Programmierpraktikum

Wintersemester 2021/22

Prof. Dr. rer. nat. habil. Petra Hofstedt
Ilja Becker M. Sc., Sven Löffler M. Sc.
Sonja Breuß B. Sc., Carlo Bückert B. Sc., Endris Wenzke

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Übungsblatt 8

Abgabedatum: 12.12.2021

Die Abgabe Ihrer Lösungen erfolgt vor Ablauf der Abgabefrist digital über die Moodle-Plattform. Erstellen Sie dazu ein PDF-Dokument, das die Lösungen Ihrer schriftlichen Aufgaben enthält. Laden Sie dieses PDF-Dokument und den erarbeiteten Java-Code (.java-Dateien) mit den in den Aufgaben vorgegebenen Namen bei Moodle hoch. Bitte laden Sie die Dateien einzeln hoch, Dateiarchive (z.B. .zip-Dateien) werden nicht akzeptiert.

Sie können maximal **(8 +2 Punkte)** mit diesem Übungsblatt erreichen.

Aufgabe 1 (Binomialkoeffizient)

2,5 Punkte

Für gegebene natürliche Zahlen n und k soll der Binomialkoeffizient $\binom{n}{k}$ mittels unterschiedlicher Verfahren ermittelt werden. Für jedes Berechnungsverfahren soll eine entsprechende Methode implementiert werden, die die Zahlen n und k als Argumente vom Typ `int` erhält und das Resultat als Wert vom Typ `long` liefert.

1. Legen Sie eine neue Klasse `Binomial.java` an. Schreiben Sie eine Methode `binomialFakultat`, die $\binom{n}{k}$ nach der Formel $\frac{n!}{k!*(n-k)!}$ berechnet. Verwenden Sie für die Berechnung der Fakultäten eine selbstgeschriebene rekursive Funktion `private static fak`.

   (0,5 Pkt.)

2. Schreiben Sie eine Methode `binomialProduktQuotient`, die zunächst die Produkte n * (n - 1) * (n - 2) * . . . * (n - k + 1) und 1 * 2 * 3 * . . . * k berechnet und schließlich das erstgenannte Produkt durch das letztgenannte dividiert, um $\binom{n}{k}$ zu erhalten.

   (0,5 Pkt.)

3. Schreiben Sie eine Methode `binomialAlternierend`, die $\binom{n}{k}$ berechnet, indem sie den Ausdruck $\frac{n}{1} * \frac{n-1}{2} * \ldots * \frac{n-k+1}{k}$ von links nach rechts auswertet. Überlegen Sie sich, ob Sie für die Berechnung Gleitkommazahlen benötigen oder ob Sie ausschließlich mit ganzen Zahlen rechnen können.

   (0,5 Pkt.)

4. Schreiben Sie eine Methode `binomialRekursiv`, die $\binom{n}{k}$ rekursiv gemäß der folgenden Definition ausrechnet:

$$\begin{aligned}\binom{n}{0} &= 1\\ \binom{0}{k+1} &= 0\\ \binom{n+1}{k+1} &= \binom{n}{k} + \binom{n}{k+1}\end{aligned}$$

   (0,5 Pkt.)

5. Implementieren Sie eine `main`-Methode, die zwei natürliche Zahlen n und k einließt, $\binom{n}{k}$ mit jeder der oben beschriebenen Methoden berechnet und die entsprechenden Resultate ausgibt. Geben Sie Vor- und Nachteile der verschiedenen Methoden an.

(0,5 Pkt.)

Aufgabe 2 (Potenzen rekursiv) 1,5 Punkte

Schreiben Sie eine Java-Klasse `Potenzen.java` die eine Potenz a^b für zwei ganzzahlige positive Zahlen a und b nach folgendem Schema rekursiv berechnet:

Wenn b gleich 1 ist, dann gib a aus, ansonsten ermittle $a^{abgerundet(b/2)} * a^{aufgerundet(b/2)}$.

Erweitern Sie Ihr Programm anschließend so, dass keine Berechnungen doppelt ausgeführt werden. Verwenden Sie zum Beispiel ein Array um die Werte für $a^{abgerundet(b/2)}$ und $a^{abgerundet(b/2)}$ abzulegen. Überprüfen Sie vor jedem rekursiven Aufruf, ob dieser nötig ist oder der entsprechende Wert bereits aus dem Array ausgelesen werden kann.

Aufgabe 3 (Turtle-Grafik) 4 Punkte

Anmerkung: Für das richtige Bearbeiten der ersten 3 Aufgaben gibt es 2 Punkte und für Aufgabe 1.4 gibt es ebenfalls 2 Punkte.

Turtle-Grafik ist ein einfacher Ansatz zum Zeichnen linienbasierter Grafiken. Man stellt sich vor, dass sich eine Schildkröte (engl. turtle) über die Zeichenfläche bewegt und gegebenenfalls mit einem Stift den von ihr zurückgelegten Weg markiert. Die Schildkröte empfängt hierzu Befehle, die sie dann ausführt. Mögliche Befehle sind:

- „Setze den Stift auf die Zeichenfläche auf."
- „Nimm den Stift von der Zeichenfläche weg."
- „Bewege dich um eine bestimmte Strecke nach vorn."
- „Drehe dich um einen bestimmten Winkel gegen den Uhrzeigersinn."

Der Winkel kann auch negativ sein, dann dreht sich die Schildkröte im Uhrzeigersinn.

Offensichtlich bestimmt der aktuelle Zustand (engl. state) der Schildkröte - also ihre Position, ihre Ausrichtung und die Stellung des Stifts - den Effekt eines Befehls. Außerdem ändert ein Befehl im Allgemeinen den Zustand der Schildkröte.

1. Programmieren Sie eine Java-Klasse `TurtleState.java`, die den Zustand der Schildkröte speichert und Modifikationen des Zustands erlaubt. Führen Sie (`private`) Attribute `double x, double y` für die Position, `double angle` für die Ausrichtung und `boolean down` für die Stellung des Stifts ein. Fügen Sie für jedes Attribut Setter- und Getter-Methoden hinzu.
2. Programmieren Sie die Java-Klasse `Turtle.java`, welche die oben genannten Befehle empfängt und im Attribut `TurtleState state` entsprechende Zustandsänderungen durchführt. Gegebenenfalls zeichnet sie außerdem eine Linie in das grafische Ausgabefenster `LineFrame frame`. Vervollständigen Sie die Klasse des Weiteren um sinnvolle JavaDoc-Kommentare.
3. Programmieren Sie ein Testprogramm für ihre Turtle-Klasse (`Wuerfel.java`), das einen Würfel mittels der Turtle-Grafik im Schrägbild zeichnet.

4. Informieren Sie sich z. B. durch den folgenden Wikipediabeitrag zum Thema Kochkurve `https://de.wikipedia.org/wiki/Koch-Kurve`.

   Schreiben Sie ein Programm `Kochkurve.java` das unter Verwendung der Turtle-Grafik eine kochsche Schneeflocke zeichnet. Dazu soll vom Nutzer zu Beginn eingegeben werden können, welche Iterationsstufe gezeichnet werden soll. Auf Stufe Null soll dabei lediglich ein gleichseitiges Dreieck mit Kantenlänge 0,7 Fenstergrößen gezeichnet werden. Für die erste Iteration wird dann jeweils jede Strecke in drei gleich große Teile geteilt und die mittlere Kante durch zwei ebenso große Strecken, die einen 60° Winkel einschließen, ersetzt. Auf jeder weiteren Iterationsstufe wird jede Strecke wieder gedrittelt und die mittlere Strecke ersetzt.

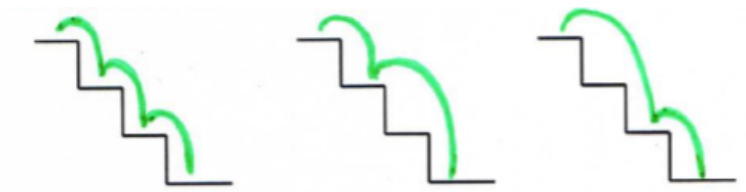
   Lassen Sie sich einen geeigneten rekursiven Ansatz zur Lösung einfallen. Beachten Sie beim Testen des Programmes, dass eine Iterationsstufe größer als zehn für den Computer sehr rechenaufwendig ist.

Aufgabe 4 (Zusatzaufgabe: Treppenproblem) (+2) Punkte

1. Eine Treppe kann man auf unterschiedliche Arten hinaufgehen. Gehen Sie davon aus, dass man immer die Möglichkeit hat eine Stufe oder aber zwei Stufen aufeinmal zu nehmen. Entwickeln Sie ein Programm `Treppen.java`, das als Eingabe die Höhe einer Treppe, also die Anzahl ihrer Stufen erhält und ausgibt, wie viele verschiedene Möglichkeiten es gibt die Treppe hinauf zu gehen.

   (1 Pkt.)

   Eine Treppe mit drei Stufen hätte z. B. die folgenden drei Möglichkeiten.



2. Schreiben Sie ein weiteres Programm `Treppen2.java` welches als Eingabe zusätzlich die maximale Schrittweite, also die Anzahl der Stufen die Maximal mit einmal überschritten werden können, einliest und die Anzahl der unterschiedlichen Möglichkeiten für das Besteigen der Treppe ausgibt.

   (1 Pkt.)