

Nachholklausur (6 ECTS): Lösungsvorschlag
Einführung in die Informatik:
Programmierung und Software-Entwicklung

Nachname
Vorname
Matrikelnummer
Studienfach
Angestrebter Abschluss
Fachsemester

Hilfsmittel jeder Art sind nicht zugelassen. Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf jedes Blatt. Die Klausurangabe mit allen Lösungen und alle verwendeten Blätter sind in **jedem Fall nach der Klausur wieder abzugeben**.

Diese Klausur soll gewertet werden: JA NEIN

Hinweis: Wenn keines der beiden Felder angekreuzt ist, wird JA angenommen.

Mit meiner Unterschrift erkläre ich die Richtigkeit und Vollständigkeit der obigen Angaben.

.....

Nicht von dem/der Studierenden auszufüllen

1	2	3	4	5	6	Σ	Note
von 9	von 7	von 12	von 11	von 14	von 7	von 60	

Aufgabe 1 EBNF-Grammatik und Syntaxdiagramm

3 + 6 = 9 Punkte

Gegeben sei folgende EBNF-Grammatik, die Namen mit voran gestelltem akademischen Grad im deutschen Sprachraum beschreibt.

```
Name = [ Titel ] Vorname Nachname
Titel = [ "Prof." ] "Dr." { "Dr." }
Vorname = Nachname
Nachname = Großbuchstabe { Kleinbuchstabe }
Großbuchstabe = "A" | "B" | ... | "Z"
Kleinbuchstabe = "a" | "b" | ... | "z"
```

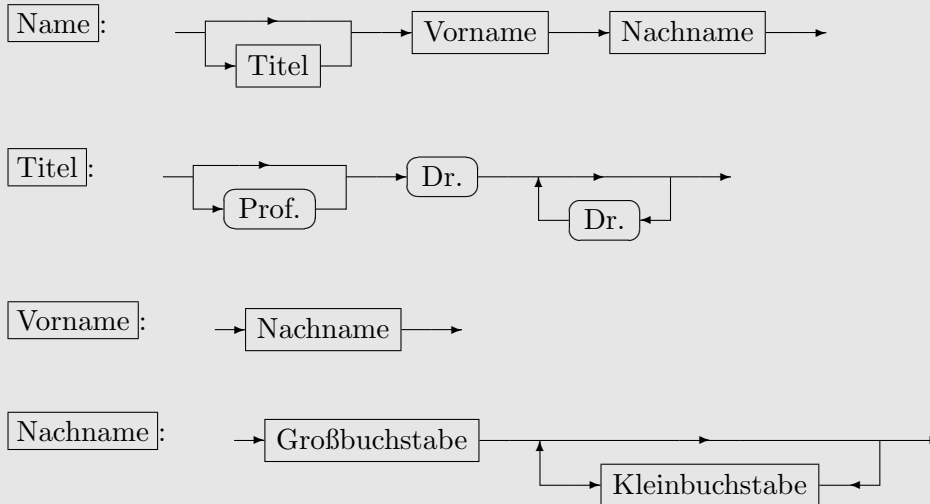
- a) Welche der folgenden Zeichenketten sind gültige Worte der oben definierten Sprache? Im Fall von nicht-gültigen Worten ist eine Begründung anzugeben.

Lösung:

Prof. Toni Maurer: nicht gültig, da man keinen Prof.-Titel ohne Dr.-Titel haben kann (1 Punkt)
Genoveva Bauer: gültig (0.5 Punkte)
Prof. Dr. Dr. Abcde Zxhast: gültig (0.5 Punkte)
Lisa müller: nicht gültig, da der Nachname mit einem Kleinbuchstaben beginnt (1 Punkt)

- b) Erstellen Sie zu der oben definierten EBNF-Grammatik ein äquivalentes Syntaxdiagramm. Diagramme für die Nichtterminale Großbuchstabe und Kleinbuchstabe brauchen nicht angegeben zu werden.

Lösung:



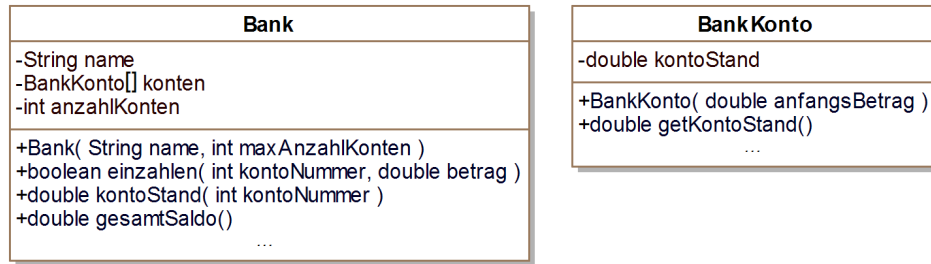
(nach Regeln: 1.5 Punkte, 2 Punkte, 1 Punkt, 1.5 Punkte)

Aufgabe 2

Ausdrücke in Java

7 Punkte

Wir verwenden die Klassen `Bank` und `BankKonto` aus der Vorlesung, die hier in UML-Notation angegeben sind.



Seien `Bank bank;` und `BankKonto konto;` lokale Variablendeklarationen (z.B. in einer `main`-Methode). Geben Sie für die folgenden Ausdrücke an, ob sie vom Java-Compiler akzeptiert werden oder nicht. Im positiven Fall ist der Typ des Ausdrucks anzugeben, im negativen Fall ist zu begründen, warum der Ausdruck zu einem Übersetzungsfehler führt.

a) `new Bank("MeineBank") == bank`

Lösung:

`new Bank("MeineBank") == bank` ist nicht korrekt, da der Konstruktor zwei Parameter erwartet. (1 Punkt)

b) `konto.kontoStand(5)`

Lösung:

`konto.kontoStand(5)` ist nicht korrekt, da das Attribut `kontoStand` nicht mit einem Parameter aufgerufen werden kann bzw. die Methode `kontoStand` nicht für die Klasse `BankKonto` definiert ist. (1 Punkt)

c) `bank.kontoStand(123) < konto.getKontoStand()`

Lösung:

`bank.kontoStand(123) < konto.getKontoStand()` ist korrekt und vom Typ `boolean`. (2 Punkte)

d) `new BankKonto(konto.getKontoStand())`

Lösung:

`new BankKonto(konto.getKontoStand())` ist korrekt und vom Typ `BankKonto`. (2 Punkte)

e) `bank.einzahlen(11, 300)`

Lösung:

`bank.einzahlen(11, 300)` ist korrekt und vom Typ `boolean`. (1 Punkt)

Aufgabe 3**Wiederholungsanweisungen**

12 Punkte

Der Igel möchte in einem Wettlauf gegen den Hasen antreten. Naturgemäß hat der Igel eine kürzere Schrittlänge als der Hase: der Igel hat eine Schrittlänge von 5.3 cm, der Hase hat eine Schrittlänge von 10.8 cm. Um diesen Vorteil des Hasens auszugleichen, bekommt der Igel einen Vorsprung von 270 cm.

Schreiben Sie eine `main`-Methode, die berechnet, nach wie vielen Schritten der Hase den Igel trotz des Vorsprungs überholt hat. Ihr Programm soll für jeden Schritt in einer neuen Zeile ausgeben, (1) wie weit der Hase insgesamt vorwärts gekommen ist und (2) wie weit der Igel zuzüglich seines Vorsprungs vorwärts gekommen ist. Diese Ausgabe soll letztmals erfolgen, wenn der Hase den Igel überholt hat.

Für die obigen Werte soll folgender Text auf der Kommandozeile ausgegeben werden:

```
Nach 1 Schritten: Hase 10.8cm, Igel 275.3cm
Nach 2 Schritten: Hase 21.6cm, Igel 280.6cm
Nach 3 Schritten: Hase 32.4cm, Igel 285.9cm
...
Nach 50 Schritten: Hase 540.0cm, Igel 535.0cm
```

Vervollständigen Sie die unten stehende `main`-Methode. Zunächst sind geeignete Variablen zu deklarieren und zu initialisieren.

Lösung:

Alternative 1:

```
1 public class HaseUndIgel {
2     public static void main(String[] args) {
3
4         // 1.5 Punkte: korrekte Init. aller Variablen
5         double schrittlaengeHase = 10.8;
6         double schrittlaengeIgel = 5.3;
7         double vorsprung = 270.0;
8
9         double streckeHase = 0.0; // 1 Punkt
10        double streckeIgel = vorsprung; // 1 Punkt
11
12        int schritte = 0; // 1 Punkt
13
14        while (streckeIgel >= streckeHase) {
15            // 0.5 Punkt: while mit richtiger Klammerung
16            // 1.5 Punkte: korrekter Abbruch
17                // 1.5 Punkte
18                streckeHase = streckeHase + schrittlaengeHase;
19                // 1.5 Punkte
20                streckeIgel = streckeIgel + schrittlaengeIgel;
21
22                schritte++; // 1.5 Punkte
23
24                // 1 Punkt
25                System.out.println("Nach " + schritte + " Schritten: Hase "
26                    + streckeHase + "cm, Igel " + streckeIgel + "cm");
27        }
28    }
29 }
```

-1 Punkt bei falscher Syntax

Alternative 2:

```
1 public class HaseUndIgel {
2     public static void main(String[] args) {
3
4         // 1.5 Punkte: korrekte Init. aller Variablen
5         double schrittlaengeHase = 10.8;
6         double schrittlaengeIgel = 5.3;
7         double vorsprung = 270.0;
8
9         int schritte = 0; // 0.5 Punkte
10        boolean igelNochVorne = true;
11
12        while (igelNochVorne) {
13            // 0.5 Punkt: while mit richtiger Klammerung
14            // 1.5 Punkte: korrekter Abbruch
15            // 2.5 Punkte
16            double streckeHase = schritte * schrittlaengeHase;
17            // 2.5 Punkte
18            double streckeIgel = vorsprung + schritte * schrittlaengeIgel;
19
20            schritte++; // 1 Punkt
21
22            // 1 Punkt
23            igelNochVorne = streckeHase <= streckeIgel;
24
25            // 1 Punkt
26            System.out.println("Nach " + schritte + " Schritten: Hase "
27                + streckeHase + "cm, Igel " + streckeIgel + "cm");
28        }
29    }
30 }
```

-1 Punkt bei falscher Syntax

- a) Eine Wetterstation misst jeden Tag um die gleiche Uhrzeit die Temperatur. Diese Daten werden der Reihe nach in einem Array vom Typ `double` gespeichert. Schreiben Sie eine Methode mit folgendem Kopf:

```
public static double groessterTemperaturSprung(double[] array)
```

Die Methode soll den größten Temperatursprung zwischen zwei aufeinander folgenden Tagen berechnen. Beispielweise soll für das Array `[-5.0, 1.7, 19.0, 16.9]` der Wert `17.3` ausgegeben werden und für das Array `[13.3, 12.0, 12.5]` der Wert `1.3`. Es kann davon ausgegangen werden, dass das Array mindestens zwei Elemente enthält.

Hinweis: Mit Hilfe der Methode `Math.abs` können Sie den Absolutbetrag einer Zahl berechnen. Beispielweise ergibt `Math.abs(-1.3)` den Wert `1.3`.

Lösung:

```
1 public static double groessterTemperaturSprung(double[] array) {
2     double groessterSprung = 0.0; // 0.5 Punkte
3     for (int i = 1; i < array.length; i++) {
4         // 1 Punkt für korrekte Verwendung der for-Schleife
5         // 1 Punkte für Initialisierung mit i=1
6         (oder nur bis array.lenght-1)
7         double differenz = Math.abs(array[i - 1] - array[i]);
8         // 1 Punkt für Differenzberechnung
9         // 1 Punkt für Absolutbetrag
10        if (differenz > groessterSprung) {
11            // 2 Punkte für gesamtes if-Konstrukt
12            groessterSprung = differenz;
13        }
14    }
15    return groessterSprung; // 0.5 Punkte
16 }
```

- b) Schreiben Sie eine `main`-Methode, in der für das Array `[-5.0, 1.7, 19.0, 16.9]` die Methode `groessterTemperaturSprung` aus Teilaufgabe a) aufgerufen wird und das Ergebnis auf der Kommandozeile ausgegeben wird.

Hinweis: Sie können davon ausgehen, dass die `main`-Methode zur selben Klasse gehört wie die Methode `groessterTemperaturSprung`.

Lösung:

```
1 public static void main(String[] args) {
2     double[] array = { -5.0, 1.7, 19.0, 16.9 }; // 1.5 Punkte
3     double wert = groessterTemperaturSprung(array); // 1.5 Punkte
4     System.out.println(wert); // 1 Punkt
5 }
```

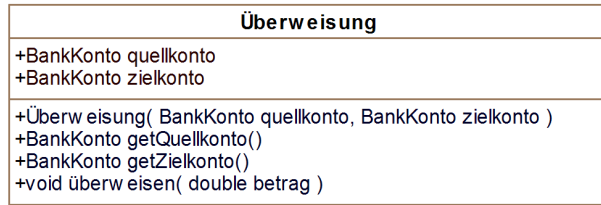
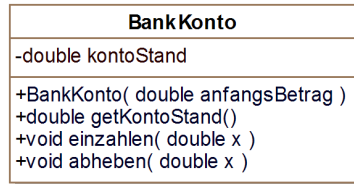
Aufgabe 5

Klassen und Vererbung

6 + 6 + 2 = 14 Punkte

Eine Bank braucht ein neues Verwaltungsprogramm für Überweisungen. Zusammen mit dem Bankmanager entwickeln Sie folgendes Programm.

- a) Für Ihr Programm wollen Sie die Klasse `BankKonto` aus der Vorlesung verwenden. Zusätzlich haben Sie eine Klasse `Überweisung` entworfen, die hier zusammen mit der Klasse `BankKonto` in UML-Notation angegeben ist.



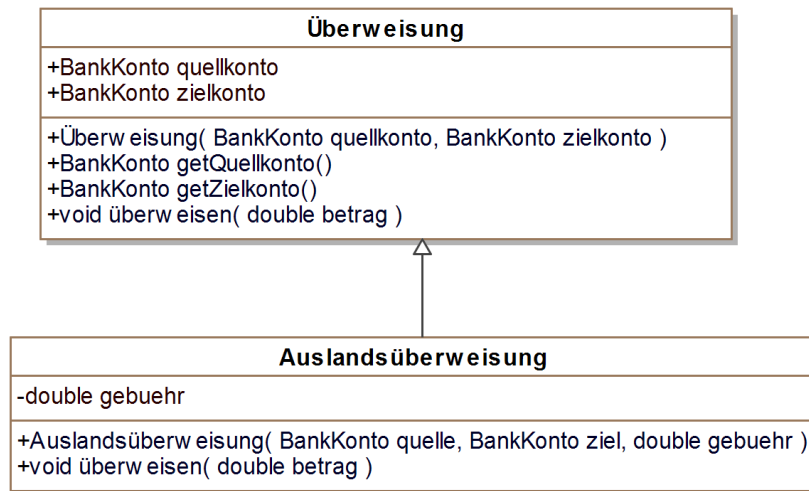
Mit einer Überweisung kann ein Geldbetrag von einem Konto auf ein anderes Konto verschoben werden. In der Klasse `Überweisung` wird das Quellkonto in dem öffentlichen Attribut `quellkonto` und das Zielkonto in dem öffentlichen Attribut `zielkonto` gespeichert. Beide Attribute werden mit dem angegebenen Konstruktor initialisiert. Für beide Attribute gibt es außerdem jeweils eine "Getter"-Methode. Die Überweisung von einem Geldbetrag `betrag` vom Quellkonto auf das Zielkonto wird durch die Methode `überweisen` ausgeführt. Dabei wird der Geldbetrag `betrag` vom Quellkonto abgehoben und auf das Zielkonto eingezahlt.

Implementieren Sie **nur** die Klasse `Überweisung` mit Konstruktor und allen Methoden in Java. Sie können davon ausgehen, dass die Klasse `BankKonto` in Java implementiert ist und können diese verwenden.

Lösung:

```
1 // 0.5 Punkte für private und public bei den Attributen und Methoden
2 public class Überweisung {
3
4     // 1 Punkt für alle Attribute
5     public BankKonto quellkonto;
6     public BankKonto zielkonto;
7
8     // 1 Punkt
9     public Überweisung(BankKonto quellkonto, BankKonto zielkonto) {
10         this.quellkonto = quellkonto;
11         this.zielkonto = zielkonto;
12     }
13
14     // 1.5 Punkte für beide Getter
15     public BankKonto getQuellkonto() {
16         return this.quellkonto;
17     }
18
19     public BankKonto getZielkonto() {
20         return this.zielkonto;
21     }
22
23     // 2 Punkte
24     public void überweisen(double betrag) {
25         this.quellkonto.abheben(betrag);
26         this.zielkonto.einzahlen(betrag);
27     }
28 }
29
30 -1 Punkt bei falscher Syntax oder falscher Klassendeklaration
```

- b) Die Bank möchte nun auch Überweisungen ins Ausland vornehmen können. Zur Überweisung zwischen zwei Konten unterschiedlicher Länder verwenden wir eine Subklasse **Auslandsüberweisung** der Klasse **Überweisung** aus Teilaufgabe a), die wie folgt in UML-Notation angegeben ist.



Für Auslandsüberweisungen wird eine Gebühr erhoben. Das Attribut **gebuehr** gibt an, wie hoch die Gebühr bei einer Überweisung ist. Im angegebenen Konstruktor werden alle geerbten und eigenen Attribute der Klasse **Auslandsüberweisung** initialisiert. Eine Überweisung wird durch die Methode **überweisen** der Klasse **Auslandsüberweisung** ausgeführt, wobei die Gebühr vom Quellkonto abgezogen wird.

Implementieren Sie die Klasse **Auslandsüberweisung** mit Konstruktor und der Methode **überweisen** in Java. Sie können davon ausgehen, dass die Klasse **BankKonto** und **Überweisung** aus Teilaufgabe a) bereits implementiert sind und können diese verwenden.

Lösung:

Alternative 1:

```
1 // 1 Punkt für extends
2 public class Auslandsüberweisung extends Überweisung {
3     // 0.5 Punkte
4     private double gebuehr;
5
6     public Auslandsüberweisung(BankKonto quellkonto, BankKonto zielkonto,
7         double gebuehr) {
8         this.quellkonto = quellkonto; // 0.5 Punkte
9         this.zielkonto = zielkonto; // 0.5 Punkte
10        this.gebuehr = gebuehr; // 0.5 Punkte
11    }
12
13    // 3 Punkte (mit 1 Punkt für Addition)
14    public void überweisen(double betrag) {
15        this.quellkonto.abheben(betrag + this.gebuehr);
16        this.zielkonto.einzahlen(betrag);
17    }
18 }
```


Alternative 2:

```
1 // 1 Punkt für extends
2 public class Auslandsüberweisung extends Überweisung {
3     // 0.5 Punkte
4     private double gebuehr;
5
6     public Auslandsüberweisung(BankKonto quellkonto,
7         BankKonto zielkonto, double gebuehr) {
8         super(quellkonto, zielkonto); // 1 Punkt
9         this.gebuehr = gebuehr; // 0.5 Punkte
10    }
11
12    // 3 Punkte
13    public void überweisen(double betrag) {
14        super.überweisen(betrag);
15        this.quellkonto.abheben(this.gebuehr);
16    }
17 }
```

-1 Punkt bei falscher Syntax oder falscher Klassendeklaration

- c) Die Klasse `BankMain` verwendet die Klassen aus den Teilaufgaben a) und b), die als gelöst vorausgesetzt werden können.

```
1 public class BankMain {
2     public static void main(String[] args) {
3         BankKonto quelle = new BankKonto(10);
4         BankKonto ziel = new BankKonto(0);
5
6         Überweisung überweisung =
7             new Auslandsüberweisung(quelle, ziel, 0.02);
8         überweisung.überweisen(5);
9
10        System.out.println("Quellkonto: " + quelle.getKontoStand());
11        System.out.println("Zielkonto: " + ziel.getKontoStand());
12    }
13 }
```

Geben Sie an, was in Zeile 10 **und** 11 genau auf der Konsole ausgegeben wird.

Lösung:

Quellkonto: 4.98 // 1 Punkt

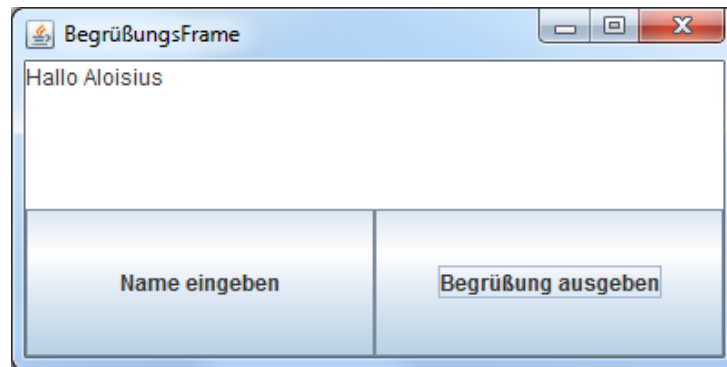
Zielkonto: 5.0 // 1 Punkt

Aufgabe 6

Grafische Benutzeroberflächen

2 + 5 = 7 Punkte

In dieser Aufgabe sollen Sie mit AWT/Swing eine grafische Benutzeroberfläche für eine kleine Anwendung implementieren, die den Benutzer begrüßt. Die grafische Benutzeroberfläche soll wie folgt aussehen:



Eine Implementierung der grafischen Benutzeroberfläche ist durch folgenden Programmausschnitt angedeutet:

```
1 public class BegrueessungsFrame extends JFrame implements ActionListener {
2     private JButton nameButton;
3     private JButton grussButton;
4     private JTextArea ausgabeBereich;
5
6     private String name;
7
8     public BegrueessungsFrame() {
9         this.setTitle("BegrüßungsFrame");
10        this.setSize(400, 200);
11
12        this.nameButton = new JButton("Name eingeben");
13        this.grussButton = new JButton("Begrüßung ausgeben");
14        this.ausgabeBereich = new JTextArea(150, 300);
15        ... // Der weitere Aufbau der GUI ist hier unwesentlich
16
17
18        ... // in Teilaufgabe a) zu implementieren
19
20
21        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22    }
23
24    public void actionPerformed(ActionEvent e) {
25        Object source = e.getSource();
26
27        ... // in Teilaufgabe b) zu implementieren
28
29    }
30 }
31 }
```

- a) Welche Anweisungen müssen im Konstruktor der Klasse `BgrueessungsFrame` eingeführt werden, so dass die Benutzeroberfläche auf Knopfdrücke reagieren kann.

Lösung:

```
1 this.nameButton.addActionListener(this); // 1 Punkt
2 this.grussButton.addActionListener(this); // 1 Punkt
```

- b) Wird der Button mit der Aufschrift “Name eingeben” gedrückt, soll der Benutzer mit Hilfe der Methode `JOptionPane.showInputDialog(String message)` nach seinem Namen gefragt werden. Dieser soll im Attribut `name` gespeichert werden. Wird der Button mit der Aufschrift “Begrüßung ausgeben” gedrückt, soll der Benutzer mit dem gespeicherten Namen im Ausgabebereich der Benutzeroberfläche mit “Hallo ...” begrüßt werden. Ergänzen Sie den Rumpf der Methode `actionPerformed` so, dass die oben gewünschte Reaktion für **beide** Buttons erfolgt.

Lösung:

```
1 public void actionPerformed(ActionEvent e) {
2     Object source = e.getSource();
3     if (source == this.nameButton) { // 1 Punkt
4         // 1 Punkt für Abfrage mit JOptionPane
5         // 1 Punkt für Zuweisung an this.name
6         this.name = JOptionPane.showInputDialog("Name eingeben:");
7     }
8     else if (source == this.grussButton) { // 1 Punkt
9         // 0.5 Punkte für richtige Benutzung des Ausgabebereichs
10        // 0.5 Punkte für richtige Benutzung von this.name
11        this.ausgabeBereich.setText("Hallo " + this.name);
12    }
13 }
```