Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. D. Sabel, Dr. P. Wendler

WS 2018/19
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  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8  import javax.swing.JTextArea;
9
10 /**
11  * Diese Klasse repraesentiert das Hauptfenster der Praedikat-Praesenzaufgabe.
12  *
13  * @author Annabelle Klarl
14  */
15 public class PraedikatFrame extends JFrame implements ActionListener {
16   private JButton duplikatfreiButton;
17
18   private JTextArea ausgabeBereich;
19
20   /** In diesem Programmstueck wird das Fenster erzeugt */
21   public PraedikatFrame() {
22     /* In der Kopfleiste des Fenster steht "PraedikatFrame" */
23     this.setTitle("PraedikatFrame");
24
25     /* Das Fenster ist 500 Pixel breit und 150 Pixel hoch. */
26     this.setSize(500, 150);
27
28     /* Hier werden alle Buttons erzeugt. */
29     this.duplikatfreiButton = new JButton("Ist Array duplikatfrei?");
30
31     /* Hier wird der Ausgabe-Bereich erzeugt. */
32     this.ausgabeBereich = new JTextArea(10, 100);
33
34     /*
35      * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
36      * Interaktionselemente (wie eine TextArea oder ein Button) platziert
37      * werden koennen.
38      */
39     Container contentPane = this.getContentPane();
40     contentPane.setLayout(new GridLayout(2, 1));
41     /* Hier wird der Button platziert. */
```

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

```
1  /**
2   * Diese Klasse startet den PraedikatFrame
3   *
4   * @author Annabelle Klarl
```

```
 */
public class PraedikatFrameMain {

  /**
   * Dieses Programmstueck startet das Programm.
   *
   * @param args
   */
  public static void main(String[] args) {
    PraedikatFrame arrayFrame = new PraedikatFrame();
    arrayFrame.setVisible(true);
  }
}
```

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

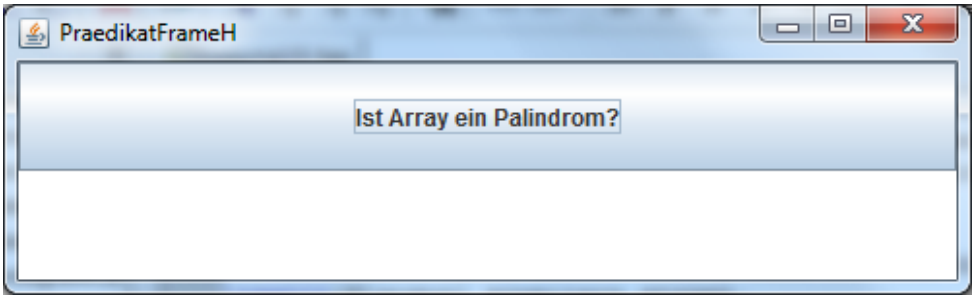
Lösung:
Im schlechtesten Fall werden die zwei geschachelte Schleifen komplett durchlaufen, daher ist die Ordnung der Zeitkomplexität $O(n^2)$, wobei n die Länge des Eingabearrays ist.

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  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8  import javax.swing.JTextArea;
9
10 /**
11  * Diese Klasse repraesentiert das Hauptfenster der Praedikat-Hausaufgabe.
12  *
13  * @author Annabelle Klarl
14  */
15 public class PraedikatFrameH extends JFrame implements ActionListener {
16   private JButton palindromButton;
17
18   private JTextArea ausgabeBereich;
19
20   /** In diesem Programmstueck wird das Fenster erzeugt */
21   public PraedikatFrameH() {
22     /* In der Kopfleiste des Fenster steht "PraedikatFrameH" */
23     this.setTitle("PraedikatFrameH");
```

```
24
25      /* Das Fenster ist 500 Pixel breit und 150 Pixel hoch. */
26      this.setSize(500, 150);
27
28      /* Hier werden alle Buttons erzeugt. */
29      this.palindromButton = new JButton("Ist Array ein Palindrom?");
30
31      /* Hier wird der Ausgabe-Bereich erzeugt. */
32      this.ausgabeBereich = new JTextArea(10, 100);
33
34      /*
35       * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
36       * Interaktionselemente (wie eine TextArea oder ein Button) platziert
37       * werden koennen.
38       */
39      Container contentPane = this.getContentPane();
40      contentPane.setLayout(new GridLayout(2, 1));
41      /* Hier wird die Gruppe von Buttons platziert. */
42      contentPane.add(this.palindromButton);
43      /* Hier wird der Ausgabebereich platziert. */
44      contentPane.add(this.ausgabeBereich);
45
46      /*
47       * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
48       * der Buttons registriert.
49       */
50      this.palindromButton.addActionListener(this);
51
52      /*
53       * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
54       * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
55       */
56      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
57    }
58
59    @Override
60    public void actionPerformed(ActionEvent e) {
61      // je nach Button entsprechende Methode ausfuehren
62      Object source = e.getSource();
63      if (source == this.palindromButton) {
64        this.palindrom();
65      }
66    }
67
68    /** implementiert die Funktionalitaet des "Palindrom"-Buttons */
69    private void palindrom() {
70      String einlesenArray = JOptionPane.showInputDialog("char-Array: ");
71      char[] array = KonverterErweitert.konvertiereZuCharArray(einlesenArray);
72
73      boolean istPalindrom = istPalindrom(array);
74
75      String eingabe = "Eingabe: " + einlesenArray + "\n\n";
76      String ergebnis;
77      if (istPalindrom) {
78        ergebnis = eingabe + "Das Array ist ein Palindrom.";
79      } else {
80        ergebnis = eingabe + "Das Array KEIN Palindrom.";
81      }
82      this.ausgabeBereich.setText(ergebnis);
83    }
84
85    /**
86     * testet, ob das Array palindromisch angeordnet ist
87     *
88     * @param array
```

```
89     * @return
90     */
91    private static boolean istPalindrom(char[] array) {
92      for (int i = 0; i < array.length / 2; i++) {
93        if (array[i] != array[array.length - 1 - i]) {
94          return false;
95        }
96      }
97      return true;
98    }
99  }
```

```
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 }
```

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.

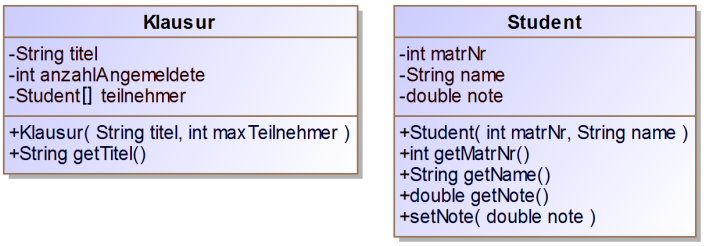
a) Schreiben Sie eine Klasse `Student`, die einen Teilnehmer an einer Klausur repräsentiert. Für einen Studenten sollen seine Matrikelnummer, sein Name und eine Note gespeichert werden können.

- 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.

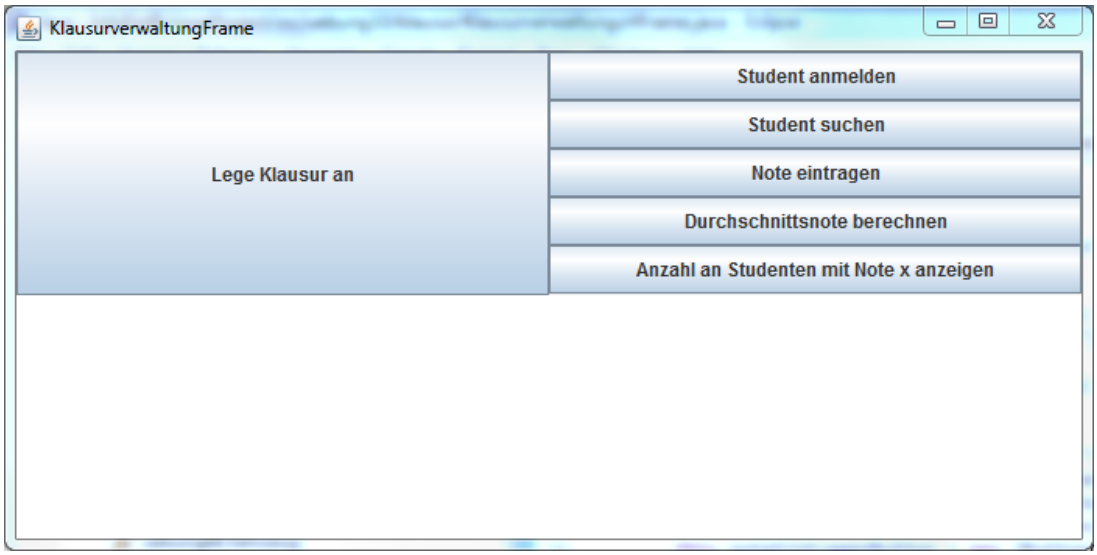
```java
/**
 * Repraesentation eines Studenten mit Matrikelnummer, Name und Note.
 *
 * @author Annabelle Klarl
 */
public class Student {
  private int matrNr;
  private String name;
  private double note;

  public Student(int matrNr, String name) {
    this.matrNr = matrNr;
    this.name = name;
    this.note = -1.0;
  }

  public int getMatrNr() {
    return this.matrNr;
  }

  public String getName() {
    return this.name;
  }

  public double getNote() {
    return this.note;
  }

  public void setNote(double note) {
    this.note = note;
  }
}
```

```java
/**
 * Repraesentation einer Klausur mit einem Titel
 * und einer Liste von angemeldeten Studenten.
 *
 * @author Annabelle Klarl
```

```
 */
public class Klausur {
  private String titel;
  private Student[] teilnehmer;
  private int anzahlAngemeldete;

  /**
   * Konstruktor
   *
   * @param titel Titel der Klausur
   * @param maxTeilnehmer maximale Anzahl an Teilnehmern
   */
  public Klausur(String titel, int maxTeilnehmer) {
    this.titel = titel;
    this.teilnehmer = new Student[maxTeilnehmer];
    this.anzahlAngemeldete = 0;
  }

  /**
   * Diese Methode liefert den Titel der Klausur
   *
   * @return
   */
  public String getTitel() {
    return this.titel;
  }

  /**
   * Diese Methode meldet einen Studenten mit der gegebenen Matrikelnummer
   * und dem gegebenen Namen an. Dazu wird zunaechst ein neues Objekt der
   * Klasse {@link Student} erzeugt, dieser an der Klausur angemeldet und
   * true zurueckgegeben. Ist die Klausur schon voll (d.h. wird die
   * Maximalanzahl an Teilnehmern ueberschritten), wird der Student nicht
   * angemeldet und false zurueckgegeben.
   *
   * @param matrNr
   * @param name
   * @return false falls die Maximalanzahl an Teilnehmern ueberschritten wurde,
   *     true sonst
   */
  public boolean anmelden(int matrNr, String name) {
    if (this.anzahlAngemeldete < this.teilnehmer.length) {
      this.teilnehmer[this.anzahlAngemeldete] = new Student(matrNr, name);
      this.anzahlAngemeldete++;
      return true;
    }
    return false;
  }

  /**
   * Diese Methode sucht in der Liste an Teilnehmern der Klausur einen
   * Studenten mit der gegebenen Matrikelnummer. Wird ein Student gefunden,
   * wird dieser zurueckgegeben. Falls kein Student mit dieser Matrikelnummer
   * angemeldet ist, wird null zurueckgegeben.
   *
   * @param matrNr
   * @return das Objekt der Klasse Student mit der gegebenen Matrikelnummer;
   *     null falls kein Student mit dieser Matrikelnummer gefunden wird.
   */
  public Student sucheStudent(int matrNr) {
    for (int i = 0; i < this.anzahlAngemeldete; i++) {
      Student aktuellerStudent = this.teilnehmer[i];
      if (aktuellerStudent.getMatrNr() == matrNr) {
        return aktuellerStudent;
      }
```

```
71      }
72      return null;
73    }
74
75    /**
76     * Diese Methode traegt fuer einen Studenten mit der gegebenen
77     * Matrikelnummer die gegebene Note ein. Falls ein Student mit dieser
78     * Matrikelnummer an der Klausur angemeldet ist, wird die Note
79     * eingetragen und true zurueckgegeben. Falls kein Student angemeldet
80     * ist, wird false zurueckgegeben.
81     *
82     * @param matrNr
83     * @param note
84     * @return true falls ein Student mit der gegebenen Matrikelnummer an
85     *      der Klausur angemeldet ist
86     *      (und die Note eingetragen wurde); false sonst
87     */
88    public boolean noteEintragen(int matrNr, double note) {
89      Student student = this.sucheStudent(matrNr);
90      if (student != null) {
91        student.setNote(note);
92        return true;
93      }
94      return false;
95    }
96  }
```

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 informiert 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.

```
1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8  import javax.swing.JPanel;
9  import javax.swing.JTextArea;
10
11 /**
12  * Eine grafische Benutzeroberflaeche zur Verwaltung von Klausuren.
13  *
14  * @author Annabelle Klarl
15  */
16 public class KlausurverwaltungsFrame extends JFrame implements ActionListener {
17
18   private JButton klausurAnlegenButton;
19   private JButton studentAnmeldenButton;
20   private JButton studentSuchenButton;
21   private JButton noteEintragenButton;
22
23   /* Hausaufgabe */
24   private JButton durchschnittsnoteButton;
25   private JButton studentenProNoteButton;
26
27   private JTextArea ausgabeBereich;
28
29   private Klausur aktuelleKlausur;
30
31   /** In diesem Programmstueck wird das Fenster erzeugt. */
32   public KlausurverwaltungsFrame() {
33     /* In der Kopfleiste des Fenster steht "KlausurverwaltungFrame" */
34     this.setTitle("KlausurverwaltungFrame");
35
36     /* Das Fenster ist 700 Pixel breit und 350 Pixel hoch. */
37     this.setSize(700, 350);
```

```
38
39      /* Hier werden alle Buttons erzeugt. */
40      this.klausurAnlegenButton = new JButton("Lege Klausur an");
41      this.studentAnmeldenButton = new JButton("Student anmelden");
42      this.studentSuchenButton = new JButton("Student suchen");
43      this.noteEintragenButton = new JButton("Note eintragen");
44
45      /* Hausaufgabe */
46      this.durchschnittsnoteButton = new JButton("Durchschnittsnote berechnen");
47      this.studentenProNoteButton =
48          new JButton("Anzahl an Studenten mit Note x anzeigen");
49
50      /* Hier wird der Ausgabe-Bereich erzeugt. */
51      this.ausgabeBereich = new JTextArea(10, 100);
52
53      /* Hier werden alle Buttons zusammengruppiert. */
54      JPanel studentenPanel = new JPanel();
55      studentenPanel.setLayout(new GridLayout(5, 1));
56      studentenPanel.add(this.studentAnmeldenButton);
57      studentenPanel.add(this.studentSuchenButton);
58      studentenPanel.add(this.noteEintragenButton);
59
60      /* Hausaufgabe */
61      studentenPanel.add(this.durchschnittsnoteButton);
62      studentenPanel.add(this.studentenProNoteButton);
63
64      JPanel buttonPanel = new JPanel();
65      buttonPanel.setLayout(new GridLayout(1, 2));
66      buttonPanel.add(this.klausurAnlegenButton);
67      buttonPanel.add(studentenPanel);
68
69      /*
70       * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
71       * Interaktionselemente (wie eine TextArea oder ein Button) platziert
72       * werden koennen.
73       */
74      Container contentPane = this.getContentPane();
75      contentPane.setLayout(new GridLayout(2, 1));
76      /* Hier wird die Gruppe von Buttons platziert. */
77      contentPane.add(buttonPanel);
78      /* Hier wird der Ausgabebereich platziert. */
79      contentPane.add(this.ausgabeBereich);
80
81      /*
82       * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
83       * Button registriert.
84       */
85      this.klausurAnlegenButton.addActionListener(this);
86      this.studentAnmeldenButton.addActionListener(this);
87      this.studentSuchenButton.addActionListener(this);
88      this.noteEintragenButton.addActionListener(this);
89
90      /*
91       * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
92       * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
93       */
94      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
95  }
96
97  /**
98   * Dieses Programmstueck wird immer dann ausgefuehrt,
99   * wenn ein Benutzer auf einen Button drueckt.
100  */
101 @Override
102 public void actionPerformed(ActionEvent e) {
```

```
103     Object source = e.getSource();
104     if (source == this.klausurAnlegenButton) {
105       this.klausurAnlegen();
106     } else if (source == this.studentAnmeldenButton) {
107       this.studentAnmelden();
108     } else if (source == this.studentSuchenButton) {
109       this.studentSuchen();
110     } else if (source == this.noteEintragenButton) {
111       this.noteEintragen();
112     }
113   }
114
115   /** Diese Methode erzeugt eine neue Klausur, falls es momentan keine gibt. */
116   private void klausurAnlegen() {
117     if (this.aktuelleKlausur != null) {
118       this.ausgabeBereich.setText(
119           "Es wurde bereits eine Klausur mit dem Titel "
120               + this.aktuelleKlausur.getTitel()
121               + " angelegt.");
122     } else {
123       String einlesenTitel = JOptionPane.showInputDialog("Titel der Klausur: ");
124
125       String einlesenMaxTeilnehmer =
126           JOptionPane.showInputDialog("Maximale Anzahl an Teilnehmern: ");
127       int maxTeilnehmer = Integer.parseInt(einlesenMaxTeilnehmer);
128
129       // !!!!
130       this.aktuelleKlausur = new Klausur(einlesenTitel, maxTeilnehmer);
131
132       this.ausgabeBereich.setText(
133           "Folgende Klausur wurde angelegt: "
134               + this.aktuelleKlausur
135               + " mit dem Titel "
136               + this.aktuelleKlausur.getTitel());
137     }
138   }
139
140   /**
141    * Diese Methode meldet einen Studenten zur Klausur an,
142    * falls es momentan eine Klausur gibt.
143    */
144   private void studentAnmelden() {
145     if (this.aktuelleKlausur == null) {
146       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
147     } else {
148       String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
149       int matrNr = Integer.parseInt(einlesenMatrNr);
150
151       String name = JOptionPane.showInputDialog("Name: ");
152
153       // !!! Methodenaufruf
154       boolean angemeldet = this.aktuelleKlausur.anmelden(matrNr, name);
155       if (angemeldet) {
156         this.ausgabeBereich.setText(
157             "Der Student "
158                 + name
159                 + " mit der Matrikelnummer "
160                 + matrNr
161                 + " wurde angemeldet.");
162       } else {
163         this.ausgabeBereich.setText(
164             "Der Student konnte nicht angemeldet werden, "
165                 + "da die Maximalanzahl an Teilnehmern "
166                 + "ueberschritten wurde.");
167       }
```

```
168     }
169   }
170
171   /**
172    * Diese Methode gibt Informationen ueber einen an der Klausur angemeldeten
173    * Studenten aus, falls es momentan eine Klausur gibt und ein Student mit
174    * der gegebenen Matrikelnummer angemeldet ist.
175    */
176   private void studentSuchen() {
177     if (this.aktuelleKlausur == null) {
178       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
179     } else {
180       String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
181       int matrNr = Integer.parseInt(einlesenMatrNr);
182
183       // !!! Methodenaufruf
184       Student student = this.aktuelleKlausur.sucheStudent(matrNr);
185       if (student == null) {
186         this.ausgabeBereich.setText(
187             "Der Student konnte nicht gefunden werden.");
188       } else {
189         this.ausgabeBereich.setText(
190             "Folgender Student wurde gefunden: "
191                 + student.getName()
192                 + " mit der Matrikelnummer "
193                 + student.getMatrNr()
194                 + " und der Note "
195                 + student.getNote());
196       }
197     }
198   }
199
200   /**
201    * Diese Methode traegt eine Note fuer einen Studenten ein, falls es
202    * momentan eine Klausur gibt und ein Student mit der gegebene
203    * Matrikelnummer angemeldet ist.
204    */
205   private void noteEintragen() {
206     if (this.aktuelleKlausur == null) {
207       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
208     } else {
209       String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
210       int matrNr = Integer.parseInt(einlesenMatrNr);
211
212       String einlesenNote = JOptionPane.showInputDialog("Note: ");
213       double note = Double.parseDouble(einlesenNote);
214
215       // !!! Methodenaufruf
216       boolean benotet = this.aktuelleKlausur.noteEintragen(matrNr, note);
217       if (benotet) {
218         this.ausgabeBereich.setText(
219             "Der Student mit der Matrikelnummer "
220                 + matrNr
221                 + " wurde mit der Note "
222                 + note
223                 + " benotet.");
224       } else {
225         this.ausgabeBereich.setText(
226             "Der Student konnte nicht benotet werden, "
227                 + "da kein Student mit der Matrikelnummer "
228                 + matrNr
229                 + " gefunden werden konnte.");
230       }
231     }
232   }
```

```
233 }
```

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

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.

```java
/**
 * Repraesentation einer Klausur mit einem Titel
 * und einer Liste von angemeldeten Studenten.
 *
 * @author Annabelle Klarl
 */
public class KlausurH {
  private String titel;
  private Student[] teilnehmer;
  private int anzahlAngemeldete;

  /**
   * Konstruktor
   *
   * @param titel Titel der Klausur
   * @param maxTeilnehmer maximale Anzahl an Teilnehmern
   */
  public KlausurH(String titel, int maxTeilnehmer) {
    this.titel = titel;
    this.teilnehmer = new Student[maxTeilnehmer];
    this.anzahlAngemeldete = 0;
  }

  /**
   * Diese Methode liefert den Titel der Klausur
   *
   * @return
   */
  public String getTitel() {
    return this.titel;
  }

  /**
   * Diese Methode meldet einen Studenten mit der gegebenen Matrikelnummer
   * und dem gegebenen Namen an. Dazu wird zunaechst ein neues Objekt der
   * Klasse {@link Student} erzeugt, dieser an der Klausur angemeldet und
   * true zurueckgegeben. Ist die Klausur schon voll (d.h. wird die
   * Maximalanzahl an Teilnehmern ueberschritten), wird der Student nicht
   * angemeldet und false zurueckgegeben.
   *
   * @param matrNr
   * @param name
   * @return false falls die Maximalanzahl an Teilnehmern ueberschritten wurde,
   *     true sonst
   */
  public boolean anmelden(int matrNr, String name) {
    if (this.anzahlAngemeldete < this.teilnehmer.length) {
      this.teilnehmer[this.anzahlAngemeldete] = new Student(matrNr, name);
      this.anzahlAngemeldete++;
      return true;
    }
    return false;
  }

  /**
   * Diese Methode sucht in der Liste an Teilnehmern der Klausur einen
   * Studenten mit der gegebenen Matrikelnummer. Wird ein Student gefunden,
   * wird dieser zurueckgegeben. Falls kein Student mit dieser Matrikelnummer
   * angemeldet ist, wird null zurueckgegeben.
   *
   * @param matrNr
   * @return das Objekt der Klasse Student mit der gegebenen Matrikelnummer;
   *     null falls kein Student mit dieser Matrikelnummer gefunden wird.
   */
```

```
65   public Student sucheStudent(int matrNr) {
66     for (int i = 0; i < this.anzahlAngemeldete; i++) {
67       Student aktuellerStudent = this.teilnehmer[i];
68       if (aktuellerStudent.getMatrNr() == matrNr) {
69         return aktuellerStudent;
70       }
71     }
72     return null;
73   }
74
75   /**
76    * Diese Methode traegt fuer einen Studenten mit der gegebenen
77    * Matrikelnummer die gegebene Note ein. Falls ein Student mit dieser
78    * Matrikelnummer an der Klausur angemeldet ist, wird die Note
79    * eingetragen und true zurueckgegeben. Falls kein Student angemeldet
80    * ist, wird false zurueckgegeben.
81    *
82    * @param matrNr
83    * @param note
84    * @return true falls ein Student mit der gegebenen Matrikelnummer an
85    *      der Klausur angemeldet ist
86    *      (und die Note eingetragen wurde); false sonst
87    */
88   public boolean noteEintragen(int matrNr, double note) {
89     Student student = this.sucheStudent(matrNr);
90     if (student != null) {
91       student.setNote(note);
92       return true;
93     }
94     return false;
95   }
96
97   /**
98    * Diese Methode berechnet die Durchschnittsnote aller benoteten
99    * Teilnehmern. Falls es keine benoteten Teilnehmer gab, wird -1
100   * zurueckgegeben.
101   *
102   * @return Die Durchschnittsnote aller benoteten Teilnehmer;
103   *      falls es keine benoteten Teilnehmer gibt, -1
104   */
105  public double durchschnittsnote() {
106    double notenAddiert = 0.0;
107    double anzahlMitschreiber = 0;
108
109    for (int i = 0; i < this.anzahlAngemeldete; i++) {
110      Student aktuellerStudent = this.teilnehmer[i];
111      double note = aktuellerStudent.getNote();
112      if (note > -1) {
113        notenAddiert = notenAddiert + note;
114        anzahlMitschreiber++;
115      }
116    }
117
118    if (anzahlMitschreiber == 0) {
119      return -1;
120    } else {
121      return notenAddiert / anzahlMitschreiber;
122    }
123  }
124
125  /**
126   * Diese Methode gibt zurueck, wie viele Studenten die gegebene Note in
127   * der Klausur erreicht haben.
128   *
129   * @param note
```

```
130     * @return Die Anzahl an Studenten, die diese Note erreicht haben.
131     */
132   public int anzahlAnStudentenMitNote(double note) {
133     int anzahl = 0;
134     for (int i = 0; i < this.anzahlAngemeldete; i++) {
135       Student aktuellerStudent = this.teilnehmer[i];
136       if (Double.compare(aktuellerStudent.getNote(), note) == 0) {
137         anzahl++;
138       }
139     }
140     return anzahl;
141   }
142 }
```

```
1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8  import javax.swing.JPanel;
9  import javax.swing.JTextArea;
10
11 /**
12  * Eine grafische Benutzeroberflaeche zur Verwaltung von Klausuren.
13  *
14  * @author Annabelle Klarl
15  */
16 public class KlausurverwaltungsHFrame extends JFrame implements ActionListener {
17
18   private JButton klausurAnlegenButton;
19   private JButton studentAnmeldenButton;
20   private JButton studentSuchenButton;
21   private JButton noteEintragenButton;
22
23   /* Hausaufgabe */
24   private JButton durchschnittsnoteButton;
25   private JButton studentenProNoteButton;
26
27   private JTextArea ausgabeBereich;
28
29   private KlausurH aktuelleKlausur;
30
31   /** In diesem Programmstueck wird das Fenster erzeugt. */
32   public KlausurverwaltungsHFrame() {
33     /* In der Kopfleiste des Fenster steht "KlausurverwaltungFrame" */
34     this.setTitle("KlausurverwaltungFrame");
35
36     /* Das Fenster ist 700 Pixel breit und 350 Pixel hoch. */
37     this.setSize(700, 350);
38
39     /* Hier werden alle Buttons erzeugt. */
40     this.klausurAnlegenButton = new JButton("Lege Klausur an");
41     this.studentAnmeldenButton = new JButton("Student anmelden");
42     this.studentSuchenButton = new JButton("Student suchen");
43     this.noteEintragenButton = new JButton("Note eintragen");
44
45     /* Hausaufgabe */
46     this.durchschnittsnoteButton = new JButton("Durchschnittsnote berechnen");
47     this.studentenProNoteButton =
48         new JButton("Anzahl an Studenten mit Note x anzeigen");
49
50     /* Hier wird der Ausgabe-Bereich erzeugt. */
51     this.ausgabeBereich = new JTextArea(10, 100);
```

```
52
53      /* Hier werden alle Buttons zusammengruppiert. */
54      JPanel studentenPanel = new JPanel();
55      studentenPanel.setLayout(new GridLayout(5, 1));
56      studentenPanel.add(this.studentAnmeldenButton);
57      studentenPanel.add(this.studentSuchenButton);
58      studentenPanel.add(this.noteEintragenButton);
59
60      /* Hausaufgabe */
61      studentenPanel.add(this.durchschnittsnoteButton);
62      studentenPanel.add(this.studentenProNoteButton);
63
64      JPanel buttonPanel = new JPanel();
65      buttonPanel.setLayout(new GridLayout(1, 2));
66      buttonPanel.add(this.klausurAnlegenButton);
67      buttonPanel.add(studentenPanel);
68
69      /*
70       * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
71       * Interaktionselemente (wie eine TextArea oder ein Button) platziert
72       * werden koennen.
73       */
74      Container contentPane = this.getContentPane();
75      contentPane.setLayout(new GridLayout(2, 1));
76      /* Hier wird die Gruppe von Buttons platziert. */
77      contentPane.add(buttonPanel);
78      /* Hier wird der Ausgabebereich platziert. */
79      contentPane.add(this.ausgabeBereich);
80
81      /*
82       * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
83       * Button registriert.
84       */
85      this.klausurAnlegenButton.addActionListener(this);
86      this.studentAnmeldenButton.addActionListener(this);
87      this.studentSuchenButton.addActionListener(this);
88      this.noteEintragenButton.addActionListener(this);
89
90      /* Hausaufgabe */
91      this.durchschnittsnoteButton.addActionListener(this);
92      this.studentenProNoteButton.addActionListener(this);
93
94      /*
95       * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
96       * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
97       */
98      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
99    }
100
101   /**
102    * Dieses Programmstueck wird immer dann ausgefuehrt,
103    * wenn ein Benutzer auf einen Button drueckt.
104    */
105   @Override
106   public void actionPerformed(ActionEvent e) {
107     Object source = e.getSource();
108     if (source == this.klausurAnlegenButton) {
109       this.klausurAnlegen();
110     } else if (source == this.studentAnmeldenButton) {
111       this.studentAnmelden();
112     } else if (source == this.studentSuchenButton) {
113       this.studentSuchen();
114     } else if (source == this.noteEintragenButton) {
115       this.noteEintragen();
116     } else if (source == this.durchschnittsnoteButton) {
```

```
117       this.durchschnittsnoteBerechnen();
118     } else if (source == this.studentenProNoteButton) {
119       this.studentenProNoteAusgeben();
120     }
121   }
122
123   /** Diese Methode erzeugt eine neue Klausur, falls es momentan keine gibt. */
124   private void klausurAnlegen() {
125     if (this.aktuelleKlausur != null) {
126       this.ausgabeBereich.setText(
127           "Es wurde bereits eine Klausur mit dem Titel "
128               + this.aktuelleKlausur.getTitel()
129               + " angelegt.");
130     } else {
131       String einlesenTitel = JOptionPane.showInputDialog("Titel der Klausur: ");
132
133       String einlesenMaxTeilnehmer =
134           JOptionPane.showInputDialog("Maximale Anzahl an Teilnehmern: ");
135       int maxTeilnehmer = Integer.parseInt(einlesenMaxTeilnehmer);
136
137       // !!!
138       this.aktuelleKlausur = new KlausurH(einlesenTitel, maxTeilnehmer);
139
140       this.ausgabeBereich.setText(
141           "Folgende Klausur wurde angelegt: "
142               + this.aktuelleKlausur
143               + " mit dem Titel "
144               + this.aktuelleKlausur.getTitel());
145     }
146   }
147
148   /**
149    * Diese Methode meldet einen Studenten zur Klausur an,
150    * falls es momentan eine Klausur gibt.
151    */
152   private void studentAnmelden() {
153     if (this.aktuelleKlausur == null) {
154       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
155     } else {
156       String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
157       int matrNr = Integer.parseInt(einlesenMatrNr);
158
159       String name = JOptionPane.showInputDialog("Name: ");
160
161       // !!! Methodenaufruf
162       boolean angemeldet = this.aktuelleKlausur.anmelden(matrNr, name);
163       if (angemeldet) {
164         this.ausgabeBereich.setText(
165             "Der Student "
166                 + name
167                 + " mit der Matrikelnummer "
168                 + matrNr
169                 + " wurde angemeldet.");
170       } else {
171         this.ausgabeBereich.setText(
172             "Der Student konnte nicht angemeldet werden, "
173                 + "da die Maximalanzahl an Teilnehmern "
174                 + "ueberschritten wurde.");
175       }
176     }
177   }
178
179   /**
180    * Diese Methode gibt Informationen ueber einen an der Klausur angemeldeten
181    * Studenten aus, falls es momentan eine Klausur gibt und ein Student mit
```

```
182     * der gegebenen Matrikelnummer angemeldet ist.
183     */
184    private void studentSuchen() {
185      if (this.aktuelleKlausur == null) {
186        this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
187      } else {
188        String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
189        int matrNr = Integer.parseInt(einlesenMatrNr);
190
191        // !!! Methodenaufruf
192        Student student = this.aktuelleKlausur.sucheStudent(matrNr);
193        if (student == null) {
194          this.ausgabeBereich.setText(
195              "Der Student konnte nicht gefunden werden.");
196        } else {
197          this.ausgabeBereich.setText(
198              "Folgender Student wurde gefunden: "
199                  + student.getName()
200                  + " mit der Matrikelnummer "
201                  + student.getMatrNr()
202                  + " und der Note "
203                  + student.getNote());
204        }
205      }
206    }
207
208    /**
209     * Diese Methode traegt eine Note fuer einen Studenten ein, falls es
210     * momentan eine Klausur gibt und ein Student mit der gegebene
211     * Matrikelnummer angemeldet ist.
212     */
213    private void noteEintragen() {
214      if (this.aktuelleKlausur == null) {
215        this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
216      } else {
217        String einlesenMatrNr = JOptionPane.showInputDialog("Matrikelnummer: ");
218        int matrNr = Integer.parseInt(einlesenMatrNr);
219
220        String einlesenNote = JOptionPane.showInputDialog("Note: ");
221        double note = Double.parseDouble(einlesenNote);
222
223        // !!! Methodenaufruf
224        boolean benotet = this.aktuelleKlausur.noteEintragen(matrNr, note);
225        if (benotet) {
226          this.ausgabeBereich.setText(
227              "Der Student mit der Matrikelnummer "
228                  + matrNr
229                  + " wurde mit der Note "
230                  + note
231                  + " benotet.");
232        } else {
233          this.ausgabeBereich.setText(
234              "Der Student konnte nicht benotet werden, "
235                  + "da kein Student mit der Matrikelnummer "
236                  + matrNr
237                  + " gefunden werden konnte.");
238        }
239      }
240    }
241
242    /**
243     * Diese Methode berechnet die Durchschnittsnote der Klausur,
244     * falls es momentan eine Klausur gibt.
245     */
246    private void durchschnittsnoteBerechnen() {
```

```
247     if (this.aktuelleKlausur == null) {
248       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
249     } else {
250       // !!! Methodenaufruf
251       double durchschnittsnote = this.aktuelleKlausur.durchschnittsnote();
252       if (durchschnittsnote > -1) {
253         this.ausgabeBereich.setText(
254             "Die Durchschnittsnote der Klausur war " + durchschnittsnote);
255       } else {
256         this.ausgabeBereich.setText(
257             "Die Klausur hat keine Teilnehmer oder wurde noch nicht benotet.");
258       }
259     }
260   }
261
262   /**
263    * Diese Methode gibt die Anzahl aller Studenten mit einer bestimmten Note
264    * in der Klausur aus, falls es momentan eine Klausur gibt.
265    */
266   private void studentenProNoteAusgeben() {
267     if (this.aktuelleKlausur == null) {
268       this.ausgabeBereich.setText("Es wurde noch keine Klausur angelegt.");
269     } else {
270       String einlesenNote = JOptionPane.showInputDialog("Note: ");
271       double note = Double.parseDouble(einlesenNote);
272
273       // !!! Methodenaufruf
274       int anzahl = this.aktuelleKlausur.anzahlAnStudentenMitNote(note);
275       this.ausgabeBereich.setText(
276           anzahl + " Studenten hatten die Note " + note);
277     }
278   }
279 }
```

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

Besprechung der Präsenzaufgaben in den Übungen am 21.12.2018 und 07.01.2019. Abgabe der Hausaufgaben bis Mittwoch, 16.01.2019, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung).

- *Erstellen Sie zu jeder Aufgabe Klassen mit den entsprechenden Namen, die in der Aufgabe gefordert sind.*
- *Geben Sie nur die entsprechenden* `.java`*-Dateien ab. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien.*

- *Geben Sie Java-Code nur in* `.java`*-Dateien ab. Java-Code in Bildern, PDF-Dokumenten und Text-Dateien wird nicht korrigiert.*