# Projektblatt P 6 (32 P)

**Abgabe:** Donnerstag 24. Oktober 2024, 12:00h

Entpacken Sie zunächst die Archiv-Datei `vorgaben-p6.zip`, in der sich eine Rahmendatei sowie eine Hilfsklasse für die zu lösenden Aufgaben befinden. Ergänzen Sie die Rahmendatei zunächst durch Ihren Namen. Legen Sie gemäß der Aufgabenstellung unten neue Dateien für die zu entwickelnden Klassen
Der Java-Sourcecode in allen Klassen sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Java-Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die von Ihnen bearbeiteten `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P6.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P6.zip  *.java
```

Nutzen Sie beim Bearbeiten der Aufgaben nur die Klassen und Methoden aus der Java API, welche explizit durch den Aufgabentext erlaubt sind.

## Aufgabe 1: OO Entwurf — 15 (= 7+4+4) P

In dieser Aufgabe sollen Sie eine Applikation entwickeln, die aus den drei Klassen `Fach`, `Schueler` und `ZeugnisSchreiber` besteht. Alle Klassen müssen Sie dabei in einer eigenen Datei erstellen.

(a) Legen Sie eine Klasse `Fach` an und statten Sie diese mit zwei privaten Instanzvariablen `name` (vom Typ `String`) und `note` (vom Typ `float`) aus. Legen Sie dazu zwei Methoden `getName` und `getNote` an, mit der Sie die Werte dieser Attribute von außerhalb der Klasse auslesen können sowie eine Methode `setNote`, mit der Sie den Notenwert von anderen Klassen aus ändern können. Schreiben Sie nun einen Konstruktor für die Klasse, der einen Namen für ein Fach als Parameter besitzt und die zugehörige Instanzvariable initialisiert. Implementieren Sie eine private Methode `toString`, die als Parameter einen ganzzahligen Wert `zeilenLaenge` besitzt, der die Länge einer Zeile (bei einer Ausgabe) repräsentiert. Die Methode soll einen `String` zurückgeben, der die Länge `zeilenLaenge` hat, linksbündig den Namen des Fachs enthält und rechtsbündig die Note. Die Note soll dabei durch einen der Werte "sehr gut", "gut","befriedigend","ausreichend", "mangelhaft" sowie "ungenuegend" repräsentiert werden. Fügen Sie der Klasse `Fach` dazu eine private Instanzmethode `umwandeln` hinzu, die den `float`-Wert für die Note entsprechend umwandelt. Für die Umwandlung dürfen Sie den float-Wert für die Note runden (mit Hilfe des Methodenaufrufs `Math.round(note)`) und dann eine Zuordnung gemäß der gängigen deutschen Schulnoten vornehmen: 1 = "sehr gut" bis 6 = "ungenuegend". Werte außerhalb des Intervalls $[1, 6]$ sollen auf den String-Wert "- -" abgebildet werden. Überladen Sie die `toString` Methode dann durch eine öffentliche Version ohne Parameter, die die erste Version dieser Methode mit einem festen Wert (z.B. 50) aufruft.
Hinweis: Ist `s` eine String-Variable, dann erhält man mit dem Aufruf `s.length()` die Länge der Zeichenkette.

(b) Entwickeln Sie eine Klasse `Schueler`, die zwei private Instanzvariablen für den Namen des Schülers (vom Typ `String`) und die Fächer, die dieser Schüler hat (vom Typ `Fach[]`). Fügen Sie der Klasse zwei öffentliche Methoden `getName` und `getFaecher` hinzu, die die Attributwerte an den Aufrufer zurückgeben. Statten Sie die Klasse mit einem Konstruktor aus, der einen Namen für den Schüler sowie ein

Feld von Fächernamen als Parameter besitzt und mit den Parameterwerten die Attributwerte initialisiert. Schreiben Sie nun eine Methode `erfasseErgebnisse`, die die Notenwerte für alle Fächer eines Schülers (mit Hilfe der `IOTools` Klasse) einliest und die Attributwerte in den zugehörigen Instanzen der Klasse Fach setzt. Wiederholen Sie die Eingabe einer einzelnen Note solange, bis der eingegebene Wert im Intervall $[1.0, 6.0]$ liegt

(c) Implementieren Sie eine Klasse `ZeugnisSchreiber` mit einer Klassenmethode `schreiben`. Die Methode soll einen Parameter der Klasse `Schueler` besitzen und beim Aufruf ein Zeugnis für diesen Schüler auf der Kommandozeile ausgeben, welches neben einer Überschrift und dem Namen des Schülers die Noten in allen Fächern beinhaltet. Sorgen Sie dafür, dass für die Ausgabe der Fächer (mit den Noten) die (öffentliche) `toString` Methode der Klasse `Fach` verwendet wird. Berechnen Sie dabei die Durchschnittsnote des Schülers über alle Fächer. Sollte die Durchschnittsnote des Schülers mindestens 1.3 sein, dann soll zusätzlich eine Belobigung hinzugefügt werden (siehe Beispiel unten).

Fügen Sie der Klasse nun noch eine `main`-Methode hinzu, in der Sie zunächst eine Instanz der Klasse `Schueler` generieren. Legen Sie dazu vorher eine Fächermenge (in einem Array) fest und lesen Sie den Namen des Schülers mit Hilfe der `IOTools` ein. Der Schüler soll mindestens vier Fächer haben. Lesen Sie die Noten mit Hilfe der Methode `erfasseErgebnisse` der Klasse `Schueler` ein. Am Ende soll mit Hilfe der Methode `schreiben` ein Zeugnis ausgegeben werden, das den Namen und die Informationen über die in den einzelnen Fächern erzielten Ergebnisse aufführt.

Eine Programmlauf könnte dann z.B. folgende Ausgabe produzieren:

```
Name des Schuelers: Alex Schmidt
Note für Mathematik: 1.0
Note für Deutsch: 1.3
Note für Latein: 1.7
Note für Sport: 1.3
```

```
------------------------------------------------------
Z E U G N I S

Name:  Alex Schmidt

Mathematik                                    sehr gut
Deutsch                                       sehr gut
Latein                                             gut
Sport                                         sehr gut

Der Schueler erhaelt eine Auszeichnung für im
Durchschnitt sehr gute Noten.

------------------------------------------------------
```

## Aufgabe 2: OO: Vererbung 17 (= 9 + 4 + 4) P

In dieser Aufgabe sollen Sie zunächst einen Datentyp entwickeln, der eine dynamisch erweiterbare Sammlung von Werten repräsentiert, die intern in einem Feld gespeichert werden, dessen Größe immer der Anzahl der in der Sammlung enthaltenen Werte entspricht. Beim Einfügen und Löschen von Elementen muss daher das Feld jeweils vergrößert bzw. verkleinert werden. Hierzu soll dann jeweils ein neues, größeres bzw. kleineres Feld mit den entsprechenden Werten angelegt werden, welches dann das ursprüngliche Feld ersetzt.
Die Klasse `DynSequence` definiert eine private Instanzvariable `array`, welche ein Feld für die zu speichernden Werte vom Typ `double` darstellt. Der vorgegebene Konstruktor initialisiert diese Variable mit einem Feld der Länge 0. Die vorgegebene Methode `size` gibt die Länge des Felds zurück.

(a) Ergänzen Sie zunächst die Klasse `DynSequence` wie folgt:

- Schreiben Sie eine Instanzmethode `isEmpty`, die den Wert `true` zurückgibt, wenn die Sequenz leer ist (d.h. wenn das Feld `array` die Länge 0 hat) und den Wert `false` sonst.
- Schreiben Sie eine Instanzmethode `insertValueAt` mit zwei Parametern, die einen einzufügenden Wert und einen Index spezifizieren, welcher die Einfügeposition kennzeichnet. Die Methode

soll nichts zurückgeben. Prüfen Sie zunächst den als Parameter übergebenen Indexwert. Falls dieser negativ ist, setzen Sie ihn auf 0. Ist der Wert zu groß, setzen Sie den Wert auf den (zukünftigen) maximalen Wert. Generieren Sie dann ein neues Feld, welches um ein Element länger als das bisherige Feld `array` ist. Kopieren Sie die bisherigen Werte und fügen Sie dabei das neue Element an der entsprechenden Position ein. Ersetzen Sie dann das alte Feld durch das neue.

- Schreiben Sie eine Instanzmethode `removeValueAt`, welcher ein Element an einem als Parameter übergebenen Index löscht und den gelöschten Wert zurückgibt. Prüfen Sie zunächst den als Parameter übergebenen Indexwert. Ist dieser ungültig in Bezug auf das aktuelle Feld `array`, geben Sie den Wert `Double.NaN` zurück. Anderenfalls legen Sie ein neues Feld an, welches um ein Element kleiner ist als das bisherige. Kopieren Sie die Elemente des alten Felds in das neue und lassen Sie dabei das zu löschende Element aus.
- Überschreiben Sie die von der Klasse `Object` geerbte Instanzmethode `toString`, so dass eine Zeichenkette mit dem Inhalt "Inhalt:", gefolgt von den Elementen des Felds `array` zurückgegeben wird.
- Ergänzen Sie die `main` Methode der Klasse, so dass dort eine Instanz der Klasse `DynSequence` generiert und mit den `double` Werten $\{0.5, 1.5, 2.5, \ldots, 9.5\}$ gefüllt wird. Alle Werte sollen nacheinander am Anfang eingefügt werden. Geben Sie diese Instanz in der Konsole aus. Löschen Sie dann nacheinander das zweitletzte, drittletzte und das viertletzte Element ( auf den jeweils aktuellen Inhalt bezogen) und geben Sie den Inhalt nach jeder Löschoperation nochmals aus.

Zur Laufzeit sollte die Ausgabe dann wie folgt aussehen:

```
Inhalt: 9.5 8.5 7.5 6.5 5.5 4.5 3.5 2.5 1.5 0.5
Inhalt: 9.5 8.5 7.5 6.5 5.5 4.5 3.5 2.5 0.5
Inhalt: 9.5 8.5 7.5 6.5 5.5 4.5 2.5 0.5
Inhalt: 9.5 8.5 7.5 6.5 4.5 2.5 0.5
```

(b) Schreiben Sie nun eine neue Klasse `DynStack`, die von der Klasse `DynSequence` abgeleitet ist. Ein *Stack* (Stapel) ist eine Datenstruktur, bei der die Elemente in einer Sequenz gespeichert werden, bei der Elemente nur an einer Seite eingefügt und entnommen werden. Dabei kann immer nur das zuletzt eingefügte Element entnommen werden.

Implementieren Sie die Klasse wie folgt:

- Schreiben Sie je eine Instanzmethode mit dem Namen `push` und `pop`: die Methode `push` soll einen als Parameter übergebenen Wert am Ende einfügen (und nichts zurückgeben), die Methode `pop` soll das Element am Ende der Sequenz entfernen und als Ergebnis zurückgeben. Verwenden Sie hierzu die aus der Oberklasse geerbten Methoden.
- Ergänzen Sie die Klasse um eine `main` Methode, in der eine Instanz der Klasse `DynStack` erzeugt und mit $n = 5$ (ganzzahligen) Zufallswerten aus dem Intervall $[0, 99]$ gefüllt wird. Geben Sie den Inhalt der Sequenz nach jeder Einfügung aus. Entnehmen Sie danach elementweise alle Werte aus der Sequenz, bis diese leer ist. Geben Sie den Inhalt der Sequenz nach jedem Löschvorgang aus. Summieren Sie die gelöschten Elemente und geben Sie die Summe am Ende aus.

  Die Ausgabe beim Ausführen der Klasse sollte dann wie folgt aussehen:

```
Inhalt: 21.0
Inhalt: 21.0 38.0
Inhalt: 21.0 38.0 53.0
Inhalt: 21.0 38.0 53.0 80.0
Inhalt: 21.0 38.0 53.0 80.0 11.0
Inhalt: 21.0 38.0 53.0 80.0
Inhalt: 21.0 38.0 53.0
Inhalt: 21.0 38.0
Inhalt: 21.0
Inhalt:
Summe: 203.0
```

(c) Schreiben Sie nun noch eine neue Klasse `DynQueue`, die von der Klasse `DynSequence` abgeleitet ist. Eine *Queue* (Schlange) ist eine Sammlung

von Daten, die sequenziell angeordnet sind und bei der man an einem Ende nur Elemente einfügen und am anderen Ende nur Elemente entnehmen kann. Implementieren Sie die Klasse wie folgt:

- Schreiben Sie je eine Instanzmethode mit dem Namen `enqueue` und `dequeue`: die Methode `enqueue` soll einen als Parameter übergebenen Wert am Ende einfügen (und nichts zurückgeben), die Methode `dequeue` soll das Element am Anfang der Sequenz entfernen und als Ergebnis zurückgeben. Verwenden Sie hierzu die aus der Oberklasse geerbten Methoden.
- Implementieren Sie die `main` Methode der Klasse analog zu der `main` Methode der Klasse `DynStack` nun für die Klasse `DynQueue`. Die Ausgabe beim Ausführen der Klasse sollte dann wie folgt aussehen:

```
Inhalt: 62.0
Inhalt: 62.0 71.0
Inhalt: 62.0 71.0 37.0
Inhalt: 62.0 71.0 37.0 37.0
Inhalt: 62.0 71.0 37.0 37.0 94.0
Inhalt: 71.0 37.0 37.0 94.0
Inhalt: 37.0 37.0 94.0
Inhalt: 37.0 94.0
Inhalt: 94.0
Inhalt:
Summe: 301.0
```