— Übungsblatt 9 (P) —

Aufgabe 35 (P) (Instanzmethoden, Konstruktoren)

Schreiben Sie eine Klasse `CD`, die geeignete Instanzvariablen beinhaltet, um den *Titel*, die *Spieldauer* und den *Ausleih-Status* einer CD zu speichern.

Die Klasse soll außerdem einen Konstruktor enthalten, der die drei Instanzvariablen mit entsprechenden Werten versorgt.

Zusätzlich soll die Klasse mit einer Instanzmethode `toString()` ausgestattet sein, die den Titel und die Spieldauer (in Klammern) des `CD`-Objekts, für das sie aufgerufen wird, als Zeichenkette zurückliefert.

Schreiben Sie außerdem eine Test-Klasse, die ihre Klasse `CD` testet, indem die Daten einer CD eingelesen, daraus ein `CD`-Objekt erzeugt und dieses mithilfe der Instanzmethode `toString()` ausgegeben wird.

Ein Beispiel-Programm-Ablauf:

```
Geben Sie die Daten einer CD ein:
Titel: Robbie Williams: Swing When You’re Winning
Spieldauer: 51
ausgeliehen: false
Ausgabe:
Robbie Williams: Swing When You’re Winning (51.0 Minuten)
```

Aufgabe 36 (P) (Sortieren)

Schreiben Sie eine Klasse `Party`, die in ihrer `main`-Methode die Daten mehrerer CDs einliest, die CDs nach ihrer Spieldauer sortiert und eine sortierte Liste derjenigen CDs ausgibt, die im Moment nicht ausgeliehen sind und somit auf einer Party gespielt werden können.

Die `main`-Methode soll zunächst die Anzahl der einzugebenden CDs einlesen und ein entsprechend dimensioniertes Feld erzeugen. Danach soll nur noch je eine Methode für *das Einlesen*, für *das Sortieren* und für *das Ausgeben* des Feldes aufgerufen werden.

Sie müssen also zusätzlich zur `main`-Methode die drei folgenden Klassen-Methoden implementieren:

- `public static void einlesen (CD[] cdf)`
  erzeugt für jede Komponente des Feldes `cdf` ein `CD`-Objekt mit zuvor von der Tastatur eingelesenen Angaben für den Titel, die Spieldauer und den Ausleih-Status.

- `public static void sortieren (CD[] cdf)`
  führt für die Feldkomponenten von `cdf` (nachfolgend als $f_0, f_1, \ldots, f_{n-1}$ bezeichnet) den folgenden Sortier-Algorithmus aus:

  Wiederhole die Schritte (1) und (2) für $i = 0, \ldots, n-2$
  (1) Bestimme den Index m der CD f_m mit der längsten Spieldauer unter den CDs $f_i, f_{i+1}, \ldots, f_{n-1}$.
  (2) Vertausche f_i und f_m.

- `public static void vorhandeneAusgeben (CD[] cdf)`
  gibt die in `cdf` gespeicherten und nicht ausgeliehenen `CD`s mithilfe der Methode `toString` auf den Bildschirm aus.

Ein typischer Programmablauf wäre z. B.:

```
Wieviele CDs willst Du eingeben? 6
Titel: Robbie Williams: Swing When You’re Winning
Spieldauer: 51
ausgeliehen: false
Titel: Pink Floyd: Echoes - The Best Of Pink Floyd
Spieldauer: 145
ausgeliehen: false
Titel: Sting: All This Time - Live
Spieldauer: 54
ausgeliehen: true
Titel: Sarah Connor: Green Eyed Soul
Spieldauer: 43
ausgeliehen: false
Titel: Paul McCartney: Driving Rain
Spieldauer: 51
ausgeliehen: true
Titel: Lighthouse Family: Whatever Gets You Through The Day
Spieldauer: 59
ausgeliehen: false
Nicht ausgeliehene CDs, nach Spieldauer sortiert:
Pink Floyd: Echoes - The Best Of Pink Floyd (145.0 Minuten)
Lighthouse Family: Whatever Gets You Through The Day (59.0 Minuten)
Robbie Williams: Swing When You’re Winning (51.0 Minuten)
Sarah Connor: Green Eyed Soul (43.0 Minuten)
Gesamt-Spieldauer: 298.0 Minuten
```

Testen Sie Ihr Programm (`Party`) auch mit den Daten in der Datei `Party.in`, indem Sie die Standardeingabe mittels

```
java Party < Party.in
```

auf diese Datei umsteuern oder den gesamten Inhalt der Datei `Party.in` in die Zwischenablage kopieren und nach dem Start Ihres Programmes im Konsolenfenster den Inhalt der Zischenablage einzufügen.

Die Datei `Party.in` gibt es zum Download auf den Moodle-Kurs-Seiten (Abschnitt Programmvorlagen).

Aufgabe 37 (P) (OOP)

Laden Sie sich von den Moodle-Kurs-Seiten (Abschnitt Programmvorlagen) die Quellcode-Datei der Klasse `TestSound` herunter und schauen Sie sich die Klasse an.

Die Klasse `TestSound` verwendet ein Objekt der Klasse `Sound` und dessen Instanzmethoden um Daten einer Musikanlage einzulesen bzw. auszugeben und deren Einstellungen zu verändern.

Sie sollen nun die fehlende Klasse `Sound` programmieren. Sie soll Objekte modellieren, die jeweils eine einfache Musikanlage zur Beschallung eines Partyraumes darstellen. Jedes Objekt der Klasse soll private Instanzvariablen zur Speicherung

- der Bezeichnung des Raumes, in dem die Anlage steht,
- der eingestellten Lautstärke (ganzzahlig),
- der eingestellten Bässe (ganzzahlig) und
- der eingestellten Höhen (ganzzahlig)

besitzen. Der Konstruktor der Klasse soll dafür sorgen, dass die Werte seiner vier Parameter in den entsprechenden Instanzvariablen gespeichert werden.

Darüber hinaus ist die Klasse `Sound` mit folgenden öffentlichen Methoden auszustatten:

- `int getLautstaerke()`
  liefert den aktuell eingestellten Lautstärken-Wert des aufrufenden `Sound`-Objekts.
- `void verstaerke (String regler, int wert)`
  erhöht die zu `regler` gehörende Einstellung des aufrufenden `Sound`-Objekts um `wert`.
- `String toString()`
  liefert (wie üblich) eine String-Darstellung für das aufrufende `Sound`-Objekt.

Wenn Sie beide Klassen compiliert haben und die Klasse `TestSound` starten, so sieht der Programmablauf wie folgt aus:

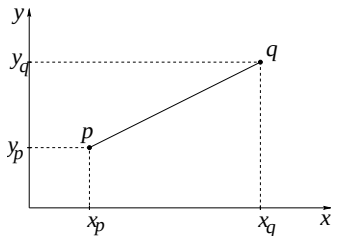
```
Daten fuer eine Anlage einlesen
Raum-Bezeichnung: S116
Standard-Einstellung fuer die Lautstaerke: 5
Standard-Einstellung fuer die Baesse: 4
Standard-Einstellung fuer die Hoehen: 3

Daten der Anlage ausgeben
Anlage im Raum S116: la: 5, ba: 4, ho: 3

Die Anlage wird nun anders eingestellt
(Baesse um 4 erhoeht, Hoehen um 2 erniedrigt)

Daten der neu eingestellten Anlage ausgeben
Anlage im Raum S116: la: 5, ba: 8, ho: 1
```

Aufgabe 38 (P) (OOP)



Ein Punkt p in der Ebene mit der Darstellung $p = (x_p, y_p)$ besitzt die x-Koordinate x_p und die y-Koordinate y_p. Die Strecke $\overline{pq}$ zwischen zwei Punkten $p = (x_p, y_p)$ und $q = (x_q, y_q)$ hat die Länge $L(\overline{pq}) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$.

Unter Verwendung der objektorientierten Konzepte von Java soll in einem Programm mit solchen Punkten und Strecken in der Ebene gearbeitet werden. Dazu soll mit

- einer Klasse `Punkt` zur Darstellung und Bearbeitung von Punkten,
- einer Klasse `Strecke` zur Darstellung und Bearbeitung von Strecken, und
- einer Klasse `TestStrecke` für den Test bzw. die Anwendung dieser beiden Klassen

gearbeitet werden. Gehen Sie wie folgt vor.

(a) Implementieren Sie die Klasse `Punkt` mit zwei privaten Instanzvariablen `x` und `y` vom Typ `double`, die die x- und y-Koordinaten eines Punktes repräsentieren, und statten Sie die Klasse `Punkt` mit Konstruktoren und Instanzmethoden aus. Schreiben Sie

1. einen Konstruktor mit zwei `double`-Parametern (die x- und y-Koordinaten des Punktes),
2. eine Methode `getX()`, die die x-Koordinate des Objekts zurückliefert,
3. eine Methode `getY()`, die die y-Koordinate des Objekts zurückliefert,
4. eine `void`-Methode `read()`, die die x- und y-Koordinaten des Objekts einliest,
5. eine `String`-Methode `toString()`, die die `String`-Darstellung des Objekts in der Form `(xStr,yStr)` zurückliefert, wobei `xStr` und `yStr` die `String`-Darstellungen der Werte von `x` und `y` sind.

(b) Implementieren Sie die Klasse `Strecke` mit zwei privaten Instanzvariablen `p` und `q` vom Typ `Punkt`, die die beiden Randpunkte einer Strecke repräsentieren, und statten Sie die Klasse `Strecke` mit Konstruktoren und Instanzmethoden aus. Schreiben Sie

1. einen Konstruktor mit zwei `Punkt`-Parametern (die Randpunkte der Strecke),
2. eine `void`-Methode `read()`, die die beiden Randpunkte `p` und `q` des Objekts einliest (verwenden Sie dazu die Instanzmethode `read` der Objekte `p` und `q`),
3. eine `double`-Methode `getLaenge()`, die (unter Verwendung der Instanzmethoden `getX` und `getY` der Randpunkte) die Länge des Strecken-Objekts berechnet und zurückliefert,
4. eine `String`-Methode `toString()`, die die `String`-Darstellung des Objekts in der Form `pStr_qStr` zurückliefert, wobei `pStr` und `qStr` die `String`-Darstellungen für die Instanzvariablen p und q des Objekts sind.

(c) Testen Sie Ihre Implementierung mit der vorgegebenen Klasse `TestStrecke`, die Sie auf den Moodle-Kurs-Seiten (Abschnitt Programmvorlagen) finden.

Aufgabe 39 (P) (OOP)

In dieser Aufgabe sollen die Klassen zur Bruchrechnung von Blatt 8 nun objektorientiert implementiert werden.

Erstellen Sie eine neue Klasse `Bruch` mit den `int`-Instanzvariablen `zaehler` und `nenner` und programmieren Sie folgende Methoden:

- eine Klassenmethode `ggt` mit zwei `int`-Parametern, die als `int`-Rückgabewert deren größten gemeinsamen Teiler liefert,
- einen Konstruktor, der die zwei `int`-Parameter `zaehler` und `nenner` den gleichnamigen Instanzvariablen zuweist,
- einen leeren Konstruktor ohne Parameter,
- eine Instanzmethode `toString` ohne Parameter, die den Bruch als String der Form `zaehler/nenner` zurückliefert,
- eine Instanzmethode `kuerze` ohne Parameter, die eine neue Instanz von Bruch zurückliefert, deren Zähler und Nenner um den ggT gekürzt sind,
- eine Instanzmethode `einlesen`, die weder Parameter noch Rückgabewert hat, die Variablen `zaehler` und `nenner` der aktuellen Instanz mit von der Tastatur eingelesenen Werten initialisiert und dabei darauf achtet, dass `nenner` nicht den Wert 0 annimmt,
- eine Instanzmethode `kehrwert` ohne Parameter, die eine neue Instanz von `Bruch` zurückgibt, die den Kehrwert des aktuellen Bruches beinhaltet,
- eine Instanzmethode `neg` ohne Parameter, die eine neue Instanz von `Bruch` zurückgibt, die den negativen Wert des aktuellen Bruches beinhaltet,
- eine Instanzmethode `wert` ohne Parameter, die den aktuellen Bruch als `double`-Wert zurückliefert
- eine Instanzmethode `mul` mit **einem** Parameter vom Typ `Bruch`, die eine neue Instanz von `Bruch` zurückgibt, die das Produkt des aktuellen Bruches und des Parameters beinhaltet,
- eine Instanzmethode `div` mit **einem** Parameter vom Typ `Bruch`, die eine neue Instanz von `Bruch` zurückgibt, die den Quotienten aus dem aktuellen Bruch und dem Parameter beinhaltet,
- eine Instanzmethode `add` mit **einem** Parameter vom Typ `Bruch`, die eine neue Instanz von `Bruch` zurückgibt, die die Summe des aktuellen Bruches und des Parameters beinhaltet und
- eine Instanzmethode `sub` mit **einem** Parameter vom Typ `Bruch`, die eine neue Instanz von `Bruch` zurückgibt, die die Differenz zwischen dem aktuellen Bruch und dem Parameter beinhaltet.

Aufgabe 40 (P) (OOP)

Schreiben Sie nun eine Klasse `BruchTest`, in der die Funktionalität der Klasse `Bruch` aus Aufgabe 39 getestet wird. Definieren und instanziieren Sie die Variablen `b1` und `b2` vom Typ `Bruch` und lassen Sie durch geeignete Methodenaufrufe Werte für beide Brüche von der Tastatur einlesen. Rufen Sie anschliessend alle Methoden auf und lassen Sie deren Ergebnisse auf dem Bildschirm ausgeben.

Beispielablauf:

```
Zaehler eingeben: 1
Nenner eingeben: 2
Zaehler eingeben: 3
Nenner eingeben: 4
Multiplikation: 3/8
Division: 2/3
Addition: 5/4
Subtrakion: -1/4
Doublewert: 0.5
Kehrwert: 2/1
negativ: -1/2
gekuerzt: 1/2
```

Aufgabe 41 (P) (OOP)

Schreiben Sie in einer Klasse `BruchRechner` ein menügesteuertes Bruchrechenprogramm, welches die Eingabe zweier Brüche und das Rechnen mit diesen Brüchen erlaubt. Das Menü soll jeweils die aktuellen Werte der beiden Brüche und die Tastaturkürzel für die möglichen Aktionen anzeigen und nach jeder Aktion wieder neu aufgebaut werden. Dies soll solange geschehen, bis als Aktion „Beenden“ gewählt und das Programm somit beendet wird.

Beispiel für das Menü:

```
Bruch 1: 0/0
Bruch 2: 0/0

Aktionen:
  x: beenden
  1: Bruch 1 eingeben
  2: Bruch 2 eingeben
  b: Brueche kuerzen
  *,/,+,-: Grundrechenarten
  n: Bruch 1 negativ
  k: Kehrwert von Bruch 1
  d: Double-Wert von Bruch 1

  Eingabe:
```