

Modellierung und Programmierung 1 Übungsserie 2

Abgabetermin: 10.11.2013, 23:55 Uhr

Grundsätzlich sind Nebenrechnungen anzugeben und Antworten zu begründen.
Einzureichen sind, bei mehreren Dateien als .zip-Archiv:
Lösungen als .pdf-Datei, Programmquellcode und Ergebnisdateien.

1. Ausdrücke

Der Praktikant der Duff-Brauerei¹ hat an seinem ersten Arbeitstag ein Programmgerüst für die Verwaltung von Verpackung und Auslieferung erhalten. Die Programmteile a) und b) sollen unter Berücksichtigung folgender Punkte vervollständigt werden:

- Ein Kasten kann bis zu 20 Flaschen enthalten.
- Eine Palette bietet Platz für 50 Kästen.
- Ein LKW kann mit maximal 30 Paletten bestückt werden.
- Ab einer Anzahl von 100 Paletten gibt es 5 Prozent Rabatt auf den Flaschenpreis.
- Die Fixkosten betragen für jeden LKW zusätzlich 200.00 (Euro).

Duff.java (Grobstruktur)

```
import Tools.IO.*;                                // Eingaben
public class Duff
{
    public static void main (String args[])
    {
        // Eingabe der gegebenen Werte
        int anzahlFlaschen = IOTools.readInteger( "Anzahl von Flaschen: ");
        double einzelPreis = IOTools.readDouble( "Preis pro Flasche: ");

        // Deklaration der gesuchten Werte
        int anzahlKaesten, anzahlPaletten, anzahlLKW;
        double flaschenPreis, rabatt, fixKosten, gesamtPreis;

        // a) Berechnen der gesuchten Werte

        // b) Ausgabe der gesuchten Werte

    }
}
```

2. Schleifen- und Auswahlanweisungen

Eine vom Computer mittels Zufallszahlengenerator erzeugte natürliche Zahl aus einem vorher festgelegten Intervall $[1, max]$ soll erraten werden. Jeder Rateversuch wird mit "zu gross", "zu klein" oder "richtig" beantwortet. Das Programm wird nach Erraten der Zahl beendet.

Hinweis: **Math.random()** liefert eine Zufallszahl im Intervall $[0, 1)$ vom Typ **double**.

¹<http://simpsonspedia.net/index.php?title=Duff-Brauerei>

-
- a) Entwerfen Sie das Programm **Zahlenraten.java**, welches dieses Spiel umsetzt.
 - b) Die Anzahl der benötigten Versuche soll gezählt und nach erfolgreichen Raten ausgegeben werden. Erweitern Sie entsprechend Ihr Programm.
 - c) Formulieren Sie eine geeignete Strategie, mit der man möglichst schnell die Zufallszahl erraten kann. Wieviel Schritte sind im allgemeinen maximal nötig? Warum?

3. Verschachtelte Schleifenanweisungen

Ein Primzahlzwillingspaar ist ein Paar von Primzahlen $\{p_1; p_2\}$, deren Abstand genau 2 beträgt. Die kleinsten Primzahlzwillingspaare sind $\{3; 5\}$, $\{5; 7\}$ und $\{11; 13\}$.

- a) Beweisen Sie, dass die Zahl zwischen p_1 und p_2 (mit Ausnahme von $\{3; 5\}$) stets durch 6 teilbar sein muss.
- b) Entwerfen Sie ein Programm **Twins.java**, welches alle Primzahlzwillingspaare von 1 bis 10^3 findet. Gehen Sie dabei effizient vor, indem Sie Ihr Ergebnis aus Teilaufgabe a) berücksichtigen.
- c) Lassen Sie alle gefundenen Paare mittels Ausgabeumlenkung in **Twins.out** notieren. (Obwohl die Existenz unendlich vieler Primzahlen bewiesen wurde, ist die Existenz unendlich vieler Primzahlzwillingspaare bis heute ungeklärt.)

4. Fehlerfortpflanzung

Wallis² lieferte mit dem unendlichen Produkt

$$p(n) = \prod_{n=1}^{\infty} \frac{4n^2}{4n^2 - 1} = \frac{4}{3} \cdot \frac{16}{15} \cdot \frac{36}{35} \cdot \dots = \frac{\pi}{2}$$

eine Approximation für π . Euler³ konnte mit der Reihenentwicklung

$$s(n) = \sum_{n=1}^{\infty} \frac{1}{n^2} = 1 + \frac{1}{4} + \frac{1}{9} + \dots = \frac{\pi^2}{6}$$

ebenfalls Näherungswerte finden.

- a) Entwerfen Sie Programme namens **Wallis.java** und **Euler.java**, in welchem die Verfahren implementiert sind. Die Iteration soll abgebrochen werden, sobald sich die Summe bzw. das Produkt nicht mehr ändern. Verwenden Sie den Datentyp `long` für n und `double` für $p(n)$ und $s(n)$.
- b) Lassen Sie das Ergebnis und den *absoluten Fehler* von $p(n)$ zu $\frac{\pi}{2}$ bzw. von $s(n)$ zu $\frac{\pi^2}{6}$ durch Ausgabeumlenkung in **Wallis.out** bzw. **Euler.out** ausgeben. Vermeiden Sie dabei das Wurzelziehen aus $6 \cdot s(n)$.

Hinweis: Zugriff auf π als Konstante können Sie in Java mittels **Math.PI** erhalten.

²John Wallis (1616-1703)

³Leonhard Euler (1707-1783)