9. Übungsblatt: Programmierpraktikum I (WS 2003/04)

Abgabe: 9. Januar 2004 (**Freitag!**)

Java: Geometrie 1

(`Aufgabe1.java, Punkt.java`, 9 Punkte)

Schreibe eine Klasse `Punkt`, die einen Punkt im Vektorraum $\mathbb{R}^3$ repräsentiert. Die Klasse soll wie folgt aufgebaut sein:

- Der eigentliche Punkt wird durch drei Koordinaten vom Typ `double` dargestellt, wobei darauf zu achten ist, dass diese Koordinaten nur durch die eigenen Methoden des Objektes geändert werden können.
- Es gibt drei Konstruktoren. Einer davon wird ohne Parameter aufgerufen, der zweite erhält drei `int`-Paramter, der dritte drei `double`-Parameter.
- Die Methoden `neuesX`, `neuesY` und `neuesZ` sind so zu überladen, dass sie sowohl einen `int`- wie auch einen `double`-Parameter akzeptieren und die jeweilige Koordinate des Punktes auf den übergebenen Wert setzen.
- Die Methoden `gibX`, `gibY`, `gibZ` geben die jeweilige Koordinate als `double`-Wert zurück.
- Die Methode `toString` gibt den Punkt als String zurück in der Form `(x,y,z)`.
- Die Methode `abstandZu` erhält als Parameter einen weiteren Punkt, berechnet den Euklidischen Abstand zwischen den beiden Punkten und gibt diesen als `double`-Wert zurück.[1]

Weiterhin ist eine Klasse `Aufgabe1` zu erstellen, die mehrere Objekte vom Typ `Punkt` anlegt und alle Methoden testet.

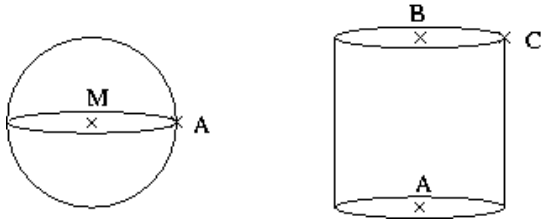
Java: Geometrie 2

(`Aufgabe2.java, Kugel.java, Quadrat.java`, 6 Punkte)

Schreibe Klassen für die folgenden geometrischen Objekte:

- Eine Klasse `Kugel`, die eine Kugel im Vektorraum $\mathbb{R}^3$ durch zwei Objekte vom Typ `Punkt` repräsentiert, wie auf Seite 2 dargestellt.

[1] Zur Lösung dieses Problems empfiehlt sich ein Blick in eine Formelsammlung und in Javas `Math`-Bibliothek.



- Eine Klasse `Zylinder`, die einen Zylinder im Vektorraum $\mathbb{R}^3$ durch drei Objekte vom Typ `Punkt` repräsentiert, wie auf Seite 2 dargestellt.

Jede dieser Klassen soll über mindestens zwei Konstruktoren verfügen (einen ohne Parameter und einen mit `Punkt`-Objekten als Parameter). Dabei soll der Konstruktor der Klasse `Zylinder` auch überprüfen, ob die Eingabeparameter konsistent sind.
Außerdem erhält jede Klasse die Methoden `flaeche` und `volumen`, die die Oberfläche bzw. den Rauminhalt des Objektes als `double`-Wert zurückgeben, sowie die Methode `toString`, die einen String zurückgibt, der die Punkte des Objektes beschreibt (z.B. in der Form `Mittelpunkt (3.0,1.24), Sphaerenpunkt (4.31,1.1)`).
Zudem soll eine Klasse `Aufgabe2` geschrieben werden, die `Kugel`- und `Zylinder`-Objekte anlegt und alle Methoden testet.

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 herankommt. 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 (die Spieler spielen abwechselnd am gleichen Rechner) 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 8 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. 1: 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. Die genaue Gestaltung der Ein-/Ausgabe ist dir überlassen; es ist jedoch grundsätzlich möglich, alle erforderlichen Eingaben mit der Methode `liesInt()` vom 7. Übungsblatt zu erledigen.

Die harte Nuss II

(`hartenuss2.tgz`, 20 Punkte)

Vorbemerkungen:

- *Die folgende Aufgabe ist als "Weihnachtsprojekt" gedacht für diejenigen, die die Feiertage nutzen wollen, um einmal ein spannenderes Programmierprojekt umzusetzen.*
- *Die Punkte zählen im Übungswettbewerb nur bei Gleichstand nach normalen Punkten.*
- *Der Umfang, in dem ihr diese Aufgabe bearbeitet, ist bewusst euch überlassen - von kleinen Beiträgen bis zu einem vollständigen Programmpaket ist alles erlaubt!*
- *Die Dateien sind einem gezippten Archiv abzugeben (Linux-Befehle* `tar` *und* `gzip`*). Die Namen der einzelnen Dateien können frei gewählt werden, eine schriftliche Abgabe ist nicht erforderlich.*
- *Abgabetermin für diese Aufgabe ist der 19. Januar 2004.*

Aufgabenstellung: Das in Aufgabe 3 beschriebene Spiel stellt natrlich nur eine Rohfassung dessen dar, was man aus der Aufgabenstellung "Implementiere ein BlackJack-Spiel" machen könnte. Wer das Spiel gerne erweitern möchte, darf das gerne tun, z.B. um

- exakte BlackJack-Spielregeln,
- Verwendung von Runden,
- Verwendung von Einsätzen,
- computergesteuerte Gegenspieler,
- graphische Benutzeroberfläche oder
- Netzwerkfähigkeit.

Alle Erweiterungen (z.B. erweiterte Spielregeln) sind unbedingt in der Abgabe (z.B. als README-Datei) zu dokumentieren.