Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. R. Hennicker, Dr. A. Klarl

WS 2016/17
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`.

Auf der Webseite finden Sie die Implementierung der Klassen `PraedikatFrame` und `PraedikatFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `PraedikatFrame` und `PraedikatFrameMain`. Die Umsetzung des Algorithmus finden Sie ins Besondere in der Methode `istArrayDuplikatfrei`.

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

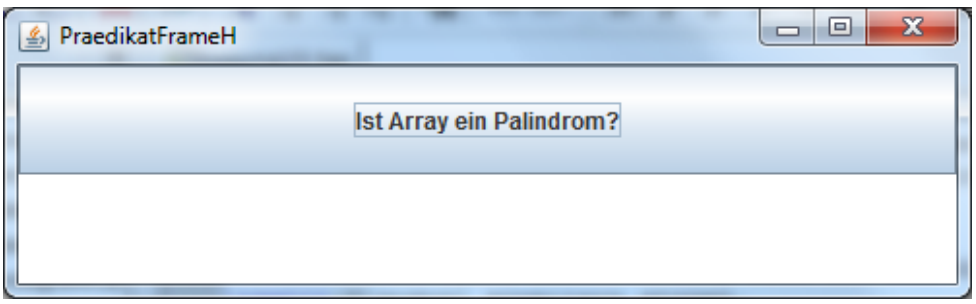
Lösung:
Im schlechtesten Fall werden die zwei geschachelte Schleifen komplett durchlaufen, daher ist die Ordnung der Zeitkomplexität $O(n^2)$, wobei n die Länge des Eingabearrays ist.

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`.

Auf der Webseite finden Sie die Implementierung der Klassen `PraedikatFrameH` und `PraedikatFrameHMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `PraedikatFrameH` und `PraedikatFrameHMain`. Die Umsetzung des Algorithmus finden Sie ins Besondere in der Methode `istPalindrom`.

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.

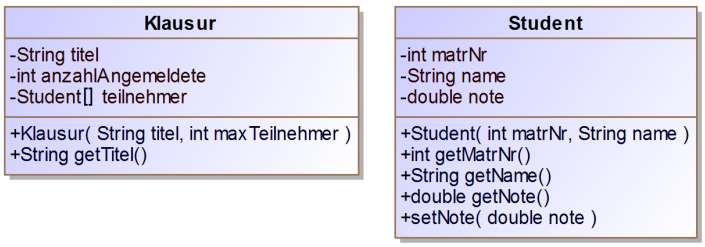
a) Schreiben Sie eine Klasse `Student`, die einen Teilnehmer an einer Klausur repräsentiert. Für einen Studenten sollen seine Matrikelnummer, sein Name und eine Note gespeichert werden können.

- 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.

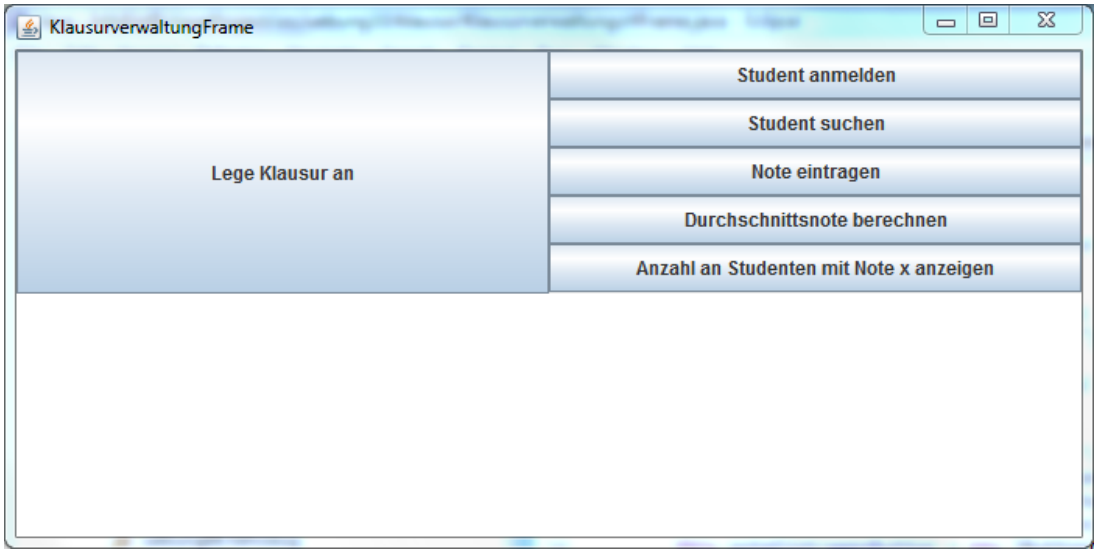
- Der Konstruktor der Klasse `Klausur` benötigt den Titel der Klausur und wie viele Studenten maximal an der Klausur teilnehmen können, um ein partielles Array geeignet initialisieren zu können.
- Schreiben Sie für das Attribut `titel` einen "Getter", der den Wert des Attribut zurück gibt.

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



Auf der Webseite finden Sie die Implementierung der Klassen `Student` und `Klausur`.

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!

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurverwaltungsFrame` und `KlausurverwaltungsFrameMain`.

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` 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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurH`, `KlausurverwaltungsFrameH` und `KlausurverwaltungsHFrameMain`.

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.

Auf der Webseite finden Sie die Implementierung der Klassen `KlausurH`, `KlausurverwaltungsFrameH` und `KlausurverwaltungsHFrameMain`.

*Besprechung der Präsenzaufgaben in den Übungen vom 21.12.2016 bis zum 11.01.2017. Abgabe der Hausaufgaben bis Mittwoch, 18.01.2017, 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.*