList`s bieten viele weitere nützliche Funktionen an, die wir in Kapitel 5 näher kennen lernen werden. Für diese Aufgabe reicht der Hinweis, dass wir `ArrayList`-Variablen *parametrisieren müssen*: Wenn wir ein `ArrayList`-Objekt für `String`-Objekte erzeugen wollen, geht das mit der Anweisung `ArrayList<String> myStringList = new ArrayList<>();` Verwenden Sie die in Digicampus zur Verfügung gestellte Klasse `FreighterMain`, um Ihre Implementierung zu testen.

a) *(*, Klassenkarten entwerfen, 6 Minuten)*
Entwerfen Sie zwei Klassenkarten: `FreightContainer` für Frachtcontainer und `Freighter` für Frachtschiffe. Alle Frachtcontainer haben dasselbe Leergewicht von 3900.0. Außerdem haben Frachtcontainer ein reellwertiges Ladungsgewicht. Ein Frachtcontainer bietet mit `setCargoWeight()` eine Methode an, das Ladungsgewicht zu ändern und mit `getTotalWeight()` eine Methode, um sein Gesamtgewicht, bestehend aus Leergewicht + Ladungsgewicht, abzufragen.

Ein Frachtschiff hat ein maximales Gewicht, mit dem es beladen werden kann. Außerdem kann ein Frachtschiff keinen, einen oder mehrere Container geladen haben. Ein Frachtschiff bietet mit `addContainer()` bzw. `removeContainer()` Methoden an, einen Container zu entfernen bzw. hinzuzufügen. Außerdem bietet ein Frachtschiff mit
`calculateContainerWeight()`
eine Methode an, das Gesamtgewicht aller geladenen Container zu bestimmen.
Lösung:



b) *(*, `ArrayList` kennenlernen, 5 Minuten)*
Recherchieren Sie in der API vier Methoden der Klasse `ArrayList` und notieren Sie hier ihre Methodenköpfe:

- Eine Methode, um die Anzahl der enthaltenen Objekte zu bestimmen
  `public int size()`
- Eine Methode, um ein Objekt an einer bestimmten Position zu erhalten
  `public E get(int index)`
- Eine Methode, um ein Objekt hinzuzufügen
  `public boolean add(E e)`
- Eine Methode, um ein Objekt an einer bestimmten Position zu entfernen
  `public E remove(int index)`

Hinweis: In der API werden Sie bei der Klasse `ArrayList` bei ganz vielen Methoden den Typ `E` sehen. Das ist der Typ, mit dem das `ArrayList`-Objekt parametrisiert wurde. In unserem Fall erwartet es dann ein Frachtcontainer-Objekt bzw. gibt eines zurück usw.

c) *(**, Klassen implementieren, 14 Minuten)*
Implementieren Sie die Klassen für Frachtcontainer bzw. Frachtschiffe:

- Sorgen Sie dafür, dass Objekte der Klasse `FreightContainer` sowohl mit als auch ohne Angabe eines initialen Ladungsgewichts erzeugt werden können.
- Verwenden Sie für die Verwaltung der auf einem Frachtschiff geladenen Container eine `ArrayList` für `FreightContainer`-Objekte.

Stellen Sie sich die `ArrayList` wie einen Mitarbeiter vor, der für die Frachtcontainer zuständig ist und verwenden Sie die zuvor recherchierten Methoden:

- Um zu bestimmen, wie viel alle Container auf dem Schiff zusammen wiegen, fragen wir den zuständigen Mitarbeiter (also die ArrayList), wie viele Container wir haben und addieren die Gewichte der einzelnen Container.
- Wenn das Schiff einen Container entfernen soll, übergibt es den Container an den zuständigen Mitarbeiter, der diese Aufgabe übernimmt.
- Beim Hinzufügen eines Containers soll überprüft werden, ob der neue Container zu schwer ist: Nur, wenn das zukünftige Gesamtgewicht nicht oberhalb des Maximalgewichts des Schiffs liegen würde, darf der Container hinzugefügt werden. Verwenden Sie einen geeigneten Rückgabetypen, um klar zu machen, ob der Container hinzugefügt wurde oder nicht.

Lösung:

```
public class FreightContainer {
    private static final double deadWeight = 3900;
    private double cargoWeight;

    public FreightContainer() {
        cargoWeight = 0;
    }

    public FreightContainer(double cargoWeight) {
        this.cargoWeight = cargoWeight;
    }

    public double getTotalWeight() {
        return deadWeight + cargoWeight;
    }

    public void setCargoWeight(double cargoWeight) {
        this.cargoWeight = cargoWeight;
    }
}
```

Freighter

```java
import java.util.ArrayList;

public class Freighter {
    private double weightCapacity;
    private ArrayList<FreightContainer> containers = new ArrayList<>();

    public Freighter(double weightCapacity) {
        this.weightCapacity = weightCapacity;
    }

    public double calculateContainerWeight() {
        double sum = 0;
        for (int i = 0; i < containers.size(); i++) {
            sum += containers.get(i).getTotalWeight();
        }
        return sum;
    }

    public boolean addContainer(FreightContainer c) {
        if (calculateContainerWeight() + c.getTotalWeight() <= weightCapacity) {
            containers.add(c);
            return true;
        }
        return false;
    }

    public void removeContainer(int i) {
        containers.remove(i);
    }
}
```

Aufgabe 11 + 12 ** *(Mehrdimensionale Arrays und die Klasse `DecimalFormat`, 50 Minuten)*

In dieser Aufgabe sollen Sie zwei Datenklassen zur Verwaltung von Vektoren und zur Verwaltung von Matrizen mit reellwertigen Einträgen erstellen. Wir verwenden dafür folgende Notationen:

- $v = (v_i)_{i=1,\dots,n} = (v_1, \dots, v_n)$ ist die Notation für einen Vektor v der Länge n mit Einträgen $v_1, \dots, v_n$, z.B.
$$v = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$
- $a = (a_{i,j})_{i=1,\dots,n,j=1,\dots,m}$ ist die Notation für eine Matrix a mit n Zeilen und m Spalten mit Einträgen $a_{i,j}$, z.B. mit 2 Zeilen und 3 Spalten:
$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

a) *(Die Klasse `MyVector`, 18 Minuten)*
Implementieren Sie eine Klasse `MyVector` nach folgenden Vorgaben:

- Ein Vektor hat zwei Attribute: eines für seine Länge und eines für seine Einträge. Die Einträge verwalten Sie in einem Array.
- Sowohl die Länge als auch das Array können nicht verändert werden.
- Die Länge ist öffentlich verfügbar und ein Vektor stellt Methoden zur Verfügung, die Einträge an einem bestimmten Index abzufragen bzw. zu ändern.
- Ein Vektor stellt eine Methode zur Verfügung, seine Einträge auf zufällige Zahlen zwischen -1 und 1 zu setzen.

- Ein Vektor stellt eine Methode zur Verfügung, seine Norm zu berechnen. Die Norm eines Vektors v der Länge n ist die Wurzel der Summe der Quadrate seiner Einträge:

$$\| \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

  Anhand eines Beispiels:

$$\| \begin{pmatrix} 3 \\ 4 \end{pmatrix} \| = \sqrt{3^2 + 4^2} = 5$$

- Die Klasse soll die Schnittstelle `Comparable` implementieren: Ein Vektor u ist kleiner als ein Vektor v, wenn seine Norm kleiner als die von v ist.
- Die Zeichenkettendarstellung eines Vektors besteht aus der zeilenweisen Darstellung seiner Einträge mit mindestens 1 Vorkomma- und exakt 3 Nachkommastellen. Verwenden Sie ein `DecimalFormat`-Objekt, um diese Darstellung zu erreichen.

Lösung:

```java
import java.text.DecimalFormat;
import java.util.Random;

public class MyVector implements Comparable<MyVector> {
    public final int length;
    private final double[] v;

    public MyVector(int length) {
        this.length = length;
        v = new double[length];
    }

    public void randomize() {
        Random rng = new Random();
        for (int i = 0; i < this.length; i++) {
            setEntry(i, rng.nextDouble(-1, 1));
        }
    }

    public double getEntry(int i) {
        return v[i];
    }
    public void setEntry(int i, double value) {
        v[i] = value;
    }

    public double calculateNorm() {
        double sum = 0;
        for (int i = 0; i < length; i++) {
            sum += getEntry(i) * getEntry(i);
        }
        return Math.sqrt(sum);
    }

    @Override
    public int compareTo(MyVector o) {
        return Double.compare(this.calculateNorm(), o.calculateNorm());
    }
    @Override
    public String toString() {
        StringBuilder s = new StringBuilder();
        DecimalFormat df = new DecimalFormat("0.000");
        for (int i = 0; i < length; i++) {
            s.append(df.format(getEntry(i))).append("\n");
        }
        return s.toString();
    }
}
```

b) *(Die Klasse `MyMatrix`, 17 Minuten)*
Implementieren Sie eine Klasse `MyMatrix` nach folgenden Vorgaben:

- Eine Matrix hat drei Attribute: Eines für die Anzahl an Zeilen, eines für die Anzahl an Spalten und eines für die Einträge. Die Einträge verwalten Sie in einem mehrdimensionalen Array mit entsprechend vielen Zeilen und Spalten.
- Sowohl die Länge als auch das Array können nicht verändert werden.
- Zeilen- und Spaltenanzahl sind öffentlich verfügbar und eine Matrix stellt Methoden zur Verfügung, die Einträge an bestimmten Indizes abzufragen bzw. zu ändern.
- Eine Matrix stellt eine Methode zur Verfügung, ihre Einträge auf zufällige Zahlen zwischen -1 und 1 zu setzen.
- Die Klasse stellt eine Methode zur Verfügung, eine Matrix mit einem Vektor zu multiplizieren:
  - Entspricht die Anzahl der Spalten der Matrix nicht der Länge des Vektors, ist der Rückgabewert `null`.
  - Sonst ist das Ergebnis der Multiplikation einer Matrix mit einem Vektor ein Vektor mit einer Länge, die der Anzahl der Zeilen der Matrix entspricht.
  - für das Ergebnis $(v_i)_{i=1,\dots,n}$ der Multiplikation von $(a_{i,j})_{i=1,\dots,n,j=1,\dots,m}$ mit $(x_j)_{j=1,\dots,m}$ gilt:
    $$v_i = \sum_{j=1}^{m} a_{i,j} \cdot x_j$$
    Anhand eines Beispiels:
    $$\begin{pmatrix} 1 & 0 & 1 \\ 0 & -1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 1 \cdot 4 + 0 \cdot 5 + 1 \cdot 6 \\ 0 \cdot 4 + (-1) \cdot 5 + 3 \cdot 6 \end{pmatrix} = \begin{pmatrix} 10 \\ 13 \end{pmatrix}$$
- Die Zeichenkettendarstellung einer Matrix besteht aus der zeilenweisen Darstellung ihrer Zeilen, wobei die einzelnen Einträge durch Leerzeichen getrennt sind. Stellen Sie die Einträge mit mindestens 1 Vorkomma- und exakt 3 Nachkommastellen dar. Verwenden Sie ein `DecimalFormat`-Objekt, um diese Darstellung zu erreichen.
  Die Matrix aus dem obenstehenden Beispiel würde also wie folgt dargestellt werden:
  ```
  1.000 0.000 1.000
  0.000 -1.000 3.000
  ```

Lösung:

```java
import java.text.DecimalFormat;
import java.util.Random;

public class MyMatrix {
    public final int cols;
    public final int rows;
    private final double[][] m;

    MyMatrix(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
        m = new double[rows][cols];
    }

    public double getEntry(int row, int col) {
        return m[row][col];
    }

    public void setEntry(int row, int col, double value) {
        m[row][col] = value;
    }
```

```
    public void randomize() {
        Random rng = new Random();
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                setEntry(i, j, rng.nextDouble(-1, 1));
            }
        }
    }

    public static MyVector matrixTimesVector(MyMatrix m, MyVector x) {
        if (m.cols != x.length) return null;
        MyVector v = new MyVector(m.rows);
        for (int i = 0; i < v.length; i++) {
            double sum = 0;
            for (int j = 0; j < x.length; j++) {
                sum += m.getEntry(i, j) * x.getEntry(j);
            }
            v.setEntry(i, sum);
        }
        return v;
    }

    @Override
    public String toString() {
        StringBuilder s = new StringBuilder();
        DecimalFormat df = new DecimalFormat("0.000");
        for (int row = 0; row < rows; row++) {
            for (int col = 0; col < cols; col++) {
                s.append(df.format(getEntry(row, col))).append(" ");
            }
            s.append("\n");
        }
        return s.toString();
    }
}
```

c) *(*, Programmklasse, 10 Minuten)*
Implementieren Sie eine Programmklasse zum Testen Ihrer Datenklassen, die `main`-Methode soll

- einen Vektor u der Länge 3 und einen Vektor v der Länge 5 anlegen,
- den Inhalt beider Vektoren randomisieren,
- ausgeben, welcher der Vektoren größer ist oder ob beide Vektoren gleich sind,
- beide Vektoren ausgeben,
- eine Matrix m mit 3 Zeilen und 5 Spalten erzeugen und ihren Inhalt randomisieren,
- versuchen, m und u zu multiplizieren und das Ergebnis auszugeben bzw. ausgeben, wenn das nicht klappt und
- versuchen, m und v zu multiplizieren und das Ergebnis auszugeben bzw. ausgeben, wenn das nicht klappt.

Lösung:

```
public class MyMatrixVectorMain {
    public static void main(String[] args) {
        MyVector u = new MyVector(3);
        u.randomize();
        MyVector v = new MyVector(5);
        v.randomize();
        if (u.compareTo(v) < 0) {
            System.out.println("u is smaller than v");
        } else if (u.compareTo(v) > 0){
            System.out.println("u is bigger than v");
        } else {
            System.out.println("u is of the same size as v");
        }
```

```
        System.out.print("u:\n" + u);
        System.out.print("v:\n" + v);
        MyMatrix m = new MyMatrix(3, 5);
        m.randomize();
        System.out.println("m:\n" + m);
        if (MyMatrix.matrixTimesVector(m, u) != null) {
            System.out.println("m * u:\n" + MyMatrix.matrixTimesVector(m, u));
        } else {
            System.out.println("Sizes of m and u dont match!");
        }
        if (MyMatrix.matrixTimesVector(m, v) != null) {
            System.out.println("m * v:\n" + MyMatrix.matrixTimesVector(m, v));
        } else {
            System.out.println("Sizes of m and v dont match!");
        }
    }
}
```

d) *(*, Alternative Implementierung, 5 Minuten)*
Kopieren Sie Ihre Klasse `MyMatrix` und geben Sie ihr den Namen `MyMatrix2`. Passen Sie die Implementierung der Klasse so an, dass ein eindimensionales statt einem zweidimensionalen Array zum Speichern der Einträge verwendet wird. Ist `rows` die Anzahl der Zeilen und `cols` die Anzahl der Spalten, dann ist `rows * cols` die Länge des verwendeten Arrays. Dabei sollen die Zeilen nacheinander im Array abgelegt werden, für die Matrix

$$a = \begin{pmatrix} 4 & 1 & 3 & 0 \\ 2 & 1 & 7 & 6 \\ 9 & 5 & 3 & 7 \end{pmatrix}$$

sieht das Array also wie folgt aus:

4	1	3	0	2	1	7	6	9	5	3	7

Der Eintrag $a_{2,1} = 2$ befindet sich also an der Position $1 \cdot 4 + 1 = 5$ im Array, weil davor eine vollständige Zeile (4 Spalten) abgespeichert ist. Passen Sie die Implementierungen aller Methoden so an, dass ihre Funktionalität erhalten bleibt. Testen Sie Ihre neue Klasse, indem Sie in Ihrer Programmklasse `MyMatrix2` statt `MyMatrix` verwenden.
Anmerkung: Genau dieser Anwendungsfall ist der Grund, warum man auch innerhalb der Klasse immer getter und setter verwenden sollte, um auf Attribute zuzugreifen. So wird die interne Implementierung einer Klasse leichter austauschbar.
Lösung:

```
import java.text.DecimalFormat;
import java.util.Random;

public class MyMatrix2 {
    public final int cols;
    public final int rows;
    private final double[] m;

    MyMatrix2(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
        m = new double[rows * cols];
    }

    public double getEntry(int row, int col) {
        return m[row * cols + col];
    }

    public void setEntry(int row, int col, double value) {
        m[row * cols + col] = value;
    }

    /* restliche Implementierung bleibt unverändert! */
```