22 Übungen und Aufgaben

WICHTIG

Kapitel 2.4.5

Parallel zu den nun folgenden Aufgaben zum Themenkomplex "Kontrollstrukturen" können die entsprechenden Struktogrammdarstellungen eingeführt und zur Programmentwicklung genutzt werden.

Lineare Struktur

Anweisung 1
Anweisung 2

Jeder Programmbefehl bzw. jede **Anweisung** wird durch ein Rechteck dargestellt. Mehrere Anweisungen werden zu Blöcken zusammengefasst.

Verzweigung

Bedingung	
ja	nein
Block 1	Block 2

Bedingung	
ja	nein
Block 1	

Wenn die **Bedingung** erfüllt ist, dann erfolgt die Auswertung der Anweisungen des **ja-Blocks**. Ansonsten werden die Anweisungen des **nein-Blocks** durchlaufen. Sollen im jeweiligen Fall keine Anweisungen ausgeführt und das Programm direkt nach der Bedingungen fortgesetzt werden, kann man die **ja-** und **nein-Blöcke** auch weglassen bzw. leer lassen.

22.2.8 Analyse der Umgebung

Bezug / Voraussetzung	
Kapitel	2.3.1 Bedingungen 22.1.5 Umgebung

In der Aufgabe 22.1.5 "Grafische **_umgebung**" sollten Sie unter anderem das Beispiel **_umgebung_beispiel-DreiEingaben** untersuchen und modifizieren. In diesem Beispiel sind folgende **if**-Bedingungen zu finden.

```
print("1. ganze Zahl eingeben: ");
if (eingabeIndex >= 0) // 1. Eingebe
{
   println(_Eingabe.getInt(0));
   print("2. ganze Zahl eingeben: ")
}
if (eingabeIndex >= 1) // 2. Eingebe
{
   println(_Eingabe.getInt(1));
   print("3. ganze Zahl eingeben: ");
}
if (eingabeIndex >= 2) // 3. Eingabe
{
   println(_Eingabe.getInt(2));
   // ... u.s.w.
```

Inzwischen kennen Sie die **if**-Bedingung und können das Programm und die Funktionsweise verstehen.

Aufgabe

1. Versuchen Sie (zunächst ohne es konkret auszuprobieren) vorherzusagen, wie sich die Programmausgabe verändert, wenn in den **if**-Anweisungen der Vergleichsoperator **>=** durch **==** ersetzt wird.
2. Ersetzen Sie nun den Vergleichsoperator **>=** durch **==** und überprüfen Sie ihre Prognosen.
3. Ziehen Sie aus dem beobachteten Verhalten Rückschlüsse auf die Arbeitsweise der Programmumgebung.

22.2.9 Zahlenspiel

Bezug / Voraussetzung	
Kapitel	2.3.1 Bedingungen

Zwei Spieler (Spieler A und Spieler B) geben unabhängig voneinander je eine nicht negative ganze Zahl ein.
Nennen beide Spieler die gleiche Zahl, so endet das Spiel unentschieden.
Andernfalls gewinnt, falls die Summe der genannten Zahlen gerade ist, der Spieler, der die kleinere Zahl genannt hat, und sonst derjenige, der die größere Zahl genannt hat.

Aufgabe

1. Schreiben Sie ein Programm, das den Gewinner des Zahlenspiels ermittelt.
2. Testen Sie Ihr Programm mit den folgenden Zahlenpaaren für die beiden Spieler **A** und **B**:

   a) A: -1, B: 7 → Eingabefehler
   b) A: 5, B: 5 → Unentschieden
   c) A: 3, B: 7 → Spieler A gewinnt!
   d) A: 4, B: 7 → Spieler B gewinnt!

Hinweis

Vergessen Sie nicht zu testen, ob die eingegebenen Zahlen auch wirklich positiv sind! Andernfalls ist die Warnung "Eingabefehler" anzuzeigen!

Wenn Sie die grafische **_umgebung** nutzen möchten, so führen Sie diese Aufgabe in der Funktion **meinProgramm()** der Klasse **MeinProgramm** aus.

22 Übungen und Aufgaben

22.2.10 Skatclub

Bezug / Voraussetzung	
Kapitel	2.3.1 Bedingungen

Der Skatclub "Grand-Hand" möchte einen "Schnuppernachmittag" veranstalten und lädt über die lokale Presse öffentlich dazu ein. Natürlich dreht sich hier fast alles um das Skatspiel, dennoch steht das gesellige Beisammensein an diesem besonderen Tag im Vordergrund. Deshalb ist es den Veranstaltern wichtig, dass wirklich ALLE erschienenen Interessenten in einer Spielerunde eingebunden werden. Um das gewährleisten zu können, entscheiden die Organisatoren so viele Skatrunden (bestehend aus jeweils drei Spielern) wie möglich zu bilden. Sollten dann allerdings einige Spieler übrigbleiben, so sollen so viele Doppelkopfrunden (bzw. Schafkopf mit jeweils vier Spielern) wie zwingend nötig gebildet werden, um tatsächlich alle Teilnehmer des "Schnuppernachmittags" zu beschäftigen.

Beispiel

Folgen genau zehn Spieler der Einladung des Skatclubs, so macht es keinen Sinn drei Skat-Runden (3*3=9) einzurichten, da dann eine Person übrig bleibt und unbeteiligt zusehen muss. Aber es können zwei Skat- und eine Doppelkopfrunde eröffnet werden (2*3+4=10) und alle Gäste können spielen.

Aufgabe

1. Schreiben Sie ein Programm, das Ihnen zu einer gegebenen Anzahl von Gästen anzeigt wie viele Skatrunden gebildet werden können und wie viele Doppelkopfrunden gebildet werden müssen.
2. Testen Sie das Programm mit folgenden Werten:
   a) -5 → Eingabefehler
   b) 2 → Keine Spielrunde
   c) 3 → Skat:1 Doppelkopf: 0
   d) 5 → Keine sinnvolle Kombination
   e) 8 → Skat: 0 Doppelkopf: 2
   f) 27 → Skat: 9 Doppelkopf: 0
   g) 124 → Skat: 40 Doppelkopf: 1
   h) 524 → Skat: 172 Doppelkopf: 2

Hinweis

Überlegen Sie wie viele Doppelkopfrunden maximal gebildet werden müssen, bevor man die verbleibende Gästezahl auf jeden Fall wieder durch drei teilen kann. Gerade bei sehr wenigen Gästen sind einige Sonderfälle zu bedenken. Überlegen Sie daher vorab, welche Ergebnisse das Programm bei 0 bis 10 Gästen jeweils liefern muss. Negative Zahlen sind als Eingabefehler zu kennzeichnen.

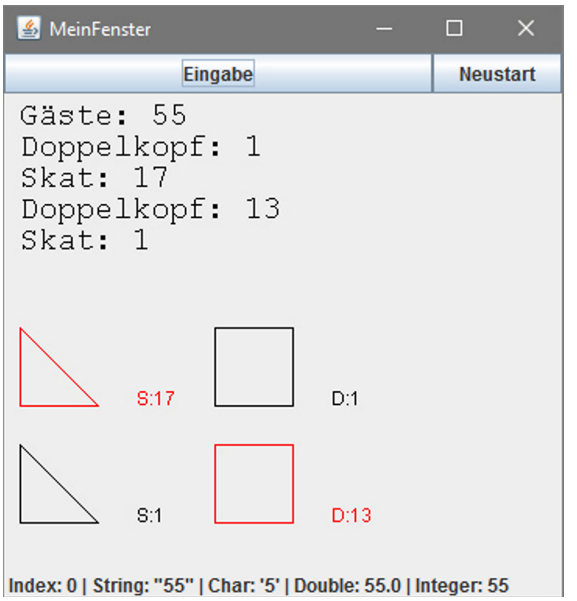
Zusatzaufgabe

Erweitern Sie das Programm so, dass es auch für einen Doppelkopfclub sinnvoll einsetzbar ist. Neben der skatoptimierten Anzeige soll nun auch ein Vorschlag ermittelt werden, bei dem so viele Doppelkopfrunden wie möglich gebildet werden und nur dann Skatrunden zustande kommen, wenn sonst Gäste übrig bleiben.

Wenn Sie die grafische **_umgebung** nutzen möchten, so führen Sie diese Aufgabe in der Funktion **meinProgramm()** der Klasse **MeinProgramm** aus und ergänzen Sie die folgende Funktionalität.

Aufgabe

Nutzen Sie die grafische **_umgebung**, um die Anzeige zu verbessern.
Deuten Sie die Skatrunden durch ein Dreieck (drei Spieler) und die Doppelkopfrunden durch ein Viereck (vier Spieler) an.
Je nachdem, ob Skat- oder Doppelkopfrunden im Vordergrund stehen, ist die jeweilige Anzeige rot hervorzuheben.



Hinweis

Es gibt in Java keinen separaten Befehl zum Zeichnen von Dreiecken. Dreiecke müssen entweder über den Befehl **drawPolygon()** oder aber aus drei Einzellinien **drawLine()** zusammengesetzt werden.
Der Befehl **drawString()** dient dem freien Positionieren eines Textes (wie z. B. "**S:17**") auf dem Fenster.

22 Übungen und Aufgaben

22.2.11 Tagesausflug

Bezug / Voraussetzung	
Kapitel	2.3.1 Bedingungen

Der Oberstufenkoordinator eines Gymnasiums möchte den 99 Schülern der Jahrgangsstufe 11 einen freiwilligen Tagesausflug zur Hannover-Messe anbieten. Das örtliche Reiseunternehmen kann dazu Busse mit 30, 40 oder 50 Sitzplätzen bereitstellen. Da allerdings die Busse pauschal bezahlt werden müssen, unabhängig davon wie viele Sitzplätze nun tatsächlich belegt sind, entschließt sich der Organisator die Fahrt nur dann stattfinden zu lassen, wenn die teilnehmenden Schüler so auf die Busse verteilt werden können, dass in einem 30er Bus maximal drei, in einem 40er Bus maximal vier und in einem 50er Bus maximal fünf Sitzplätze frei bleiben. Bei noch mehr freien Sitzplätzen würde der Fahrpreis je Schüler zu hoch.

Beispiele

Nehmen ALLE 99 Schüler der Jahrgangsstufe an der Fahrt teil, werden zwei 50er Busse benötigt und ein Sitzplatz bleibt frei. Melden sich hingegen nur 53 Schüler zur Tagesfahrt nach Hannover an, so kann die Fahrt leider nicht stattfinden, da zwei 30er Busse benötigt würden, in denen insgesamt maximal sechs Sitzplätze nach den Vorgaben frei bleiben dürften. Das bedeutet aber, dass noch genau eine Anmeldung fehlt, um den Grenzwert von 54 zu erreichen.

Aufgabe

Schreiben Sie ein Programm, das bei gegebener Teilnehmerzahl eine Aussage darüber trifft, ob unter den obigen Bedingungen die Tagesfahrt stattfinden kann oder nicht. Das Programm bekommt die Teilnehmerzahl als Eingabe. Diese Eingabe ist zunächst auf Zulässigkeit zu überprüfen. Da in der Jahrgangsstufe genau 99 Schüler sind, sind Eingaben > 99 definitiv falsch. Auch Eingaben < 21 sollen direkt mit dem Fehlertext "Falscher Wert" quittiert werden. Ist die eingegebene Teilnehmerzahl > 20 und < 100 soll der Text "Die Fahrt kann stattfinden." bzw. "Die Fahrt kann NICHT stattfinden!“ auf dem Bildschirm angezeigt werden.

Hinweise

Die Anzahl der Vergleiche lässt sich durch einen Trick drastisch verringern. Zerlegen Sie die eingegebene Teilnehmerzahl in Zehnerstelle und Einerstelle. Versuchen Sie nun aus der Summe dieser beiden Zahlen einen entsprechenden Zusammenhang abzuleiten.

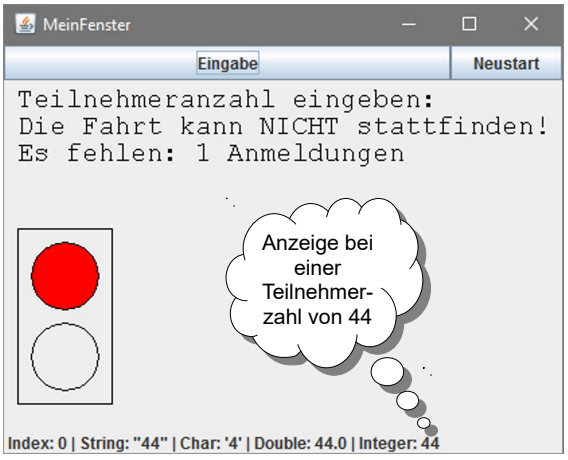
Zusatzaufgabe

Wenn die Fahrt stattfinden kann, soll zusätzlich die Anzahl freier Sitzplätze angezeigt werden. Kann die Fahrt hingegen nicht stattfinden, lassen Sie sich bitte anzeigen, wie viele Schüler sich noch zusätzlich anmelden müssten, damit die Fahrt wieder stattfinden kann.

Wenn Sie die grafische **_umgebung** nutzen möchten, so führen Sie diese Aufgabe in der Funktion **meinProgramm()** der Klasse **MeinProgramm** aus und ergänzen Sie die folgende Funktionalität.

Aufgabe

Nutzen Sie die grafische **_umgebung**, um die Anzeige zu verbessern. Eine zusätzliche Zweifarbenampel soll GRÜN leuchten, wenn die Fahrt zustande kommt und ROT anzeigen, wenn die Fahrt abgesagt werden muss.



Hinweis

Mit den folgenden Befehlen lässt sich in der grafischen **_umgebung** eine Ampel darstellen bei der beide Signalfarben aufleuchten.

Kopiervorlage

```
int x = 10;  // x-Position
int y = 150; // y-Position
int b = 50;  // Breite
int a = 10;  // Abstand

g.setColor(Color.RED);    // ROT leuchtet
g.fillOval(x+a, y+a, b, b);

g.setColor(Color.GREEN);  // GRÜN leuchtet
g.fillOval(x+a, y+b+a+a, b, b);

g.setColor(Color.BLACK);  // Ampelrahmen
g.drawRect(x, y, b+a+a, (b+a)*2+a);
g.drawOval(x+a, y+a, b, b);
g.drawOval(x+a, y+b+a+a, b, b);
```

22 Übungen und Aufgaben

Mehrfachauswahl

Variable			
Wert 1	Wert 2	Wert n	sonst
Block 1	Block 2	Block n	Block sonst

Wenn der Wert der Variable dem Vergleichswert **Wert 1** entspricht[1], werden die Anweisungen des **Blocks 1** ausgeführt. Analog werden die Anweisungen der jeweiligen Blöcke nur dann durchlaufen, wenn der zugehörige Vergleichswert mit dem Wert der Variablen übereinstimmt. Für den Fall, dass es keine Übereinstimmung zwischen dem Variablenwert und den Vergleichswerten gibt, kann man optional einen Alternativblock (**sonst**) angeben.

22.2.12 Buchstabiertafeln

Bezug / Voraussetzung	
Kapitel	2.3.2 Verzweigungen
Befehle	scanner.next().charAt(0)

Es gibt verschiedene Buchstabiertafeln, die zum Buchstabieren einzelner Wörter verwendet werden können.

Buchstabiertafeln

Buchstabe	Deutsch	Nato	International
A	Anton	Alfa	Amsterdam
E	Emil	Echo	Edison
I	Ida	Inda	Italia
O	Otto	Oscar	Oslo
U	Ulrich	Uniform	Upsala

Beispiel

In Deutschland ist es beispielsweise üblich das Wort "Ei" mit den Namen "**E**mil" und "**I**da" zu buchstabieren.

Aufgabe

Schreiben Sie ein Programm, das ein einzelnes Zeichen von der Tastatur einliest. Handelt es sich bei dem Zeichen um einen Vokal (a, e, i, o, u bzw. A, E, I, O, U) soll der entsprechende Name der deutschen Buchstabiertafel angezeigt werden. Bei allen anderen Buchstaben zeigt das Programm "Kein Vokal" als Meldung an.
ACHTUNG: Das Programm soll für Klein- UND Großbuchstaben funktionieren und komplett OHNE **if**-Bedingungen auskommen.

Hinweis

Ein einzelnes Zeichen kann man von der Tastatur mit dem Befehl **scanner.next().charAt(0)** einlesen. Verdeutlichen Sie sich vorab nochmal, wie eine **oder**-Verknüpfung mit der **switch**-Anweisung realisiert wird. Welche Bedeutung hat der Befehl **break**?

Zusatzaufgabe

Erweitern Sie das Programm so, dass der Anwender zunächst über eine zusätzliche Eingabe eine der drei Buchstabiertafeln auswählen kann:

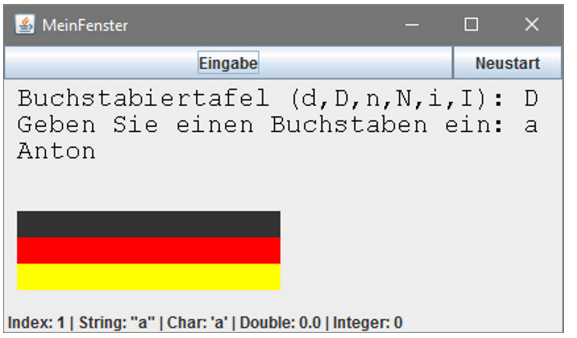
1. **d** oder **D** → Deutsch (Standardwert!)
2. **n** oder **N** → Nato
3. **i** oder **I** → International

Anschließend werden wieder bei der Eingabe von Vokalen die passenden Begriffe ausgegeben. Auch das erweiterte Programm soll komplett OHNE **if**-Bedingungen auskommen und sowohl für Klein- als auch für Großbuchstaben funktionieren.

Wenn Sie die grafische **_umgebung** nutzen möchten, so führen Sie diese Aufgabe in der Funktion **meinProgramm()** der Klasse **MeinProgramm** aus und ergänzen Sie die folgende Funktionalität.

Aufgabe

Nutzen Sie die grafische **_umgebung**, um die Anzeige zu verbessern. Zeigen Sie immer dann, wenn die deutsche Buchstabiertafel ausgewählt wurde, zusätzlich eine stilisierte Deutschlandfahne auf dem Bildschirm an.



Zusatzaufgabe

Lassen Sie sich auch für die anderen Buchstabiertafeln (Nato und International) sprechende grafische Darstellungen einfallen, beispielsweise ein stilisiertes Nato-Symbol auf blauem Grund.
Wird nun die Buchstabiertafel "Nato" gewählt, soll diese Darstellung anstelle der Deutschlandfahne angezeigt werden.

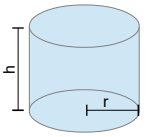
1 Es können nicht nur Einzelwerte, sondern auch Wertebereiche angegeben werden.

22 Übungen und Aufgaben

22.2.13 Zylinderberechnung (Teil 1)

Bezug / Voraussetzung	
Kapitel	2.3.2 Verzweigungen
Befehle	scanner.next().charAt(0) scanner.nextDouble() Math.PI

Umfang: $U = 2 * \pi * r$
Grundfläche: $G = \pi * r^2$
Mantel: $M = 2 * \pi * r * h$
Oberfläche: $O = 2 * \pi * r * (r + h)$
Volumen: $V = \pi * r^2 * h$



Mit diesen Formeln lassen sich der Umfang, die Grundfläche, der Mantel, die Oberfläche und das Volumen eines Zylinders berechnen.

Aufgabe

Schreiben Sie ein Programm bei dem der Anwender zunächst über die Eingabe eines Buchstabens U, G, M, O oder V bestimmen kann, welche Berechnung er ausführen möchte. Anschließend sind noch die Höhe h und der Radius r anzugeben.

1. Das Programm soll nur die ausgewählte Berechnung ausführen.
2. Die Auswahl selbst soll über Klein- und Großbuchstaben möglich sein.
3. Die Berechnung soll mit Gleitkommazahlen und hoher Genauigkeit (**double**) erfolgen.
4. Die Werte für die Höhe **h** und den Radius **r** müssen größer oder gleich **0** sein!
5. Sobald **h / r >= 10000** ist die Warnung "Ungünstige Werte" auszugeben, die Berechnung aber trotzdem auszuführen.
6. Die Verzweigung zwischen den verschiedenen Berechnungen kann man mit der **switch**-Anweisung realisieren.

Hinweise

Die Konstante **PI** (π) ist in der Klasse **Math** hinterlegt und kann über **Math.PI** angesprochen werden. Der Befehl **scanner.next().charAt(0)** ermöglicht das Einlesen eines einzelnen Zeichens von der Tastatur.

Zusatzaufgabe

Ermöglichen Sie dem Anwender auch die Einheit anzugeben, in der er die Angaben für die Höhe **h** und den Radius **r** machen möchte.

- mm → Millimeter
- cm → Zentimeter
- dm → Dezimeter
- m → Meter
- km → Kilometer

Die Angabe zur Einheit soll in einer Variable vom Typ **String** gespeichert werden und kann über den Befehl

```
String einheit = scanner.next();
```

z. B. in die Variable **einheit** eingelesen werden. Stellen Sie sicher, dass tatsächlich nur zulässige Einheiten (mm, cm, dm, m, km) angebbar sind. Andernfalls ist die Berechnung mit der Fehlermeldung "Eingabefehler" abzubrechen. Auch bei der Ergebnisausgabe soll die richtige Einheit angezeigt werden (Einheit **m** wird beim Volumen zu m^3 bzw. **m^3**).

Hinweis

Zwei Strings vergleicht man nicht mit dem Operator **==**, sondern mit der Methode **equals()** auf Gleichheit. So testet beispielsweise der folgende Befehl

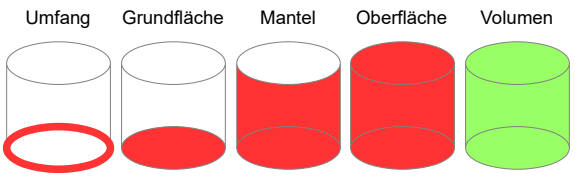
```
einheit.equals("dm")
```

ob sich in der String-Variablen **eingabe** tatsächlich der Text „**dm**“ befindet.

Wenn Sie die grafische **_umgebung** nutzen möchten, so führen Sie diese Aufgabe in der Funktion **meinProgramm()** der Klasse **MeinProgramm** aus und ergänzen Sie die folgende Funktionalität.

Aufgabe

Nutzen Sie die grafische Umgebung, um zusätzlich anzuzeigen, welche Bereiche des Zylinders gerade berechnet werden. Heben Sie die Bereiche farblich hervor. Beispielsweise wie folgt:



Hinweis

Mit der folgenden Befehlsfolge kann ein einfacher weißer Zylinder ohne besondere farbige Hervorhebungen in der grafischen **_umgebung** erzeugt werden.

Kopiervorlage

```
int x = 50;      // x-Position
int y = 150;     // y-Position
int b = 100;     // Breite
int h = 25;      // Höheneinheit

g.setColor(Color.WHITE);
g.fillRect(x, y+h, b, 4*h);
g.setColor(Color.BLACK);
g.drawRect(x, y+h, b, 4*h);
g.setColor(Color.WHITE);
g.fillOval(x, y, b, 2*h);
g.fillOval(x, y+4*h, b, 2*h);
g.setColor(Color.BLACK);
g.drawOval(x, y, b, 2*h);
g.drawOval(x, y+4*h, b, 2*h);
g.drawString("Zylinder", x+h, y+h);
```