Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. R. Hennicker, P. Wendler

WS 2017/18
Übungsblatt 10

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung: Lösungsvorschlag

Aufgabe 10-1 Prädikat für Arrays *Präsenz*

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches ein Prädikat auf Arrays berechnet.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:

Es soll einen Button mit der oben angegebenen Aufschrift geben. Darunter soll ein Ausgabebereich platziert werden.

Schreiben Sie eine Klasse `PraedikatFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `PraedikatFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `PraedikatFrame` abspeichern.

Lösung:
Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse `PraedikatFrame`, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse `JFrame` erbt. Ihre Klasse `PraedikatFrame` soll zwei Attribute haben:

- ein Attribut vom Klassentyp `JButton` für den Button,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über das Ergebnis dient.

Schreiben Sie einen Konstruktor, der den `PraedikatFrame` mit einem entsprechenden Titel und Größe (wir wählen hier 500x150 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Das Layout des `ContentPane` des `PraedikatFrame`s wird auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und der Button sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `PraedikatFrame` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

Weiter unten finden Sie eine Implementierung der Klassen `PraedikatFrame` und `PraedikatFrameMain` für die gesamte Aufgabe.

b) Erweitern Sie Ihre Klasse `PraedikatFrame` um eine Ereignisbehandlung für den Button. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach einem `char`-Array gefragt werden. Anschließend soll überprüft werden, ob das Array duplikatfrei ist, d.h. ob jedes Element nur einmal im Array vorkommt. Dazu soll eine statische Methode mit Kopf `private static boolean istArrayDuplikatfrei(char[] array)` geschrieben werden. Der Benutzer soll im Ausgabebereich über seine Eingabe und das Ergebnis des Tests informiert werden.

Hinweis: Die Klasse `KonverterErweitert.java` stellt Methoden `konvertiereZuCharArray` und `konvertiereZuString(char[] array)` zur Konvertierung zwischen `String` und `char[]` bereit.

Lösung: Algorithmus für das Prädikat `istArrayDuplikatfrei`:

Gehe das eingegebene Array `array` elementweise durch. Gehe für jede dieser Positionen `i` das Array ein zweites Mal durch, wobei dabei in jedem Schritt geprüft wird, ob das Element `array[i]` an einer späteren Position `j` vorkommt. Ist dies der Fall, gib `false` zurück. Ist dies für alle Positionen `i` nicht der Fall, gib `true` zurück.

Eine Implementierung der Klassen `PraedikatFrame` und `PraedikatFrameMain` finden Sie auch im ZIP-Archiv. Die Umsetzung des Algorithmus finden Sie ins Besondere in der Methode `istArrayDuplikatfrei`.

```
1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JOptionPane;
9  import javax.swing.JTextArea;
10
11 /**
12  * Diese Klasse repraesentiert das Hauptfenster der Praedikat-Praesenzaufgabe.
13  *
14  * @author Annabelle Klarl
15  */
16 public class PraedikatFrame extends JFrame implements ActionListener {
17     private JButton duplikatfreiButton;
18
19     private JTextArea ausgabeBereich;
20
21     /**
22      * In diesem Programmstueck wird das Fenster erzeugt
23      */
24     public PraedikatFrame() {
25         /* In der Kopfleiste des Fenster steht "PraedikatFrame" */
26         this.setTitle("PraedikatFrame");
27
28         /* Das Fenster ist 500 Pixel breit und 150 Pixel hoch. */
29         this.setSize(500, 150);
30
31         /* Hier werden alle Buttons erzeugt. */
32         this.duplikatfreiButton = new JButton(
33                 "Ist Array duplikatfrei?");
34
35         /* Hier wird der Ausgabe-Bereich erzeugt. */
36         this.ausgabeBereich = new JTextArea(10, 100);
37
38         /*
39          * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
40          * Interaktionselemente (wie eine TextArea oder ein Button) platziert
41          * werden koennen.
```

```
42          */
43         Container contentPane = this.getContentPane();
44         contentPane.setLayout(new GridLayout(2, 1));
45         /* Hier wird der Button platziert. */
46         contentPane.add(this.duplikatfreiButton);
47         /* Hier wird der Ausgabebereich platziert. */
48         contentPane.add(this.ausgabeBereich);
49
50         /*
51          * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
52          * der Buttons registriert.
53          */
54         this.duplikatfreiButton.addActionListener(this);
55
56         /*
57          * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
58          * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
59          */
60         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
61     }
62
63     @Override
64     public void actionPerformed(ActionEvent e) {
65         // je nach Button entsprechende Methode ausfuehren
66         Object source = e.getSource();
67         if (source == this.duplikatfreiButton) {
68             this.duplikatFrei();
69         }
70     }
71
72     /**
73      * implementiert die Funktionalitaet des "Duplikatfrei"-Buttons
74      */
75     private void duplikatFrei() {
76         String einlesenArray = JOptionPane
77                 .showInputDialog("char-Array: ");
78         char[] array = KonverterErweitert
79                 .konvertiereZuCharArray(einlesenArray);
80
81         boolean duplikatFrei = istArrayDuplikatfrei(array);
82
83         String eingabe = "Eingabe: " + einlesenArray + "\n\n";
84         String ergebnis;
85         if (duplikatFrei) {
86             ergebnis = eingabe + "Das Array ist duplikatfrei.";
87         } else {
88             ergebnis = eingabe + "Das Array ist NICHT duplikatfrei.";
89         }
90         this.ausgabeBereich.setText(ergebnis);
91     }
92
93     /**
94      * testet, ob das Array duplikatfrei ist
95      *
96      * @param array
97      * @return
98      */
99     private static boolean istArrayDuplikatfrei(char[] array) {
100        for (int i = 0; i < array.length; i++) {
101            for (int j = i + 1; j < array.length; j++) {
102                if (array[i] == array[j]) {
103                    return false;
104                }
105            }
106        }
```

```
107             return true;
108         }
109     }
```

```
1  /**
2   * Diese Klasse startet den PraedikatFrame
3   *
4   * @author Annabelle Klarl
5   */
6  public class PraedikatFrameMain {
7
8      /**
9       * Dieses Programmstueck startet das Programm.
10      *
11      * @param args
12      */
13     public static void main(String[] args) {
14         PraedikatFrame arrayFrame = new PraedikatFrame();
15         arrayFrame.setVisible(true);
16     }
17
18 }
```

c) Untersuchen die Zeitkomplexität und die Speicherplatzkomplexität Ihrer Methode `istArrayDuplikatfrei` im schlechtesten Fall und bestimmen Sie die Größenordnung der beiden Komplexitäten.

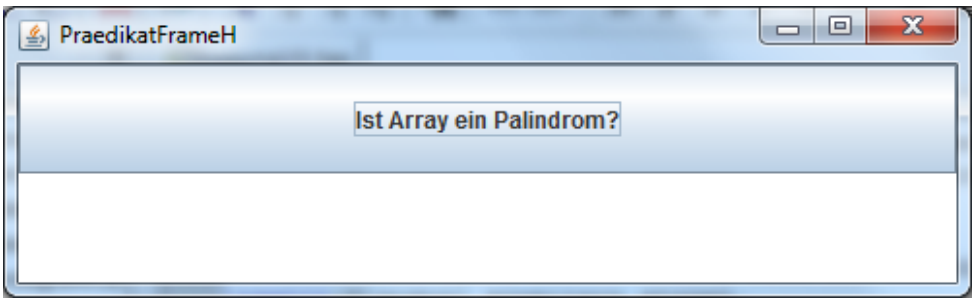
Lösung:
Im schlechtesten Fall werden die zwei geschachelte Schleifen komplett durchlaufen, daher ist die Ordnung der Zeitkomplexität $O(n^2)$, wobei n die Länge des Eingabearrays ist.

Neben dem Eingabearray der Länge n werden in jedem Fall nur zwei lokale Variablen `i` und `j` benötigt sowie ein Platz für die Rückgabe. Daher ist die Ordnung der Speicherplatzkomplexität $O(n)$.

Aufgabe 10-2 Prädikat für Arrays *Hausaufgabe*

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches ein Prädikat auf Arrays berechnet.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll einen Button mit der oben angegebenen Aufschrift geben. Darunter soll ein Ausgabebereich platziert werden.

Schreiben Sie eine Klasse `PraedikatFrameH`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `PraedikatFrameHMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `PraedikatFrameH` abspeichern.

Lösung:
Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse `PraedikatFrameH`, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse `JFrame` erbt. Ihre Klasse `PraedikatFrameH` soll zwei Attribute haben:

- ein Attribut vom Klassentyp `JButton` für den Button,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über das Ergebnis dient.

Schreiben Sie einen Konstruktor, der den `PraedikatFrameH` mit einem entsprechenden Titel und Größe (wir wählen hier 500x150 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Das Layout des `ContentPane` des `PraedikatFrameH`s wird auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und der Button sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `PraedikatFrameH` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

Weiter unten finden Sie eine Implementierung der Klassen `PraedikatFrameH` und `PraedikatFrameHMain`. für die gesamte Aufgabe.

b) Erweitern Sie Ihre Klasse `PraedikatFrameH` um eine Ereignisbehandlung für den Button. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach einem `char`-Array gefragt werden. Anschließend soll für dieses `char`-Array überprüft werden, ob es palindromisch ist, d.h. von hinten wie von vorne gelesen die gleiche Zeichenfolge enthält. Dazu soll eine statische Methode mit Kopf `private static boolean istPalindrom(char[] array)` verwendet werden. Der Benutzer soll im Ausgabebereich über seine Eingabe und das Ergebnis des Tests informiert werden.

Hinweis: Auf der Vorlesungswebseite finden Sie die Klasse `KonverterErweitert.java`, mit der Sie die Konvertierung zwischen `String` und `char[]` vornehmen können.

Lösung: Algorithmus für das Prädikat `istPalindrom`:

Gehe das eingegebene Array `array` elementweise bis zur Position `array.length/2 - 1` durch. Prüfe in jedem Schritt `i`, ob `array[i] != array[array.length - 1 - i]` gilt. Ist diese Bedingung erfüllt, gib `false` zurück. Ist diese Bedingungen für alle Schritte `i` nicht erfüllt, gib `true` zurück.

Eine Implementierung der Klassen `PraedikatFrameH` und `PraedikatFrameHMain` finden Sie auch im ZIP-Archiv. Die Umsetzung des Algorithmus finden Sie ins Besondere in der Methode `istPalindrom`.

```
1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JOptionPane;
9  import javax.swing.JTextArea;
10
11 /**
12  * Diese Klasse repraesentiert das Hauptfenster der Praedikat-Hausaufgabe.
13  *
14  * @author Annabelle Klarl
```

```
15  */
16 public class PraedikatFrameH extends JFrame implements ActionListener {
17     private JButton palindromButton;
18 
19     private JTextArea ausgabeBereich;
20 
21     /**
22      * In diesem Programmstueck wird das Fenster erzeugt
23      */
24     public PraedikatFrameH() {
25         /* In der Kopfleiste des Fenster steht "PraedikatFrameH" */
26         this.setTitle("PraedikatFrameH");
27 
28         /* Das Fenster ist 500 Pixel breit und 150 Pixel hoch. */
29         this.setSize(500, 150);
30 
31         /* Hier werden alle Buttons erzeugt. */
32         this.palindromButton = new JButton("Ist Array ein Palindrom?");
33 
34         /* Hier wird der Ausgabe-Bereich erzeugt. */
35         this.ausgabeBereich = new JTextArea(10, 100);
36 
37         /*
38          * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
39          * Interaktionselemente (wie eine TextArea oder ein Button) platziert
40          * werden koennen.
41          */
42         Container contentPane = this.getContentPane();
43         contentPane.setLayout(new GridLayout(2, 1));
44         /* Hier wird die Gruppe von Buttons platziert. */
45         contentPane.add(this.palindromButton);
46         /* Hier wird der Ausgabebereich platziert. */
47         contentPane.add(this.ausgabeBereich);
48 
49         /*
50          * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
51          * der Buttons registriert.
52          */
53         this.palindromButton.addActionListener(this);
54 
55         /*
56          * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
57          * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
58          */
59         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
60     }
61 
62     @Override
63     public void actionPerformed(ActionEvent e) {
64         // je nach Button entsprechende Methode ausfuehren
65         Object source = e.getSource();
66         if (source == this.palindromButton) {
67             this.palindrom();
68         }
69     }
70 
71     /**
72      * implementiert die Funktionalitaet des "Palindrom"-Buttons
73      */
74     private void palindrom() {
75         String einlesenArray = JOptionPane
76                 .showInputDialog("char-Array: ");
77         char[] array = KonverterErweitert
78                 .konvertiereZuCharArray(einlesenArray);
79 
```

```
80          boolean istPalindrom = istPalindrom(array);
81
82          String eingabe = "Eingabe: " + einlesenArray + "\n\n";
83          String ergebnis;
84          if (istPalindrom) {
85              ergebnis = eingabe + "Das Array ist ein Palindrom.";
86          } else {
87              ergebnis = eingabe + "Das Array KEIN Palindrom.";
88          }
89          this.ausgabeBereich.setText(ergebnis);
90      }
91
92      /**
93       * testet, ob das Array palindromisch angeordnet ist
94       *
95       * @param array
96       * @return
97       */
98      private static boolean istPalindrom(char[] array) {
99          for (int i = 0; i < array.length / 2; i++) {
100             if (array[i] != array[array.length - 1 - i]) {
101                 return false;
102             }
103         }
104         return true;
105     }
106 }
```

```
1  /**
2   * Diese Klasse startet den PraedikatFrameH
3   *
4   * @author Annabelle Klarl
5   */
6  public class PraedikatFrameHMain {
7
8      /**
9       * Dieses Programmstueck startet das Programm.
10      *
11      * @param args
12      */
13     public static void main(String[] args) {
14         PraedikatFrameH arrayFrame = new PraedikatFrameH();
15         arrayFrame.setVisible(true);
16     }
17
18 }
```

c) Untersuchen die Zeitkomplexität und die Speicherplatzkomplexität Ihrer Methode `istPalindrom` und bestimmen Sie die Größenordnung der beiden Komplexitäten.

Lösung:
Das Eingabearray wird in einer Schleife bis zur Hälfte seiner Länge durchlaufen. Daher ist die Ordnung der Zeitkomplexität $O(n)$, wobei n die Länge des Eingabearrays ist.

Neben dem Eingabearray der Länge n wird in jedem Fall nur eine lokale Variable `i` benötigt sowie ein Platz für die Rückgabe. Daher ist die Ordnung der Speicherplatzkomplexität $O(n)$.

Aufgabe 10-3 **Arrays von Objekten** *Präsenz*

In dieser Aufgabe sollen Sie den Umgang mit Arrays von Objekten üben. Dazu werden Sie ein Verwaltungsprogramm schreiben, mit dem eine Klausur angelegt werden kann, Studenten zu dieser Klausur angemeldet, Noten vergeben und verschiedene Informationen über die Klausur und die Teilnehmer ausgegeben werden können.

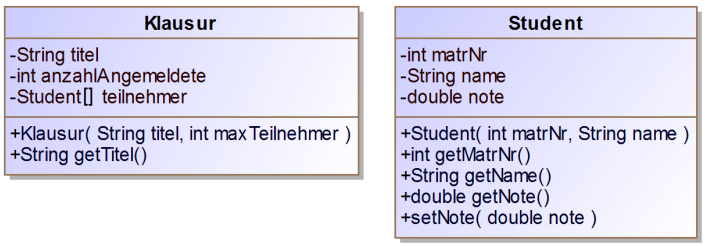
a) Schreiben Sie eine Klasse `Student`, die einen Teilnehmer an einer Klausur repräsentiert. Für einen Studenten sollen seine Matrikelnummer, sein Name und eine Note gespeichert werden können.

- Der Konstruktor der Klasse `Student` benötigt nur Matrikelnummer und Name des Studenten zur Erzeugung eines neuen Objekts. Die Note des Studenten wird standardmäßig mit -1.0 initialisiert.
- Schreiben Sie für jedes Attribut der Klasse einen "Getter", der den Wert des jeweiligen Attributs zurück gibt.
- Schreiben Sie außerdem für das Attribut `note` einen "Setter", mit dem einem Studenten eine Note zugewiesen werden kann.

Schreiben Sie eine Klasse `Klausur`, die alle teilnehmenden Studenten speichert. Jedes `Klausur`-Objekt soll einen Titel haben und speichern können, wie viele und welche Studenten an dieser Klausur teilnehmen.

- Der Konstruktor der Klasse `Klausur` benötigt den Titel der Klausur und wie viele Studenten maximal an der Klausur teilnehmen können, um ein partielles Array geeignet initialisieren zu können.
- Schreiben Sie für das Attribut `titel` einen "Getter", der den Wert des Attribut zurück gibt.

Im folgenden UML-Diagramm sind die Eigenschaften der Klassen grafisch dargestellt.



Eine Implementierung der Klassen `Student` und `Klausur` finden Sie auch im ZIP-Archiv.

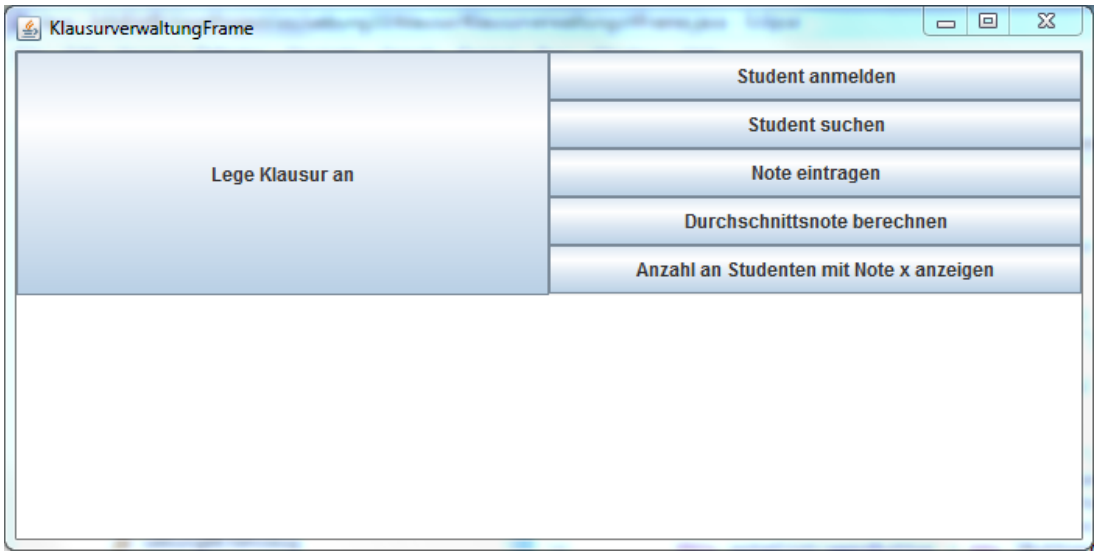
```
1  /**
2   *
3   * Repraesentation eines Studenten mit Matrikelnummer, Name und Note.
4   *
5   * @author Annabelle Klarl
6   */
7  public class Student {
8      private int matrNr;
9      private String name;
10     private double note;
11
12     public Student(int matrNr, String name) {
13         this.matrNr = matrNr;
14         this.name = name;
15         this.note = -1.0;
16     }
```

```
17
18      public int getMatrNr() {
19          return this.matrNr;
20      }
21
22      public String getName() {
23          return this.name;
24      }
25
26      public double getNote() {
27          return this.note;
28      }
29
30      public void setNote(double note) {
31          this.note = note;
32      }
33  }
```

```
1  /**
2   * Repraesentation einer Klausur mit einem Titel und einer Liste von angemeldeten
3   * Studenten.
4   *
5   * @author Annabelle Klarl
6   *
7   */
8  public class Klausur {
9      private String titel;
10     private Student[] teilnehmer;
11     private int anzahlAngemeldete;
12
13     /**
14      * Konstruktor
15      *
16      * @param titel
17      *             Titel der Klausur
18      * @param maxTeilnehmer
19      *             maximale Anzahl an Teilnehmern
20      */
21     public Klausur(String titel, int maxTeilnehmer) {
22         this.titel = titel;
23         this.teilnehmer = new Student[maxTeilnehmer];
24         this.anzahlAngemeldete = 0;
25     }
26
27     /**
28      * Diese Methode liefert den Titel der Klausur
29      *
30      * @return
31      */
32     public String getTitel() {
33         return this.titel;
34     }
35
36     /**
37      * Diese Methode meldet einen Studenten mit der gegebenen Matrikelnummer und
38      * dem gegebenen Namen an. Dazu wird zunaechst ein neues Objekt der Klasse
39      * {@link Student} erzeugt, dieser an der Klausur angemeldet und true
40      * zurueckgegeben. Ist die Klausur schon voll (d.h. wird die Maximalanzahl an
41      * Teilnehmern ueberschritten), wird der Student nicht angemeldet und false
42      * zurueckgegeben.
43      *
44      * @param matrNr
45      * @param name
46      * @return false falls die Maximalanzahl an Teilnehmern ueberschritten wurde,
47      *         true sonst
```

```
48       */
49      public boolean anmelden(int matrNr, String name) {
50          if (this.anzahlAngemeldete < this.teilnehmer.length) {
51              this.teilnehmer[this.anzahlAngemeldete] = new Student(
52                      matrNr, name);
53              this.anzahlAngemeldete++;
54              return true;
55          }
56          return false;
57      }
58
59      /**
60       * Diese Methode sucht in der Liste an Teilnehmern der Klausur einen
61       * Studenten mit der gegebenen Matrikelnummer. Wird ein Student gefunden,
62       * wird dieser zurueckgegeben. Falls kein Student mit dieser Matrikelnummer
63       * angemeldet ist, wird null zurueckgegeben.
64       *
65       * @param matrNr
66       * @return das Objekt der Klasse Student mit der gegebenen Matrikelnummer;
67       *         null falls kein Student mit dieser Matrikelnummer gefunden wird.
68       */
69      public Student sucheStudent(int matrNr) {
70          for (int i = 0; i < this.anzahlAngemeldete; i++) {
71              Student aktuellerStudent = this.teilnehmer[i];
72              if (aktuellerStudent.getMatrNr() == matrNr) {
73                  return aktuellerStudent;
74              }
75          }
76          return null;
77      }
78
79      /**
80       * Diese Methode traegt fuer einen Studenten mit der gegebenen Matrikelnummer
81       * die gegebene Note ein. Falls ein Student mit dieser Matrikelnummer an der
82       * Klausur angemeldet ist, wird die Note eingetragen und true zurueckgegeben.
83       * Falls kein Student angemeldet ist, wird false zurueckgegeben.
84       *
85       * @param matrNr
86       * @param note
87       * @return true falls ein Student mit der gegebenen Matrikelnummer an der
88       *         Klausur angemeldet ist (und die Note eingetragen wurde); false
89       *         sonst
90       */
91      public boolean noteEintragen(int matrNr, double note) {
92          Student student = this.sucheStudent(matrNr);
93          if (student != null) {
94              student.setNote(note);
95              return true;
96          }
97          return false;
98      }
99  }
```

b) Schreiben Sie eine grafische Benutzeroberfläche zur Verwaltung einer Klausur, die wie folgt aussehen soll:



Es soll einen Button geben, um eine Klausur anzulegen, sowie weitere fünf Buttons um Verwaltungsaufgaben an dieser Klausur vorzunehmen. Die fünf Buttons sind in fünf Zeilen übereinander angeordnet. Der Button zum Anlegen einer Klausur ist in einer linken Spalte platziert, während die Gruppe von fünf Buttons in der rechten Spalte platziert ist. Darunter gibt es einen Ausgabebereich, in dem der Benutzer über die Auswirkungen seiner Verwaltungsaktionen informiert werden soll.

Schreiben Sie eine Klasse `KlausurverwaltungsFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `KlausurverwaltungsFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `KlausurverwaltungsFrame` abspeichern.

Lösung:
Implementieren Sie den Frame analog zu den vorherigen Aufgaben. Beachten Sie dabei allerdings die Gruppierung der Buttons:

- Gruppieren Sie die fünf Verwaltungsbuttons in einem `JPanel studentenPanel`, dem Sie ein `GridLayout` mit einer Spalte und fünf Zeilen zuordnen.
- Gruppieren Sie den Button zum Klausur Anlegen und den `studentenPanel` in einem `JPanel buttonpanel`, dem Sie ein `GridLayout` mit zwei Spalten und einer Zeile zuordnen.
- Ordnen Sie den `buttonPanel` und den Ausgabebereich auf dem `ContentPane` an, dem Sie ein `GridLayout` mit einer Spalte und zwei Zeilen geben.

Beachten Sie außerdem, dass Sie hier ein weiteres Attribut benötigen, um die aktuelle Klausur zu speichern!

Weiter unten finden Sie eine Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` für die gesamte Aufgabe.

c) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Anlegen einer Klausur**. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` gefragt werden, welchen Titel die Klausur haben soll und wie viele Teilnehmer maximal an der Klausur teilnehmen können. Anschließend soll eine entsprechende Klausur angelegt und gespeichert werden sowie der Benutzer darüber in-

formiert werden. Wurde zuvor schon eine Klausur angelegt, soll keine neue Klausur angelegt werden und der Benutzer auch darüber informiert werden.

Weiter unten finden Sie eine Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` für die gesamte Aufgabe.

d) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Anmelden eines Studenten**. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` gefragt werden, welche Matrikelnummer und welchen Namen der Student hat. Anschließend soll ein entsprechender Student erzeugt werden und an der Klausur angemeldet werden (d.h. als Teilnehmer zur Klausur hinzugefügt werden) sowie der Benutzer darüber informiert werden. Existiert noch keine Klausur, soll eine Fehlermeldung ausgegeben werden.

Hinweis: Beachten Sie, dass Sie bei dieser Aufgabe auch die Klasse `Klausur` erweitern müssen.

Weiter unten finden Sie eine Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` für die gesamte Aufgabe.

e) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Suchen eines Studenten**. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` gefragt werden, nach welcher Matrikelnummer er suchen möchte. Anschließend soll in der aktuellen Klausur nach einem Teilnehmer mit dieser Matrikelnummer gesucht werden und der Benutzer über das Ergebnis der Suche informiert werden. Existiert noch keine Klausur, soll eine Fehlermeldung ausgegeben werden. Ebenso soll eine Fehlermeldung ausgegeben werden, wenn es keinen Teilnehmer mit der gegebenen Matrikelnummer in der Klausur gibt.

Hinweis: Beachten Sie, dass Sie bei dieser Aufgabe auch die Klasse `Klausur` erweitern müssen.

Weiter unten finden Sie eine Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` für die gesamte Aufgabe.

f) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Eintragen einer Note**. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` gefragt werden, für welchen Studenten (Matrikelnummer) er welche Note eintragen möchte. Anschließend soll in der aktuellen Klausur nach einem Teilnehmer mit dieser Matrikelnummer gesucht werden, die Note bei diesem Teilnehmer eingetragen werden und der Benutzer informiert werden, ob die Benotung erfolgreich war. Existiert noch keine Klausur, soll eine Fehlermeldung ausgegeben werden. Ebenso soll eine Fehlermeldung ausgegeben werden, wenn es keinen Teilnehmer mit der gegebenen Matrikelnummer in der Klausur gibt.

Hinweis: Beachten Sie, dass Sie bei dieser Aufgabe auch die Klasse `Klausur` erweitern müssen. Benutzen Sie außerdem Ihre zuvor implementierte Methode zum Suchen eines Studenten.

Eine Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` finden Sie auch im ZIP-Archiv.

```java
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

/**
 * Eine grafische Benutzeroberflaeche zur Verwaltung von Klausuren.
 *
 * @author Annabelle Klarl
 *
 */
public class KlausurverwaltungsFrame extends JFrame implements
        ActionListener {

    private JButton klausurAnlegenButton;
    private JButton studentAnmeldenButton;
    private JButton studentSuchenButton;
    private JButton noteEintragenButton;

    /* Hausaufgabe */
    private JButton durchschnittsnoteButton;
    private JButton studentenProNoteButton;

    private JTextArea ausgabeBereich;

    private Klausur aktuelleKlausur;

    /**
     * In diesem Programmstueck wird das Fenster erzeugt.
     */
    public KlausurverwaltungsFrame() {
        /* In der Kopfleiste des Fenster steht "KlausurverwaltungFrame" */
        this.setTitle("KlausurverwaltungFrame");

        /* Das Fenster ist 700 Pixel breit und 350 Pixel hoch. */
        this.setSize(700, 350);

        /* Hier werden alle Buttons erzeugt. */
        this.klausurAnlegenButton = new JButton("Lege Klausur an");
        this.studentAnmeldenButton = new JButton("Student anmelden");
        this.studentSuchenButton = new JButton("Student suchen");
        this.noteEintragenButton = new JButton("Note eintragen");

        /* Hausaufgabe */
        this.durchschnittsnoteButton = new JButton(
                "Durchschnittsnote berechnen");
        this.studentenProNoteButton = new JButton(
                "Anzahl an Studenten mit Note x anzeigen");

        /* Hier wird der Ausgabe-Bereich erzeugt. */
        this.ausgabeBereich = new JTextArea(10, 100);

        /* Hier werden alle Buttons zusammengruppiert. */
        JPanel studentenPanel = new JPanel();
        studentenPanel.setLayout(new GridLayout(5, 1));
        studentenPanel.add(this.studentAnmeldenButton);
        studentenPanel.add(this.studentSuchenButton);
        studentenPanel.add(this.noteEintragenButton);
```

```
65
66          /* Hausaufgabe */
67          studentenPanel.add(this.durchschnittsnoteButton);
68          studentenPanel.add(this.studentenProNoteButton);
69
70          JPanel buttonPanel = new JPanel();
71          buttonPanel.setLayout(new GridLayout(1, 2));
72          buttonPanel.add(this.klausurAnlegenButton);
73          buttonPanel.add(studentenPanel);
74
75          /*
76           * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
77           * Interaktionselemente (wie eine TextArea oder ein Button) platziert
78           * werden koennen.
79           */
80          Container contentPane = this.getContentPane();
81          contentPane.setLayout(new GridLayout(2, 1));
82          /* Hier wird die Gruppe von Buttons platziert. */
83          contentPane.add(buttonPanel);
84          /* Hier wird der Ausgabebereich platziert. */
85          contentPane.add(this.ausgabeBereich);
86
87          /*
88           * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
89           * Button registriert.
90           */
91          this.klausurAnlegenButton.addActionListener(this);
92          this.studentAnmeldenButton.addActionListener(this);
93          this.studentSuchenButton.addActionListener(this);
94          this.noteEintragenButton.addActionListener(this);
95
96          /*
97           * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
98           * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
99           */
100         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
101     }
102
103     /**
104      * Dieses Programmstueck wird immer dann ausgefuehrt, wenn ein Benutzer auf
105      * einen Button drueckt.
106      */
107     @Override
108     public void actionPerformed(ActionEvent e) {
109         Object source = e.getSource();
110         if (source == this.klausurAnlegenButton) {
111             this.klausurAnlegen();
112         } else if (source == this.studentAnmeldenButton) {
113             this.studentAnmelden();
114         } else if (source == this.studentSuchenButton) {
115             this.studentSuchen();
116         } else if (source == this.noteEintragenButton) {
117             this.noteEintragen();
118         }
119     }
120
121     /**
122      * Diese Methode erzeugt eine neue Klausur, falls es momentan keine gibt.
123      */
124     private void klausurAnlegen() {
125         if (this.aktuelleKlausur != null) {
126             this.ausgabeBereich
127                     .setText("Es wurde bereits eine Klausur mit dem Titel "
128                             + this.aktuelleKlausur.getTitel()
129                             + " angelegt.");
```

```
130         } else {
131             String einlesenTitel = JOptionPane
132                     .showInputDialog("Titel der Klausur: ");
133 
134             String einlesenMaxTeilnehmer = JOptionPane
135                     .showInputDialog("Maximale Anzahl an Teilnehmern: ");
136             int maxTeilnehmer = Integer
137                     .parseInt(einlesenMaxTeilnehmer);
138 
139             // !!!!
140             this.aktuelleKlausur = new Klausur(einlesenTitel,
141                     maxTeilnehmer);
142 
143             this.ausgabeBereich
144                     .setText("Folgende Klausur wurde angelegt: "
145                             + this.aktuelleKlausur
146                             + " mit dem Titel "
147                             + this.aktuelleKlausur.getTitel());
148         }
149     }
150 
151     /**
152      * Diese Methode meldet einen Studenten zur Klausur an, falls es momentan
153      * eine Klausur gibt.
154      */
155     private void studentAnmelden() {
156         if (this.aktuelleKlausur == null) {
157             this.ausgabeBereich
158                     .setText("Es wurde noch keine Klausur angelegt.");
159         } else {
160             String einlesenMatrNr = JOptionPane
161                     .showInputDialog("Matrikelnummer: ");
162             int matrNr = Integer.parseInt(einlesenMatrNr);
163 
164             String name = JOptionPane.showInputDialog("Name: ");
165 
166             // !!! Methodenaufruf
167             boolean angemeldet = this.aktuelleKlausur.anmelden(
168                     matrNr, name);
169             if (angemeldet) {
170                 this.ausgabeBereich.setText("Der Student " + name
171                         + " mit der Matrikelnummer " + matrNr
172                         + " wurde angemeldet.");
173             } else {
174                 this.ausgabeBereich
175                         .setText("Der Student konnte nicht angemeldet werden, "
176                                 + "da die Maximalanzahl an Teilnehmern "
177                                 + "ueberschritten wurde.");
178             }
179         }
180     }
181 
182     /**
183      * Diese Methode gibt Informationen ueber einen an der Klausur angemeldeten
184      * Studenten aus, falls es momentan eine Klausur gibt und ein Student mit
185      * der gegebenen Matrikelnummer angemeldet ist.
186      */
187     private void studentSuchen() {
188         if (this.aktuelleKlausur == null) {
189             this.ausgabeBereich
190                     .setText("Es wurde noch keine Klausur angelegt.");
191         } else {
192             String einlesenMatrNr = JOptionPane
193                     .showInputDialog("Matrikelnummer: ");
194             int matrNr = Integer.parseInt(einlesenMatrNr);
```

```
195
196             // !!! Methodenaufruf
197             Student student = this.aktuelleKlausur
198                     .sucheStudent(matrNr);
199             if (student == null) {
200                 this.ausgabeBereich
201                         .setText("Der Student konnte nicht gefunden werden.");
202             } else {
203                 this.ausgabeBereich
204                         .setText("Folgender Student wurde gefunden: "
205                                 + student.getName()
206                                 + " mit der Matrikelnummer "
207                                 + student.getMatrNr()
208                                 + " und der Note "
209                                 + student.getNote());
210             }
211         }
212     }
213
214     /**
215      * Diese Methode traegt eine Note fuer einen Studenten ein, falls es momentan
216      * eine Klausur gibt und ein Student mit der gegebene Matrikelnummer
217      * angemeldet ist.
218      */
219     private void noteEintragen() {
220         if (this.aktuelleKlausur == null) {
221             this.ausgabeBereich
222                     .setText("Es wurde noch keine Klausur angelegt.");
223         } else {
224             String einlesenMatrNr = JOptionPane
225                     .showInputDialog("Matrikelnummer: ");
226             int matrNr = Integer.parseInt(einlesenMatrNr);
227
228             String einlesenNote = JOptionPane
229                     .showInputDialog("Note: ");
230             double note = Double.parseDouble(einlesenNote);
231
232             // !!! Methodenaufruf
233             boolean benotet = this.aktuelleKlausur.noteEintragen(
234                     matrNr, note);
235             if (benotet) {
236                 this.ausgabeBereich
237                         .setText("Der Student mit der Matrikelnummer "
238                                 + matrNr
239                                 + " wurde mit der Note "
240                                 + note + " benotet.");
241             } else {
242                 this.ausgabeBereich
243                         .setText("Der Student konnte nicht benotet werden, "
244                                 + "da kein Student mit der Matrikelnummer "
245                                 + matrNr + " gefunden werden konnte.");
246             }
247         }
248     }
249 }
```

```
1 /**
2  * Diese Klasse startet den KlausurverwaltungFrame
3  *
4  * @author Annabelle Klarl
5  *
6  */
7 public class KlausurverwaltungsFrameMain {
8
9     /**
```

```
10       * Dieses Programmstueck startet das Programm.
11       *
12       * @param args
13       */
14      public static void main(String[] args) {
15          KlausurverwaltungsFrame frame = new KlausurverwaltungsFrame();
16          frame.setVisible(true);
17      }
18
19  }
```

Aufgabe 10-4 **Arrays von Objekten** *Hausaufgabe*

In dieser Aufgabe werden Sie Ihr Verwaltungsprogramm aus Aufgabe 10-3 um zusätzliche Funktionen erweitern. Nehmen Sie daher für alle folgenden Teilaufgaben die Klassen `Student`, `Klausur` und `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain` aus der Lösung von Aufgabe 10-3 zur Grundlage.

a) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Berechnen der Durchschnittsnote**. Wird dieser Button gedrückt, wird die Durchschnittsnote aller Teilnehmer der Klausur berechnet. Beachten Sie, dass Sie dabei nur den Durchschnitt aller schon benoteter Teilnehmer berechnen. Der Benutzer soll anschließend über die Durchschnittsnote informiert werden. Existiert noch keine Klausur, soll eine Fehlermeldung ausgegeben werden. Ebenso soll eine Fehlermeldung ausgegeben werden, wenn bisher kein Teilnehmer eine Note erhalten hat.

Hinweis: Beachten Sie, dass Sie bei dieser Aufgabe auch die Klasse `Klausur` erweitern müssen.

Weiter unten finden Sie eine Implementierung der Klassen `KlausurH`, `KlausurverwaltungsHFrame` und `KlausurverwaltungsHFrameMain` für die gesamte Aufgabe.

b) Erweitern Sie Ihre Klasse `KlausurverwaltungsFrame` um eine Ereignisbehandlung für den Button zum **Anzeigen der Anzahl aller Studenten mit einer bestimmten Note**. Wird dieser Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` gefragt werden, für welche Note er die Anzahl an Studenten wissen möchte. Anschließend soll in der aktuellen Klausur gesucht werden, wie viele Studenten diese Note erreicht haben und der Benutzer darüber informiert werden. Existiert noch keine Klausur, soll eine Fehlermeldung ausgegeben werden.

Hinweis: Beachten Sie, dass Sie bei dieser Aufgabe auch die Klasse `Klausur` erweitern müssen.

Eine Implementierung der Klassen `KlausurH`, `KlausurverwaltungsHFrame` und `KlausurverwaltungsHFrameMain` finden Sie auch im ZIP-Archiv.

```
1  /**
2   * Repraesentation einer Klausur mit einem Titel und einer Liste von angemeldeten
3   * Studenten.
4   *
5   * @author Annabelle Klarl
6   *
7   */
8  public class KlausurH {
9      private String titel;
10     private Student[] teilnehmer;
11     private int anzahlAngemeldete;
12
13     /**
14      * Konstruktor
15      *
16      * @param titel
17      *             Titel der Klausur
18      * @param maxTeilnehmer
19      *             maximale Anzahl an Teilnehmern
20      */
21     public KlausurH(String titel, int maxTeilnehmer) {
22         this.titel = titel;
23         this.teilnehmer = new Student[maxTeilnehmer];
24         this.anzahlAngemeldete = 0;
```

```
25      }
26
27      /**
28       * Diese Methode liefert den Titel der Klausur
29       *
30       * @return
31       */
32      public String getTitel() {
33          return this.titel;
34      }
35
36      /**
37       * Diese Methode meldet einen Studenten mit der gegebenen Matrikelnummer und
38       * dem gegebenen Namen an. Dazu wird zunaechst ein neues Objekt der Klasse
39       * {@link Student} erzeugt, dieser an der Klausur angemeldet und true
40       * zurueckgegeben. Ist die Klausur schon voll (d.h. wird die Maximalanzahl an
41       * Teilnehmern ueberschritten), wird der Student nicht angemeldet und false
42       * zurueckgegeben.
43       *
44       * @param matrNr
45       * @param name
46       * @return false falls die Maximalanzahl an Teilnehmern ueberschritten wurde,
47       *         true sonst
48       */
49      public boolean anmelden(int matrNr, String name) {
50          if (this.anzahlAngemeldete < this.teilnehmer.length) {
51              this.teilnehmer[this.anzahlAngemeldete] = new Student(matrNr, name);
52              this.anzahlAngemeldete++;
53              return true;
54          }
55          return false;
56      }
57
58      /**
59       * Diese Methode sucht in der Liste an Teilnehmern der Klausur einen
60       * Studenten mit der gegebenen Matrikelnummer. Wird ein Student gefunden,
61       * wird dieser zurueckgegeben. Falls kein Student mit dieser Matrikelnummer
62       * angemeldet ist, wird null zurueckgegeben.
63       *
64       * @param matrNr
65       * @return das Objekt der Klasse Student mit der gegebenen Matrikelnummer;
66       *         null falls kein Student mit dieser Matrikelnummer gefunden wird.
67       */
68      public Student sucheStudent(int matrNr) {
69          for (int i = 0; i < this.anzahlAngemeldete; i++) {
70              Student aktuellerStudent = this.teilnehmer[i];
71              if (aktuellerStudent.getMatrNr() == matrNr) {
72                  return aktuellerStudent;
73              }
74          }
75          return null;
76      }
77
78      /**
79       * Diese Methode traegt fuer einen Studenten mit der gegebenen Matrikelnummer
80       * die gegebene Note ein. Falls ein Student mit dieser Matrikelnummer an der
81       * Klausur angemeldet ist, wird die Note eingetragen und true zurueckgegeben.
82       * Falls kein Student angemeldet ist, wird false zurueckgegeben.
83       *
84       * @param matrNr
85       * @param note
86       * @return true falls ein Student mit der gegebenen Matrikelnummer an der
87       *         Klausur angemeldet ist (und die Note eingetragen wurde); false
88       *         sonst
89       */
```

```
90      public boolean noteEintragen(int matrNr, double note) {
91          Student student = this.sucheStudent(matrNr);
92          if (student != null) {
93              student.setNote(note);
94              return true;
95          }
96          return false;
97      }
98
99      /**
100      * Diese Methode berechnet die Durchschnittsnote aller benoteten
101      * Teilnehmern. Falls es keine benoteten Teilnehmer gab, wird -1
102      * zurueckgegeben.
103      *
104      * @return Die Durchschnittsnote aller benoteten Teilnehmer; falls es keine
105      *         benoteten Teilnehmer gibt, -1
106      */
107     public double durchschnittsnote() {
108         double notenAddiert = 0.0;
109         double anzahlMitschreiber = 0;
110
111         for (int i = 0; i < this.anzahlAngemeldete; i++) {
112             Student aktuellerStudent = this.teilnehmer[i];
113             double note = aktuellerStudent.getNote();
114             if (note > -1) {
115                 notenAddiert = notenAddiert + note;
116                 anzahlMitschreiber++;
117             }
118         }
119
120         if (anzahlMitschreiber == 0) {
121             return -1;
122         }
123         else {
124             return notenAddiert / anzahlMitschreiber;
125         }
126     }
127
128     /**
129      * Diese Methode gibt zurueck, wie viele Studenten die gegebene Note in der
130      * Klausur erreicht haben.
131      *
132      * @param note
133      * @return Die Anzahl an Studenten, die diese Note erreicht haben.
134      */
135     public int anzahlAnStudentenMitNote(double note) {
136         int anzahl = 0;
137         for (int i = 0; i < this.anzahlAngemeldete; i++) {
138             Student aktuellerStudent = this.teilnehmer[i];
139             if (Double.compare(aktuellerStudent.getNote(), note) == 0) {
140                 anzahl++;
141             }
142         }
143         return anzahl;
144     }
145 }
```

```
1 import java.awt.Container;
2 import java.awt.GridLayout;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8 import javax.swing.JOptionPane;
```

```
 9 import javax.swing.JPanel;
10 import javax.swing.JTextArea;
11
12 /**
13  * Eine grafische Benutzeroberflaeche zur Verwaltung von Klausuren.
14  *
15  * @author Annabelle Klarl
16  *
17  */
18 public class KlausurverwaltungsHFrame extends JFrame implements
19         ActionListener {
20
21     private JButton klausurAnlegenButton;
22     private JButton studentAnmeldenButton;
23     private JButton studentSuchenButton;
24     private JButton noteEintragenButton;
25
26     /* Hausaufgabe */
27     private JButton durchschnittsnoteButton;
28     private JButton studentenProNoteButton;
29
30     private JTextArea ausgabeBereich;
31
32     private KlausurH aktuelleKlausur;
33
34     /**
35      * In diesem Programmstueck wird das Fenster erzeugt.
36      */
37     public KlausurverwaltungsHFrame() {
38         /* In der Kopfleiste des Fenster steht "KlausurverwaltungFrame" */
39         this.setTitle("KlausurverwaltungFrame");
40
41         /* Das Fenster ist 700 Pixel breit und 350 Pixel hoch. */
42         this.setSize(700, 350);
43
44         /* Hier werden alle Buttons erzeugt. */
45         this.klausurAnlegenButton = new JButton("Lege Klausur an");
46         this.studentAnmeldenButton = new JButton("Student anmelden");
47         this.studentSuchenButton = new JButton("Student suchen");
48         this.noteEintragenButton = new JButton("Note eintragen");
49
50         /* Hausaufgabe */
51         this.durchschnittsnoteButton = new JButton(
52                 "Durchschnittsnote berechnen");
53         this.studentenProNoteButton = new JButton(
54                 "Anzahl an Studenten mit Note x anzeigen");
55
56         /* Hier wird der Ausgabe-Bereich erzeugt. */
57         this.ausgabeBereich = new JTextArea(10, 100);
58
59         /* Hier werden alle Buttons zusammengruppiert. */
60         JPanel studentenPanel = new JPanel();
61         studentenPanel.setLayout(new GridLayout(5, 1));
62         studentenPanel.add(this.studentAnmeldenButton);
63         studentenPanel.add(this.studentSuchenButton);
64         studentenPanel.add(this.noteEintragenButton);
65
66         /* Hausaufgabe */
67         studentenPanel.add(this.durchschnittsnoteButton);
68         studentenPanel.add(this.studentenProNoteButton);
69
70         JPanel buttonPanel = new JPanel();
71         buttonPanel.setLayout(new GridLayout(1, 2));
72         buttonPanel.add(this.klausurAnlegenButton);
73         buttonPanel.add(studentenPanel);
```

```
74
75          /*
76           * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
77           * Interaktionselemente (wie eine TextArea oder ein Button) platziert
78           * werden koennen.
79           */
80          Container contentPane = this.getContentPane();
81          contentPane.setLayout(new GridLayout(2, 1));
82          /* Hier wird die Gruppe von Buttons platziert. */
83          contentPane.add(buttonPanel);
84          /* Hier wird der Ausgabebereich platziert. */
85          contentPane.add(this.ausgabeBereich);
86
87          /*
88           * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
89           * Button registriert.
90           */
91          this.klausurAnlegenButton.addActionListener(this);
92          this.studentAnmeldenButton.addActionListener(this);
93          this.studentSuchenButton.addActionListener(this);
94          this.noteEintragenButton.addActionListener(this);
95
96          /* Hausaufgabe */
97          this.durchschnittsnoteButton.addActionListener(this);
98          this.studentenProNoteButton.addActionListener(this);
99
100         /*
101          * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
102          * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
103          */
104         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
105     }
106
107     /**
108      * Dieses Programmstueck wird immer dann ausgefuehrt, wenn ein Benutzer auf
109      * einen Button drueckt.
110      */
111     @Override
112     public void actionPerformed(ActionEvent e) {
113         Object source = e.getSource();
114         if (source == this.klausurAnlegenButton) {
115             this.klausurAnlegen();
116         } else if (source == this.studentAnmeldenButton) {
117             this.studentAnmelden();
118         } else if (source == this.studentSuchenButton) {
119             this.studentSuchen();
120         } else if (source == this.noteEintragenButton) {
121             this.noteEintragen();
122         } else if (source == this.durchschnittsnoteButton) {
123             this.durchschnittsnoteBerechnen();
124         } else if (source == this.studentenProNoteButton) {
125             this.studentenProNoteAusgeben();
126         }
127     }
128
129     /**
130      * Diese Methode erzeugt eine neue Klausur, falls es momentan keine gibt.
131      */
132     private void klausurAnlegen() {
133         if (this.aktuelleKlausur != null) {
134             this.ausgabeBereich
135                     .setText("Es wurde bereits eine Klausur mit dem Titel "
136                             + this.aktuelleKlausur.getTitel()
137                             + " angelegt.");
138         } else {
```

```
139             String einlesenTitel = JOptionPane
140                     .showInputDialog("Titel der Klausur: ");
141
142             String einlesenMaxTeilnehmer = JOptionPane
143                     .showInputDialog("Maximale Anzahl an Teilnehmern: ");
144             int maxTeilnehmer = Integer
145                     .parseInt(einlesenMaxTeilnehmer);
146
147             // !!!
148             this.aktuelleKlausur = new KlausurH(einlesenTitel,
149                     maxTeilnehmer);
150
151             this.ausgabeBereich
152                     .setText("Folgende Klausur wurde angelegt: "
153                             + this.aktuelleKlausur
154                             + " mit dem Titel "
155                             + this.aktuelleKlausur.getTitel());
156         }
157     }
158
159     /**
160      * Diese Methode meldet einen Studenten zur Klausur an, falls es momentan
161      * eine Klausur gibt.
162      */
163     private void studentAnmelden() {
164         if (this.aktuelleKlausur == null) {
165             this.ausgabeBereich
166                     .setText("Es wurde noch keine Klausur angelegt.");
167         } else {
168             String einlesenMatrNr = JOptionPane
169                     .showInputDialog("Matrikelnummer: ");
170             int matrNr = Integer.parseInt(einlesenMatrNr);
171
172             String name = JOptionPane.showInputDialog("Name: ");
173
174             // !!! Methodenaufruf
175             boolean angemeldet = this.aktuelleKlausur.anmelden(
176                     matrNr, name);
177             if (angemeldet) {
178                 this.ausgabeBereich.setText("Der Student " + name
179                         + " mit der Matrikelnummer " + matrNr
180                         + " wurde angemeldet.");
181             } else {
182                 this.ausgabeBereich
183                         .setText("Der Student konnte nicht angemeldet werden, "
184                         + "da die Maximalanzahl an Teilnehmern "
185                         + "ueberschritten wurde.");
186             }
187         }
188     }
189
190     /**
191      * Diese Methode gibt Informationen ueber einen an der Klausur angemeldeten
192      * Studenten aus, falls es momentan eine Klausur gibt und ein Student mit
193      * der gegebenen Matrikelnummer angemeldet ist.
194      */
195     private void studentSuchen() {
196         if (this.aktuelleKlausur == null) {
197             this.ausgabeBereich
198                     .setText("Es wurde noch keine Klausur angelegt.");
199         } else {
200             String einlesenMatrNr = JOptionPane
201                     .showInputDialog("Matrikelnummer: ");
202             int matrNr = Integer.parseInt(einlesenMatrNr);
203
```

```
204             // !!! Methodenaufruf
205             Student student = this.aktuelleKlausur
206                     .sucheStudent(matrNr);
207             if (student == null) {
208                 this.ausgabeBereich
209                         .setText("Der Student konnte nicht gefunden werden.");
210             } else {
211                 this.ausgabeBereich
212                         .setText("Folgender Student wurde gefunden: "
213                                 + student.getName()
214                                 + " mit der Matrikelnummer "
215                                 + student.getMatrNr()
216                                 + " und der Note "
217                                 + student.getNote());
218             }
219         }
220     }
221
222     /**
223      * Diese Methode traegt eine Note fuer einen Studenten ein, falls es momentan
224      * eine Klausur gibt und ein Student mit der gegebene Matrikelnummer
225      * angemeldet ist.
226      */
227     private void noteEintragen() {
228         if (this.aktuelleKlausur == null) {
229             this.ausgabeBereich
230                     .setText("Es wurde noch keine Klausur angelegt.");
231         } else {
232             String einlesenMatrNr = JOptionPane
233                     .showInputDialog("Matrikelnummer: ");
234             int matrNr = Integer.parseInt(einlesenMatrNr);
235
236             String einlesenNote = JOptionPane
237                     .showInputDialog("Note: ");
238             double note = Double.parseDouble(einlesenNote);
239
240             // !!! Methodenaufruf
241             boolean benotet = this.aktuelleKlausur.noteEintragen(
242                     matrNr, note);
243             if (benotet) {
244                 this.ausgabeBereich
245                         .setText("Der Student mit der Matrikelnummer "
246                                 + matrNr
247                                 + " wurde mit der Note "
248                                 + note + " benotet.");
249             } else {
250                 this.ausgabeBereich
251                         .setText("Der Student konnte nicht benotet werden, "
252                                 + "da kein Student mit der Matrikelnummer "
253                                 + matrNr + " gefunden werden konnte.");
254             }
255         }
256     }
257
258     /**
259      * Diese Methode berechnet die Durchschnittsnote der Klausur, falls es
260      * momentan eine Klausur gibt.
261      */
262     private void durchschnittsnoteBerechnen() {
263         if (this.aktuelleKlausur == null) {
264             this.ausgabeBereich
265                     .setText("Es wurde noch keine Klausur angelegt.");
266         } else {
267             // !!! Methodenaufruf
268             double durchschnittsnote = this.aktuelleKlausur
```

```
269                     .durchschnittsnote();
270             if (durchschnittsnote > -1) {
271                 this.ausgabeBereich
272                         .setText("Die Durchschnittsnote der Klausur war "
273                                 + durchschnittsnote);
274             } else {
275                 this.ausgabeBereich
276                         .setText("Die Klausur hat keine Teilnehmer "
277                                  + "oder wurde noch nicht benotet.");
278             }
279         }
280     }
281
282     /**
283      * Diese Methode gibt die Anzahl aller Studenten mit einer bestimmten Note
284      * in der Klausur aus, falls es momentan eine Klausur gibt.
285      */
286     private void studentenProNoteAusgeben() {
287         if (this.aktuelleKlausur == null) {
288             this.ausgabeBereich
289                     .setText("Es wurde noch keine Klausur angelegt.");
290         } else {
291             String einlesenNote = JOptionPane
292                     .showInputDialog("Note: ");
293             double note = Double.parseDouble(einlesenNote);
294
295             // !!! Methodenaufruf
296             int anzahl = this.aktuelleKlausur
297                     .anzahlAnStudentenMitNote(note);
298             this.ausgabeBereich.setText(anzahl
299                     + " Studenten hatten die Note " + note);
300         }
301     }
302 }
```

```
1  /**
2   * Diese Klasse startet den KlausurverwaltungFrame
3   *
4   * @author Annabelle Klarl
5   *
6   */
7  public class KlausurverwaltungsHFrameMain {
8
9      /**
10      * Dieses Programmstueck startet das Programm.
11      *
12      * @param args
13      */
14     public static void main(String[] args) {
15         KlausurverwaltungsHFrame frame = new KlausurverwaltungsHFrame();
16         frame.setVisible(true);
17     }
18
19 }
```

Besprechung der Präsenzaufgaben in den Übungen am 12.01.2018 und 15.01.2018. Abgabe der Hausaufgaben bis Mittwoch, 24.01.2018, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung). Erstellen Sie zu jeder Aufgabe Klassen, die die Namen tragen, die in der Aufgabe gefordert sind. Geben Sie nur die entsprechenden `.java`*-Dateien ab. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien.*