

Übungen zu Kapitel 9

9.1 Nullsummen-Spiel:

Zwei Spieler *A* und *B* treten gegeneinander an. Jeder hat das gleiche Anfangskapital. Es wird anfangs festgelegt, wie groß die Wahrscheinlichkeit p ($0 < p < 1$) ist, dass Spieler *A* gewinnt. Es wird immer wieder eine Zufallszahl z zwischen 0 und 1 gezogen. Wenn $z \leq p$ ist, dann hat *A* gewonnen, andernfalls *B*. Der Gewinner bekommt eine Geldeinheit, der Verlierer muss eine Einheit abgeben. Die Zahl der Gewinnzüge vermerkt, wie oft ein Spieler gewonnen hat. Das Spiel läuft solange, bis ein Spieler ruiniert ist (Schleife!).

Es sollen zwei Klassen angelegt werden:

Die *nichtausführbare* Klasse Spieler:

Attribute :

```
public int geld
public int zahlGewinnzuege.
```

Methoden :

```
public Spieler( int g ): Der Konstruktor setzt das Attribut geld auf den
Übergabewert und das Attribut zahlGewinnzuege auf 0.
```

```
public boolean kannSpielen(): Gibt true zurück, wenn geld noch > 0 ist,
andernfalls false.
```

Spieler
public int geld
public int zahlGewinnzuege
public Spieler(int g)
public boolean kannSpielen()

Die *ausführbare* Klasse Spiel:

In ihrer `main()`-Methode soll der Benutzer die Höhe des Anfangskapitals und die Wahrscheinlichkeit p eingeben, dass *A* gewinnt.

Es werden die zwei Spieler-Objekte **A** und **B** erzeugt.

Es werden Spiele durchgeführt: Solange noch beide Objekte weiterspielen können, wird mit `Math.random()` eine `double`-Zufallszahl zwischen 0 und 1 gezogen und geprüft, wer gewonnen hat. Der Sieger erhält eine Geldeinheit, beim der Verlierer wird eine Geldeinheit abgezogen. Beim Sieger wird außerdem die Zahl der Gewinnzüge hochgezählt.

Wenn einer der Spieler ruiniert ist, wird ausgegeben, wer gewonnen hat. Es wird das Verhältnis der Zahl der Gewinnzüge von *A* zur Zahl der Gewinnzüge von *B* berechnet und auf zwei Kommastellen gerundet ausgegeben. Der theoretische Wert lautet: $p / (1-p)$.

9.2 Bruchrechner: Bruchrechnen ist objektorientiert zu implementieren. Es sollen zwei Klassen angelegt werden:

Die *nichtausführbare* Klasse Bruch:

Attribute :

```
public int zaehler
public int nenner
```

Methoden :

```
public Bruch(): Der parameterlose Konstruktor soll den parametertragenden Konstruktor mit
den Werten 1 und 1 aufrufen.
```

```
public Bruch(int z, int n): Der parametrisierte Konstruktor soll die Attribute der Klasse
Bruch mit den übergebenen Werten belegen.
```

```
public void show(): Soll den Wert eines Bruchs ausgeben in der Form zaehler/nenner.
```

```
public voidkehrWert(): Soll den Kehrwert bilden durch Vertauschen von Zähler und Nenner
innerhalb eines Bruchobjekts.
```

Bruch
public int zaehler
public int nenner
public Bruch() public Bruch(int z, int n)
public void show()
public voidkehrWert()
public void negiere()
public void kuerze()

`public void negiere():` Soll den Bruch negieren, zB durch `zaehler = - zaehler`.

`public void kuerze():` Soll den Bruch kürzen, indem die Attribute `zaehler` und `nenner` durch ihren größten gemeinsamen Teiler (ggT) geteilt werden. Der ggT zweier ganzer Zahlen kann durch folgende Methode ermittelt werden:

```
public static int ggT( int a, int b ) {
    if( b < 0 ) b = -b;
    int r = a % b;
    while( r != 0 ){
        a = b;    b = r;    r = a % b;
    }
    return b;
}
```

Die *ausführbare* Klasse `Bruchrechner` mit den Methoden:

`public static boolean istGleich(Bruch bx, Bruch by):` Es soll festgestellt werden, ob die beiden Bruchobjekte identisch sind, d.h. gleiche Werte für die Attribute `zaehler` und `nenner` haben. Ist dies der Fall, soll `true` zurückgegeben werden, andernfalls `false`. Vor dem Vergleich müssen die Objekte gekürzt werden.

`public static Bruch addiere(Bruch bx , Bruch by):` Es werden zwei Brüche (übergebene Bruch-Objekte) gemäß den Regeln der Bruchrechnung addiert und das Resultat als neu erzeugtes Bruch-Objekt an den Aufrufer zurückgeben. Vor der Rückgabe soll gekürzt werden.

In der `main()`-Methode sollen Objekte der Klasse `Bruch` angelegt werden.

Testen Sie mit den Bruch-Objekten die in `Bruch` und `Bruchrechner` enthaltenen Methoden.

Wie müsste die Klasse `Bruch` umgeschrieben werden, falls diese nur immutable Bruch-Objekte zulassen würde?

9.3 Währungsrechner: Schreiben Sie eine Klasse `Waehrungsrechner`, mit deren Objekten zwischen je zwei Währungen hin- und hergerechnet werden kann. Dazu besitzt die Klasse ein Attribut `public double kurs`, das den Umrechnungskurs enthält und im Konstruktor der Klasse initialisiert wird. Zum Umrechnen in der einen und anderen Richtung dienen die beiden Methoden:

```
public double hin( double betrag )
public double her( double betrag )
```

Der umzurechnende Betrag wird als Parameter übergeben und durch Multiplikation bzw. Division mit dem Kurswert umgerechnet und zurückgeliefert.

In der `main()`-Methode der ausführbaren Klasse `Waehrungen` soll der Benutzer solange wie gewünscht (Schleife!) nach dem Kurswert gefragt und ein entsprechendes `Waehrungsrechner`-Objekt instanziiert werden. Der Benutzer soll vorgeben können, in welcher Richtung er umrechnen möchte. Der eingegebene Betrag wird mittels der Methoden des Objekts umgerechnet und ausgegeben. Orientieren Sie sich hinsichtlich der Benutzerführung an der Klasse `Bank` im Buch.