Übung06: Klassen als Datenstrukturen

Abgabetermin: 03.05.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe17	Java Programm, Bildschirmausgaben	Java Programm	☐
Aufgabe18	Java Programm, Testausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe17: Zeichnen einfacher Shapes (12 Punkte)

Die Klasse `Window` aus der Datei `Window.java` erlaubt es, wie in der letzten Übung vorgestellt, einfache grafische Objekte in einem Fenster zu zeichnen. Die Methode `Window.open(int width, int height)` öffnet ein Fenster mit gegebener Breite und Höhe. `(0, 0)` markiert den Koordinatenursprung links oben. Positive x/y Werte verschieben die Elemente nach rechts/unten. Folgende Methoden werden verwendet, um Linien, Rechtecke oder Kreise zu zeichnen:

- `drawLine(int x1, int y1, int x2, int y2)`
- `drawRectangle(int x, int y, int w, int h)`
- `drawCircle(int x, int y, int radius)`

Implementieren Sie ein Programm, welches aus einer gegebenen Datei Breite und Höhe für das zu erzeugende Fenster und gespeicherten Shapes einliest (Linien, Rechtecke und Kreise), und diese anschließend mit der Klasse `Window` visualisiert.

Beispieldatei (shapes.txt):

```
300 300
L 0 200 300 200
L 150 100 200 50
L 200 50 250 100
R 150 100 100 100
R 185 160 30 40
R 170 120 20 20
L 180 120 180 140
L 170 130 190 130
R 210 120 20 20
L 220 120 220 140
L 210 130 230 130
C 50 50 10
L 30 50 40 50
L 35 35 42 42
L 50 30 50 40
L 65 35 58 42
L 60 50 70 50
L 58 58 65 65
L 50 60 50 70
L 42 58 35 65
```

Hinweise:

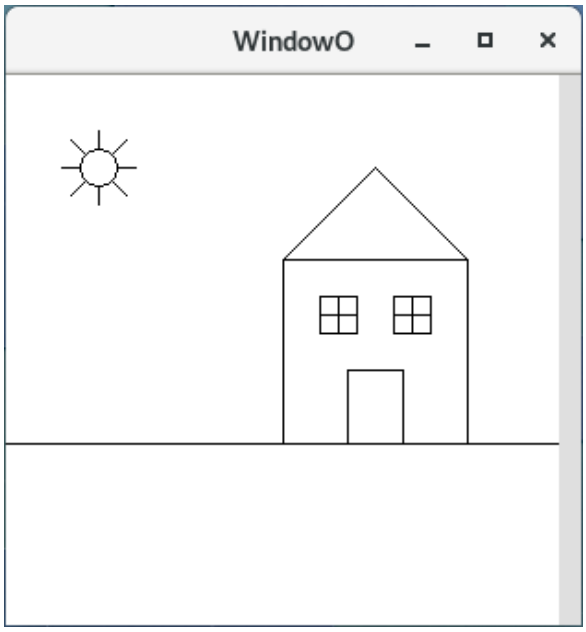
Erstellen Sie die Klassen `Point`, Line, `Rectangle` und `Circle` mit entsprechenden Feldern. `Point` besteht aus zwei Ganzzahlen für x- und y-Koordinaten. Line besteht aus zwei `Point` Objekten für Start- und Endpunkt. `Rectangle` und `Circle` bestehen jeweils aus einem `Point` Objekt für Start- bzw. Mittelpunkt, sowie Ganzzahlen für Breite und Höhe bzw. Radius.

Erstellen Sie weiters für jeden Shape-Typ zwei statische Methoden, die es erlauben, ein Objekt vom jeweiligen Typ zu Erzeugen (indem entsprechende Werte aus einer Datei gelesen werden), sowie ein vorhandenes Objekt in ein Fenster zu zeichnen. Schreiben Sie für das Einlesen von `Point`-Objekten auch eine eigene Methode.

Schreiben Sie eine `main`-Methode, in der Sie die Shapes aus einer gegebenen Datei einlesen und in das Fenster zeichnen. Lesen Sie zu Beginn Breite und Höhe für das Fenster aus der Datei ein und öffnen Sie das Fenster mit den gegebenen Werten.

Lesen Sie anschließend Shape für Shape ein und zeichnen Sie das entsprechende Shape in das Fenster. Sie können anhand des ersten Zeichens in der Zeile prüfen, um welchen Shape-Typ es sich handelt, um die entsprechenden Methoden für das Einlesen und Zeichnen der Shapes aufzurufen.

Beispielausgabe (shapes.txt):



Abzugeben sind:

- Java Programm
- Bildschirmausgaben für simple Beispieldateien

Aufgabe18: Verwaltung von Bankkonten (12 Punkte)

Schreiben Sie für eine kleine Privatbank ein Java Programm zur Verwaltung von Kundenkonten. Erstellen Sie drei Klassen für Kunden (`Customer`), Konten (`Account`) und die Bank selbst (`Banking`).

Ein Kunde (Klasse `Customer`) hat:

- einen Vornamen,
- einen Nachnamen,
- und eine Telefonnummer.

Ein Konto (Klasse `Account`) hat:

- eine Kontonummer,
- einen Kontostand,
- einen Überziehungsrahmen,
- und einen Inhaber (ein `Customer`).

Eine Bank (Klasse `Banking`) besteht aus einer Menge an Konten. Da die Bank nur sehr exquisite Kunden betreut, ist es ausreichend, wenn maximal 100 Konten verwaltet werden können. Verwenden Sie ein Array, um eine Menge an `Account`-Objekten zu speichern. Gehen Sie dabei wie beim Beispiel `Birthdays` aus der Übung vor: Legen Sie ein Array von `Account`-Objekten mit der gegebenen Kapazität an und verwenden Sie eine `count`-Variable, um sich die tatsächliche Anzahl von gespeicherten Objekten zu merken.

Implementieren Sie in der Klasse `Banking` folgende Methoden:

- `static int createAccount(Customer owner, double overdraftLimit)`

  Erstellt ein Konto und liefert die Kontonummer für das erstellte Konto. Die Kontonummer soll hierbei fortlaufend (mit 0 beginnend) vergeben werden. Falls bereits zu viele Konten angelegt worden sind, soll –1 als Kontonummer zurückgegeben werden.

- `static boolean deposit(int accountNo, double amount)`

  Bucht den gegebenen Betrag auf das gegebene Konto und liefert `true`, falls die Einzahlung erfolgreich durchgeführt wurde; ansonsten `false` (wenn z.B. das Konto nicht existiert).

- `static boolean withdraw(int accountNo, double amount)`

  Bucht den gegebenen Betrag vom gegebenen Konto ab und liefert `true`, falls die Abhebung erfolgreich durchgeführt wurde; ansonsten `false` (wenn z.B. das Konto nicht existiert oder der Überziehungsrahmen nicht ausreicht).

- `static boolean transfer(int fromNo, int toNo, double amount)`

  Bucht den gegebenen Betrag vom Konto `fromNo` ab und auf das Konto `toNo` auf. Liefert `true`, falls die Überweisung durchgeführt wurde; ansonsten `false` (wenn z.B. eines der beiden Konten nicht existiert oder der Überziehungsrahmen am Konto `fromNo` nicht ausreicht).

- `static double getAccountBalance(int accountNo)`

  Liefert den aktuellen Kontostand für das gegebene Konto. Existiert das Konto nicht, soll 0 zurückgeliefert werden.

- `static double getBalance()`

  Liefert den Bilanzwert (=Summe aller Kontostände) der Bank.

- `static void printAccounts()`

  Gibt die Liste aller Konten der Bank aus. Beispielausgabe:

```
------------- Bankauszug -------------
Kontonummer: 0
Kontoinhaber: Max Mustermann
Telefonnummer: +43 1 555 666
Kontostand: 1500.0
Überziehungsrahmen: 2000.0
--------------------------------------
Kontonummer: 1
Kontoinhaber: Maximilia Musterfrau
Telefonnummer: +43 1 777 888
Kontostand: 1300.0
Überziehungsrahmen: 2200.0
--------------------------------------
```

Hinweise:

Definieren Sie selbst, wenn notwendig und sinnvoll, weitere Methoden. Testen Sie Ihr Programm in einer `main`-Methode, indem Sie darin händisch mehrere `Customer`- und `Account`-Objekte erzeugen, Überweisungen durchführen, sowie Konten, Kontostände und die Bilanz der Bank zwischen einzelnen Überweisungen ausgeben.

Abzugeben sind:

- Java Programm
- Testausgaben

Optional:

Sie können, um Ihr Programm ausgiebiger zu testen, ein Konsolenmenü zur Verwaltung von Bankkonten implementieren. Das Menü könnte in der `main`-Methode innerhalb einer Schleife ausgegeben werden, und der User könnte nach einer bestimmten Operation gefragt werden. Je nach Operation werden weitere Daten vom User abgefragt, um z.B. Konten und Inhaber zu erstellen, Überweisungen zu tätigen, usw.

Beispiel:

```
********** Bankverwaltung ********
Kunde+Konto anlegen ............. a
Einzahlen ....................... e
Beheben ......................... b
Überweisen ...................... t
Übersicht drucken ............... d
Beenden ......................... q
Welche Menuoption? [a|e|b|t|d|q]: a

*** Kunde und Bankkonto anlagen ***
Vorname: Max
Nachname: Mustermann
Telefonnummer: +43 1 555 666
Überziehungsrahmen: 2000
```

```
Anlegen erfolgreich durchgeführt

********** Bankverwaltung ********
Kunde+Konto anlegen ..... a
Einzahlen ..... e
Beheben ..... b
Überweisen ..... t
Übersicht drucken ..... d
Beenden ..... q
Welche Menuoption? [a|e|b|t|d|q]: e

*** Einzahlen ***
Kontonummer: 0
Einzahlungsbetrag: 1500
Einzahlen erfolgreich durchgeführt

********** Bankverwaltung ********
Kunde+Konto anlegen ..... a
Einzahlen ..... e
Beheben ..... b
Überweisen ..... t
Übersicht drucken ..... d
Beenden ..... q
Welche Menuoption? [a|e|b|t|d|q]: d

---------- Bankauszug ----------
Kontonummer: 0
Kontoinhaber: Max Mustermann
Telefonnummer: +43 1 555 666
Kontostand: 1500,00
Überziehungsrahmen: 2000,00
--------------------------------
Bilanzsumme: 1500,00
--------------------------------

********** Bankverwaltung ********
Kunde+Konto anlegen ..... a
Einzahlen ..... e
Beheben ..... b
Überweisen ..... t
Übersicht drucken ..... d
Beenden ..... q
Welche Menuoption? [a|e|b|t|d|q]: q
```