5. Übungsblatt:
Programmierpraktikum I (WS 2002/03)

Abgabe: 13. Januar 2002

Bem. 1: Für jede Aufgabe sind die angegebenen Dateien mit den jeweils angegebenen Namen abzugeben (Groß-/Kleinschreibung beachten!).

Bem. 2: Es dürfen nur Bibliotheken verwendet werden, die der Ein-/Ausgabe dienen oder die explizit angegeben sind.

Java: Klasse *Rational*

(`Rational.java`, 9 Punkte)

Schreibe eine Klasse **Rational**, die für die Arithmetik rationaler Zahlen a/b verwendet werden kann. Dazu gehören mindestens die folgenden Funktionen:

- Drei Konstruktoren:
  - ohne Parameter (setzt $a/b = 0/1$),
  - mit einem Parameter (long) x (setzt $a/b = x/1$),
  - mit zwei Parametern (long) x, y (setzt $a/b = x/y$).
- Funktionen `long zaehler()` und `long nenner()`, die Zähler bzw. Nenner zurückliefern.
- Eine Funktion `String alsString()`, die a/b als String zurückgibt.
- Eine interne Hilfsfunktion `long ggT()`, die den größten gemeinsamen Teiler $ggT(a, b)$ berechnet.[1]
- Eine interne Hilfsfunktion `void kuerzen()`, die Zähler und Nenner durch den $ggT(a, b)$ teilt.
- Eine interne Hilfsfunktion `void normieren()`, die den Nenner positiv macht, indem sie ggf. Zähler und Nenner mit (-1) multipliziert.
- Eine Funktion `Rational betrag()`, die den Betrag der Zahl zurückgibt.
- Eine Funktion `Rational inverses()`, die das Inverse der Zahl (also bei a/b den Wert b/a) zurückgibt.

[1] Ein effizienter Algorithmus zur Berechnung des ggT wird in der Übung vorgestellt; wer das verpasst hat, der sucht in Bibliothek oder Internet unter dem Schlagwort "Euklidischer Algorithmus".

- Eine Funktion `boolean gleich(Rational)`, die vergleicht, ob die rationale Zahl gleich dem Parameter ist.
- Eine Funktion `boolean kleiner(Rational)`, die feststellt, ob die rationale Zahl kleiner als der Parameter ist.
- Funktionen
  - `Rational plus(Rational)`,
  - `Rational minus(Rational)`,
  - `Rational mal(Rational)` und
  - `Rational durch(Rational)`,

  die die vier Grundrechenarten zur Verfügung stellen.

Beachte bei allen Funktionen, dass keine ungültigen Zahlen wie $a/0$ entstehen dürfen. Tritt dieser Fall ein, so soll eine `ArithmeticException` ausgeworfen werden.
Außerdem sollen korrekte rationale Zahlen am Ende jedes Funktionsaufrufes (insbesondere nach dem Neuanlegen) in gekürzter und normierter Form sein. Umgekehrt darf man dann davon ausgehen, dass rationale Zahlen als Funktionsinput in gekürzter und normierter Form sind.

Java: Klasse *Complex* ableiten

(`Complex.java`, 6 Punkte)

In der Vorlesung wurde die Klasse *Complex* vorgestellt. Modifiziere diese Klasse so, dass sie von der abstrakten Klasse *Number* (aus *java.lang*) abgeleitet ist und die entsprechenden Methoden zur Typumwandlung bereitstellt.

Bem.: Zur Umwandlung von komplexen in nicht-komplexe Zahlen genügt es, den Real-Teil zu betrachten und entsprechend zu konvertieren.

Java: Black-Jack-Programm

(eine Datei pro Klasse, 20 Punkte)

Gegeben seien die folgenden, stark vereinfachten Spielregeln für das Kartenspiel "Black Jack" (*17 und 4*), bei dem 2-4 Spieler gegeneinander (statt wie sonst üblich gegen die Bank) spielen:

- Gespielt wird mit einem unendlich großen Vorrat an Spielkarten, aus dem die Karten mit den Werten $2, 3, \ldots, 10$, *Bube, Dame, König, As* stets mit gleicher Wahrscheinlichkeit gezogen werden. Die Kartenfarbe spielt dabei keine Rolle.
- Das As zählt 11 Punkte, die Bilder zählen 10 Punkte und die Zahlen den normalen Wert.

- Ziel des Spieles ist es, Karten so zu ziehen, dass man dabei so nah wie möglich an die 21 heranzukommt. Die Spieler ziehen abwechselnd (verdeckt) Karten vom Stapel. Ein Spieler, der glaubt, eine gute Punktzahl erreicht zu haben, hört auf zu ziehen. Hört er zu spät auf und hat mehr als 21 Punkte auf der Hand, so scheidet er aus.

- Haben alle Spieler aufgehört, so gewinnt derjenige Spieler, der am nächsten an die 21 herangekommen ist. Haben alle Spieler die 21 überschritten, so gibt es keinen Gewinner. Haben mehrere Spieler gleich viele Punkte, so teilen sie sich den Sieg.

Dieses Spiel soll nun implementiert werden. Das Hauptprogramm, das den Spielfluss steuert, soll von der Klasse `Blackjack` implementiert werden. Es erfragt die Zahl der Mitspieler, regelt den Spielverlauf und gibt das Ergebnis bekannt. Dabei sollen die folgenden Hilfsklassen verwendet werden:

- Die Klasse `Karte` implementiert die Spielkarten. Ein Objekt vom Typ Karte hat immer einen zufälligen Wert zwischen 2 und *As*. Dazu kann wieder der auf Blatt 4 vorgestellt Zufallsgenerator verwendet werden.

- Die Klasse `Spieler` implementiert den jeweiligen Spieler. Sie regelt die Kommunikation mit dem "realen" Spieler an der Tastatur und speichert seinen aktuellen Spielstand.

Bem.: Die Ein-/Ausgabe spielt bei diesem Programm eine große Rolle. Zum einen muss die Anwendung des Programmes gut verständlich sein (die Spielregeln dürfen aber als bekannt vorausgesetzt werden). Auch sollen unsinnige Eingaben der Spieler abgefangen werden. Alle erforderlichen Eingaben sollten sich allerdings mit der Methode `liesInt()` vom 4. Übungsblatt umsetzen lassen.