Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Rudolf Berrendorf

Einführung in die Programmierung
WS 19/20
Ausgabe : 02.12.2019
Bearbeiten bis : 08.12.2019

Übung 10

Aufgabe 1 :
Diese Aufgabe ist eine größere Aufgabe, bestehend aus mehreren Teilaufgaben mit ansteigendem Schwierigskeitsgrad, die aufeinander aufbauen. Hinweise vorab:

- Nutzen Sie ausschließlich Java-Konstrukte, die in der Vorlesung bis jetzt vorkamen.
- Nutzen Sie *keine* Umlaute.
- Sie können zum Beispiel für spätere Teilaufgaben auch weitere Methoden in die geforderten Klasse aufnehmen.
- Alle Methoden (außer **`main`**) sind Instanzmethoden, also *ohne* **`static`**.
- Sie dürfen diese Aufgabenstellung nach eigenem Belieben erweitern.

1. Geben Sie eine Java-Klasse **`Person`** an, die eine Person beschreibt. Eine Person hat einen Vornamen und ein Alter, beides muss man bei der Instanziierung angeben. Geben Sie eine Methode **`public String toString()`** an, die einen String mit Namen und Alter der Person liefert.

   Nutzen Sie folgendes Testprogramm in einem **`main`**, um ihre **`Person`**-Klasse zu testen:

   ```
   public static void main(String[] args) {
       Person paul = new Person("Paul", 16);
       // hier wird implizit die toString-Methode aufgerufen
       System.out.println(paul);
   }
   ```

   Die Ausgabe dazu ist:

   ```
   Paul(16)
   ```

2. Geben Sie in Java eine Klasse **`Getraenk`** an, die eine Getränkeportion in einem Restaurant / Bar beschreibt. Zu einer solchen Beschreibung gehören der Name, die Abgabemenge pro Glas in ml (ganzzahlig), der Preis in Cent dafür (ebenfalls ganzzahlig) sowie ein Freigabealter (ganzzahlig), ab dem ein Gast dieses Getränk bestellen kann. Sehen Sie einen Konstruktor vor, der aus solchen Angaben ein entsprechendes Java-Objekt erzeugt. Die Klasse soll eine **`toString`**-Methode haben, die einen Text zu den gespeicherten Angaben zu diesem Getränk liefert.

   Nutzen Sie folgendes Testprogramm in einem **`main`**, um ihre **`Getraenk`**-Klasse zu testen:

   ```
   public static void main(String[] args) {
       // Getraenke erzeugen
       Getraenk cola  = new Getraenk("Cola", 200, 250, 0);
       Getraenk bier  = new Getraenk("Bier", 200, 200, 16);
       Getraenk vodka = new Getraenk("Vodka", 20, 300, 18);

       // Getraenk auf Bildschirm ausgeben
       // hier wird implizit die toString-Methode aufgerufen
       System.out.println(cola);
   }
   ```

   Die Ausgabe dazu ist:

   ```
   Name: Cola, Menge: 200, Preis: 250, Altersfreigabe: 0
   ```

3. Geben Sie eine Java-Klasse **`Getraenkekarte`** für Getränkekarten an. Auf einer Getränkekarte gibt es insgesamt $n \in \mathbb{N}$ Getränke (in Java also Getränkeobjekte), jeweils wie oben beschrieben. Man kann eine leere Getränkekarte erzeugen und anschließend ein Getränk(-objekt) nach dem anderen hinzufügen. Hinweis: Feld, das jeweils neu und größer angelegt wird mit kopiertem Inhalt des alten Feldes. Mit einer Methode **`toString`** kann man sich die

gesamte Getränkekarte *als String* (über mehrere Zeilen) geben lassen, pro Zeile ein Getränk. Weiterhin kann man in einer Getränkekarte über einen Getränkenamen eine Getränk suchen. Das Ergebnis einer erfolgreichen Suche ist das entsprechende Java-Objekt für das Getränk. Wenn man kein solches Getränk findet, soll diese Methode `null` liefern.

Nutzen Sie folgendes Testprogramm in einem `main`, um die Getränkekarte zu testen:

```
public static void main(String[] args) {
    // leere Karte erzeugen
    Getraenkekarte gk = new Getraenkekarte();
    // Getraenke erzeugen
    gk.neuesGetraenk(new Getraenk("Cola", 200, 250, 0));
    gk.neuesGetraenk(new Getraenk("Bier", 200, 200, 16));
    gk.neuesGetraenk(new Getraenk("Vodka", 20, 300, 18));

    // Karte auf dem Bildschirm anzeigen lassen
    System.out.println(gk);

    // Getraenk suchen
    Getraenk g = gk.suchen("Cola");
    if(g != null) {
        System.out.println(g);
    }
}
```

Die Ausgabe dazu ist:

```
Name: Cola, Menge: 200, Preis: 250, Altersfreigabe: 0
Name: Bier, Menge: 200, Preis: 200, Altersfreigabe: 16
Name: Vodka, Menge: 20, Preis: 300, Altersfreigabe: 18

Name: Cola, Menge: 200, Preis: 250, Altersfreigabe: 0
```

4. Geben Sie eine Java-Klasse `Lounge` an, die eine Lounge (Bar) beschreibt. In einer Lounge gibt es bei Instanziierung eine Getränkekarte (mit Inhalt). Siehe Beispiel oben. Weiterhin kann eine Person in einer Lounge über den Getränkenamen eine Bestellung aufgeben. In diesem Fall muss überprüft werden, ob das Getränk auf der Karte steht und ob die Person alt genug für dieses Getränk ist. Im Fehlerfall soll eine entsprechende Nachricht auf dem Bildschirm ausgegeben werden.

   Nutzen Sie folgendes Testprogramm in einem `main`, um die Lounge zu testen:

```
public static void main(String[] args) {
    // erzeuge Lounge mit Standardgetraenkekarte
    Lounge lounge = new Lounge();

    // erzeuge eine Person
    Person paul = new Person("Paul", 16);

    // Paul laesst sich die Karte zeigen
    // Zum Verstaendnis: hier wird implizit die toString-Methode
    // des Getraenkekartenobjekts dieses Lounge-Objekts aufgerufen
    System.out.println("Unsere Karte:\n" + lounge.karte);

    // Paul bestellt
    lounge.bestellen(paul, "Vodka");
    lounge.bestellen(paul, "Bier");
}
```

   Die Ausgabe dazu ist:

```
Unsere Karte:
Name: Cola, Menge: 200, Preis: 250, Altersfreigabe: 0
Name: Bier, Menge: 200, Preis: 200, Altersfreigabe: 16
Name: Vodka, Menge: 20, Preis: 300, Altersfreigabe: 18

nicht alt genug fuer das Getraenk
bitte schoen, ihr Getraenk
```

Aufgabe 2 :

Programmieren Sie eine Klasse `Rueckgabeautomat`, die die minimale Anzahl an vorhandenen Geldstücken zurückgibt,

die auf einen Einzahlbetrag *einzahlung* (Beispiel 10 Euro) und einen zu zahlenden Betrag *zahlbetrag* (Beispiel 8,57 Euro) sich aufgrund des aktuellen Münzbestands im Automaten ergeben.

Ein Automat kann nur Geldstücke der Wertigkeit 1,2,5,10,20,50,100 und 200 Cent enthalten. Beim Erzeugen eines Geldautomaten gibt man als Argument an, wieviele Geldstücke davon zu Beginn im Automaten enthalten sein sollen (ein Feld mit 8 Zahlen).

Geben Sie eine Instanzmethode an, die den aktuellen Münzbestand ermittelt. Das Ergebnis ist die Anzahl der Münzen in den entsprechenden Wertigkeiten (also wieder ein int-Feld der Größe 8).

Sehen Sie weiterhin eine Instanzmethode vor, die für einen Einzahlbetrag und den tatsächlich zu zahlenden Betrag die Anzahl an Münzen mit der entsprechenden Wertigkeit des Rückgelds berechnet und als Ergebnis der Methode liefert. Falls der aktuelle Bestand kein korrektes Rückgeld erlaubt, geben Sie eine Fehlermeldung auf dem Bildschirm aus. Das Ergebnis der Methode ist wiederum ein Feld der Länge 8 mit der Anzahl zu der entsprechenden Wertigkeit.

Beispiel:
Ein Rückgabeautomat enthält die folgende Anzahl an Geldstücken für die vorgegebenen Wertigkeiten
`[200,100,50,20,10,5,2,1]: [2,2,2,2,2,2,2,2]`
also jeweils zwei Geldmünzen der entsprechenden Wertigkeit. Auf den Einzahlbetrag 1 Euro und den tatsächlich zu zahlenden Betrag von 57 Cent gibt der Automat als Ergebnis `[0,0,0,2,0,0,1,1]` (43 Cent als 2 Zwanziger, 1 Zweier und 1 Einer) und hat als Bestand danach noch `[2,2,2,0,2,2,1,1]`.

Instanziieren Sie zwei Rückgabeautomaten mit unterschiedlichem Bestand und lassen sich für den zu zahlenden Betrag von 57 Cent auf den Betrag von 100 Cent Rückgeld geben.

Aufgabe 3 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Erweitern Sie ihre Lösung aus der letzten Übung zur Studierenden-Klasse. Passen Sie ggfs. ihre Lösung gegenüber der Musterlösung so an, dass die Typen ihrer Instanzvariablen mit denen der Musterlösung übereinstimmen. Die Erweiterung in dieser Aufgabe soll so sein, dass die Klasse zusätzlich

1. einen Konstruktor erhalten soll, mit dem man eine(n) Studierende(n) über Angaben zu Vornamen, Nachnamen sowie Matrikelnummer erzeugen kann.
2. eine Methode `public void neueNote(float note)` enthält, die genau eine neue Note zu einer Prüfung zu den Noten hinzufügt, wodurch auch der Notendurchschnitt aktualisiert werden muss.
3. eine Methode `public float getNotendurchschnitt()`, die den aktuellen Notendurchschnitt liefert.
4. eine Methode `public String toString()` enthält, die einen String mit Daten zu eine(n) Studierende(n) liefert.

Geben Sie weiterhin (insbesondere für ihre eigenen Tests in der Entwicklung) in der Klasse eine Methode `main` an, in der nacheinander zwei Studierende erzeugt werden, einige Noten dazu nacheinander eingetragen werden und diese Studierende auf dem Bildschirm ausgegeben werden (siehe etwa Beispiel unten).

Beispiel: Zu folgendem `main`

```
public static void main(String[] args) {
        // 2 neue Studierende
        Studierender s1 = new Studierender("Willi", "Winzig", 123456);
        Studierender s2 = new Studierender("Helga", "Hurtig", 123457);

        // Noten werden eingetragen
        s1.neueNote(3.0F);
        s1.neueNote(2.3F);
        s2.neueNote(1.3F);

        // auf Bildschirm ausgeben
        System.out.println(s1);
        System.out.println(s2);
}
```

wäre eine mögliche Ausgabe:

```
Name: Willi Winzig, Matr.Nr.: 123456, Ergebnisse: 3.0 2.3 , Notendurchschnitt: 2.65
Name: Helga Hurtig, Matr.Nr.: 123457, Ergebnisse: 1.3 , Notendurchschnitt: 1.3
```

Es war in der Aufgabe kein bestimmtes Ausgabeformat vorgegeben. Sie dürfen diese Informationen also auch in anderer sinnvoller Form ausgeben.

Aufgabe 4 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Erweitern Sie ihre Lösung aus der letzten Übung zur Musikstueck-Klasse dahingehend, dass die Klasse zusätzlich

1. einen Konstruktor erhalten soll, mit dem man ein neues Musikstück erzeugen kann. Der Konstruktur bekommt als Argumente die Interpreten (Feld von Strings), den Titel (String) und die Töne (Feld von int).
2. eine Methode `public void abspielen()` enthält, die die Noten des Stücks in einer Ausgabezeile nacheinander auf dem Bildschirm ausgibt (gefolgt von einem Zeilenende dieser Zeile).
3. eine Methode `public void neueBewertung(int bewertung)` enthält, in der genau eine neue Einzelbewertung angegeben wird, die in die Gesamtbewertung zu diesem Musikstück einfließen soll, wobei dabei gerundet werden soll, um zu einem erlaubten ganzzahligen Wert zwischen 1 und 5 zu kommen. Wie geht das, wenn man die alten Einzelbewertungen nicht behalten hat? Wie kann man runden?
4. eine Methode `public int getGesamtbewertung()`, die den aktuellen Wert der Gesamtbewertung zu einem Musikstück liefert.
5. eine Methode `public String toString()` enthält, die einen String mit Daten zu einem Musikstück liefert.

Geben Sie weiterhin (insbesondere für ihre eigenen Tests in der Entwicklung) in der Klasse eine Methode `main` an, in der nacheinander zwei Musikstücke erzeugt werden, Bewertungen eingetragen werden usw. (siehe etwa Beispiel unten).

Beispiel: Zu folgendem `main`

```
public static void main(String[] args) {
        String[] interpreten1 = {"Wham"};
        String[] interpreten2 = {"Koelner Domspatzen", "Heino"};
        int[] toene1 = {1,2,3,4,5,6};
        int[] toene2 = {6,5,4,3,2,1};

        Musikstueck m1 = new Musikstueck(interpreten1, "Last Christmas", toene1);
        Musikstueck m2 = new Musikstueck(interpreten2, "O Tannenbaum", toene2);

        m1.abspielen();
        m1.neueBewertung(1);
        m1.neueBewertung(1);
        m1.neueBewertung(3);
        System.out.println(m1);
        System.out.println(m2);
    }
```

wäre eine mögliche Ausgabe:

```
1 2 3 4 5 6
Interpreten: Wham, Titel: Last Christmas, Toene: 1 2 3 4 5 6 , Anz.Bewertungen: 3, Gesamtbewertung: 2
Interpreten: Koelner Domspatzen,Heino, Titel: O Tannenbaum, Toene: 6 5 , Anz.Bewertungen: 0, Gesamtbewertung: 0
```

Es war in der Aufgabe kein bestimmtes Ausgabeformat vorgegeben. Sie dürfen diese Informationen also auch in anderer sinnvoller Form ausgeben.

Aufgabe 5 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Hinweis: Diese Aufgabe entspricht ungefähr einer früheren Klausuraufgabe, für die 27 Minuten Berarbeitungszeit veranschlagt wurden / 27 Punkte vergeben wurden.

Modellieren Sie einen Kalender für Nichtschaltjahre in Java. Beschreiben Sie dazu den Kalender, Jahre und Tage durch jeweils eine Klasse.

Ein `Tag`-Objekt kann einen Termin haben. In diesem einfachen Kalender ist maximal ein Termin pro Tag zulässig. Ein solcher Termin ist beschrieben durch einen `String` und einen `int`-Wert von 0 bis 10, der die Priorität/Wichtigkeit dieses Termins angibt. Gibt es keinen Termin für den Tag, ist die Beschreibung `null` und die Priorität 0. Die `Tag`-Klasse besitzt mindestens diese Instanzmethoden:

- Die Methode `public void eintragen(String was, int prioritaet)` trägt zu einem Tag einen neuen Termin ein (ohne Überprüfung, ob schon ein Termin zu diesem Tag existiert).
- Die Methode `public String getVerabredung()` liefert den String mit der Verabredung an diesem Tag.
- Die Methode `public int getPrioritaet()` liefert die Priorität des Termins.

Falls der Termin zu einem Tag leer ist, so werden die oben angegeben Werte für einen leeren Termin von den beiden letzten Methoden geliefert.

Ein `Jahr`-Objekt besitzt eine Jahreszahl (z.B. 2018) und 365 `Tag`-Objekte. Jeder Tag in einem Jahr wird durch eine Nummer von 1 bis 365 identifiziert. Alle Jahre haben also 365 Tage, Schaltjahre werden als solche also nicht berücksicht (haben auch 365 Tage). Sehen Sie folgenden Konstruktoren / Instanzmethoden vor:

- Sehen Sie einen Konstruktor vor, der eine Jahresangabe annimmt (z.B. 2018) und einen leeren Kalender mit 365 (leeren) Tag-Objekten zu diesem Jahr erzeugt.
- Eine Methode `public void eintragen(int tag, String was, int prioritaet)` trägt zu einem Tag einen neuen Termin ein. Ist für diesen Tag aber schon ein Termin eingetragen, so soll die Fehlermeldung "Fehler: was" auf dem Bildschirm ausgegeben werden (was soll der Text des schon vorhandenen Termins sein). Beispiel: zum vorherigen Aufruf `bezugsobjekt.eintragen(13, "Praktomatabgabe", 10)` soll die Ausgabe für einen weiteren Aufruf `bezugsobjekt.eintragen(13, "erholen", 7)` sein: `Fehler: Praktomatabgabe`
- Zwei Methoden `public String getTermin(int tag)` und `public int getPrioritaet(int tag)`, die zu dem Tag mit der Nummer `tag` den Termin bzw. die Priorität liefern. Falls zu diesem Tag kein Termin existiert, werden die Default-Werte geliefert (siehe Tag). Hinweis: Über eine Abfrage `bezugsobjekt.getTermin(25) == null` lässt sich also herausfinden, ob für den 25. Tag ein Termin vergeben ist oder nicht.
- Eine Methode `public int getBelegt()` liefert die Anzahl der belegten Tage des Jahres. Überlegen Sie sich, ob Sie bei jeder solchen Anfrage diese Tage ermitteln wollen oder ob es vielleicht ratsamer ist, diese Anzahl (in der Klasse) nachzuhalten.

Erzeugen Sie in der `main`-Methode der Klasse `Kalender` zwei Objekte für die Jahre 2018 und 2019. Tragen Sie für den 45. Tag des Jahres 2018 als Termin `"Klausur"` mit Priorität 1 ein und für den 37. Tag des Jahres 2019 als Termin `"Treffen"` mit Priorität 5. Fragen Sie anschließend ab, ob am 17. Tag des Jahres 2018 ein Termin vorliegt (das Ergebnis müsste `null` sein) und fragen SDie den 45. Tag in 2018 ab (sollte ungleich `null` sein) und geben Sie das Ergebnis auf dem Bildschirm aus (Format siehe unten im Beispiel). Abschließend geben Sie die Gesamtzahl aller belegten Termine für die beiden Jahre (also die Summe zu den beiden Jahren) auf dem Bildschirm aus (das Ergebnis müsste für das Beispiel 2 sein (1+1).

Die Ausgabe dazu sollte dann sein:

```
17. Tag ist frei
45. Tag ist belegt
Belegte Tage insgesamt: 2
```

Hinweise:

- Der Praktomat testet direkt ihre Methoden, Sie müssen sich also strikt an die Vorgaben der Methodensignaturen halten.
- Nummerierungen von Tagen fangen bei 1 an, Feldindizes in Java bei 0.
- Sie programmieren also drei Klassen, jeweils in einer eigenen Datei `Tag.java`, `Jahr.java` bzw. `Kalender.java`. Laden Sie diese Dateien einzeln im Praktomat hoch. Also **kein** zip-Archiv o.Ä. erstellen!

Aufgabe 6 :
Enwickeln Sie in dieser Aufgabe Klassen für Flugzeuge. Die Basisklasse soll die Klasse **`Flugzeug`** sein. Jedes Flugzeug hat eine Spannweite (reelle Zahl) und eine Sitzkapazität (natürliche Zahl).

Ein Segelflugzeug ist ein Flugzeug, das eine beliebige Spannweite hat, die man bei Erzeugung angeben muss, aber immer nur einen Sitzplatz für den Piloten (Schulungssegelflugzeuge mit 2 Sitzen betrachten wir hier nicht).

Ein Passagierflugzeug ist ein Flugzeug, hat aber (im Gegensatz zu einem Segelflugzeug) Motoren mit einem bestimmten Gesamtschub (in kN). Beim Erzeugen eines Passagierflugzeugs muss man Spannweite, Sitzplatzzahl und Schub eines solchen Flugzeugs angeben.

Eine A380 ist ein Passagierflugzeug, das 79,8 m Spannweite hat, 558 Sitzplätze und 4 Motoren mit jeweils 320 kN (mit dem Trent 972 Triebwerk).

Entwickeln Sie geeignete Klassen mit einer entsprechenden Klassenhierarchie. Sehen Sie in jeder Klasse einen Konstruktor vor, der der obigen Beschreibung angemessen ist. Sehen Sie in jeder Klasse eine **`public String toString()`**-Methode vor, die wesentliche Eigenschaften eines Objekts dieser Klasse als String liefert. Zählen Sie weiterhin (innerhalb der Klassen!), wieviele Flugzeuge von jeder Kategorie erzeugt wurden und sehen Sie entsprechende getter-Funktionen vor.

Nehmen Sie folgendes **`main`** in einer Klasse **`FlugzeugTest`** als Test:

```
    public static void main(String[] args) {
        // ein Segelflugzeug mit 17m Spannweite erzeugen
        Segelflugzeug sf = new Segelflugzeug(17.0);
        System.out.println(sf);

        // ein Passagierflugzeug mit 10,97 Spannweite, 4 Plaetzen und 1,3 kN Schub (Cessna 172 P)
        Passagierflugzeug pg = new Passagierflugzeug(10.97, 4, 1.3);
        System.out.println(pg);

        // eine A380 erzeugen (Jede A380 sieht gleich aus.)
        A380 a380 = new A380();
        System.out.println(a380);

        // XXX hier die Anzahl jeder Kategorie jeweils ausgeben
}
```

Ergänzen Sie an der mit **`XXX`** markierten Stelle Code, so dass die Anzahl aller erzeugten Kategorien ausgegeben wird. Für das obige Beispiel müsste die Ausgabe etwa sein (eine A380 ist auch ein Passagierflugzeug):

```
Spannweite: 17.0, Sitze: 1 (Segelflugzeug)
Spannweite: 10.97, Sitze: 4, Schub: 1.3 (Passagierflugzeug)
Spannweite: 79.8, Sitze: 558, Schub: 1280.0 (Passagierflugzeug) A380
Anzahl Fluzgzeuge: 3
Anzahl Segelfluzgzeuge: 1
Anzahl Passagierfluzgzeuge: 2
Anzahl A380: 1
```

Aufgabe 7 :
Eine mathematische Funktion f bildet eine Zahl x (Parameter) auf eine neue Zahl $f(x)$ (Funktionswert) ab. Aus der Sicht von Java kann das durch eine Methode mit dem Kopf **`double map(double x)`** ausgedrückt werden, die den entsprechenden Funktionswert berechnet.

1. Entwickeln Sie eine Basisklasse **`Function`** mit dieser Methode. Diese realisierte Methode soll in der Basisklasse die Identitätsfunktion sein mit $map(x) = x$.
2. Eine Parabel ist eine Funktion der Form $f(x) = ax^2 + bx + c$. Definieren Sie eine Klasse **`Parabel`**, die von **`Function`** abgeleitet ist und deren Konstruktor Werte für a, b, und c akzeptiert.
3. Definieren Sie ebenso eine Klasse **`Hyperbel`** für Funktionen der Form $f(x) = \frac{a}{x}$. Wie muss hier der Konstruktor aussehen?

4. Ein Funktionsverlauf kann in einer Wertetabelle grob dargestellt werden, die für eine Reihe von x-Werten in regelmäßigen Abständen jeweils den Funktionswert $f(x)$ auflistet. Definieren Sie in der passenden Klasse eine Methode `print`, die drei Parameter akzeptiert (kleinster x-Wert, größter x-Wert, Abstand zwischen zwei x-Werten), und die eine derartige Wertetabelle auf dem Bildschirm ausgibt.

5. Schreiben Sie ein Java-Programm, das die Funktion $f(x) = x^2 - 2x + 2$ erzeugt und deren Wertetabelle im Bereich $-5 \leq x \leq 5$ in Schritten von $0,1$ ausgibt.

6. (*) Zwei Funktionen f und g können verkettet werden. Die Verkettung ist eine neue Funktion $h(x) = g(f(x))$. Definieren Sie eine Klasse `Composed`, die von `Function` abgeleitet ist und im Konstruktor zwei andere Funktionen akzeptiert, deren Verkettung Sie repräsentiert.

7. (*) Schreiben Sie ein Java-Programm, das die Funktion $f(x) = \frac{1}{x^2-2x+2}$ erzeugt und deren Wertetabelle im Bereich $-5 \leq x \leq 5$ in Schritten von $0,1$ ausgibt.

(nach R.Schiedermeier)