Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Jens-Peter Redlich
M. Kabierski, Dr. W. Müller, L. Pfahlsberger, M. Sänger

Berlin, den 05.12.2022

Übungen zur Vorlesung
Grundlagen der Programmierung

Übungsblatt 4

Abgabe: Die Abgabe Ihrer Lösungen erfolgt über den Moodle-Kurs der Vorlesung (https://moodle.hu-berlin.de/enrol/index.php?id=114664). Nutzen Sie die dort im Abschnitt Übung angegebene **Aufgabe 4** und laden Sie insgesamt **drei Dateien hoch:** `Aufgabe1.java`, `ArmstrongNumbers.java`, `PythagorasTree.java`. Die Abgabe ist bis zum **02.01.2023** um **09:15 Uhr** möglich.

Hinweise:

- Achten Sie darauf, dass Ihre Java-Programmdateien mittels des UTF-8-Formats (ohne byte order marker (BOM)) kodiert sind und keinerlei Formatierungen enthalten.
- Verzichten Sie auf eigene Packages bei der Erstellung Ihrer Lösungen, d.h. **kein** `package`-Statement am Beginn Ihrer Java-Dateien.
- Quelltextkommentare können die Korrektoren beim Verständnis Ihrer Lösung (insbesondere bei inkorrekten Abgaben) unterstützen; sind aber nicht verpflichtend.
- Testen Sie Ihre Lösung bitte auf einem Rechner aus dem Informatik Computer-Pool z.B. gruenau6.informatik.hu-berlin.de (siehe auch Praktikumsordnung: Referenzplattform)

Aufgabe 1 (pass-by-value / pass-by-reference) — 6 Punkte

In den folgenden fünf Teilaufgaben (1a bis 1e) sollen Sie für jede gegebene Klasse entscheiden, welche der jeweils im Anschluss aufgelisteten Aussagen korrekt sind. Bei jeder Teilaufgabe sind mehrere Antwortmöglichkeiten richtig (Multiple Choice). Für jede richtig gewählte Aussage erhalten Sie einen halben Punkt und für jede falsch gewählte Aussage wird Ihnen ein halber Punkt abgezogen. Je Teilaufgabe können Sie nicht weniger als 0 Punkte bekommen. Nutzen Sie zur Angabe Ihrer gewählten Antwortmöglichkeiten bitte das Programm `Aufgabe1.java`, welches via Moodle bereitgestellt wird. Diese enthält fünf Methoden für jede Teilaufgabe. Jede Methode wiederum enthält sechs Variablen `a`, `b`, `c`, `d`, `e`, `f` vom Typ `boolean`, welche initial auf `true` gesetzt wurden. Ändern Sie jeweils die Werte dieser sechs Variablen, sodass diese mit Ihren gewählten Antwortmöglichkeiten übereinstimmen. Testen und überprüfen Sie Ihre Abgabe mit der `main`-Methode, in welcher ihre gewählten Antworten noch einmal ausgegeben werden. Geben Sie in Moodle die Datei `Aufgabe1.java` mit Ihren gewählten Antworten ab.

a) Betrachten Sie die Klasse `Aufgabe1a`.

```
public class Aufgabe1a {

	public static void main(String[] args) {
		int a = 7;
		int b = 3;
		magic(a, b);
		System.out.println("a = " + a + "; b = " + b);
	}

	public static void magic(int a, int b) {
		if (a > b) {
			int tmp = a;
			a = b;
			b = tmp;
			return;
		}
	}
}
```

Geben Sie an, welche der folgenden Aussagen korrekt sind.

A. Die Werte der Variablen `a` und `b` in der `main`-Methode sind nach Aufruf der Methode `magic` vertauscht.
B. Die Methode `magic` vertauscht die Werte der beiden Parameter innerhalb der Methode, wenn der Wert von `a` größer als der Wert von `b` ist.
C. Wenn `a` und `b` denselben Wert haben, wird die Methode `magic` nicht aufgerufen.
D. Beim Kompilieren kommt es zu einer Fehlermeldung, da eine `return`-Anweisung nicht für eine `void`-Methode zulässig ist.
E. Es handelt sich um das Prinzip „pass-by-reference“.
F. Es handelt sich um das Prinzip „pass-by-value“.

b) Betrachten Sie die Klasse `Aufgabe1b`.

```
public class Aufgabe1b {

	public static void main(String[] args) {
		int[] a = { 7, 3 };
		magic(a);
		System.out.println("a[0] = " + a[0] + "; a[1] = " + a[1]);
	}

	public static void magic(int[] a) {
		a[0] = a[1] - a[0];
		a[1] = a[1] - a[0];
		a[0] = a[1] + a[0];
	}
}
```

Geben Sie an, welche der folgenden Aussagen korrekt sind.

A. Wenn das Arrays am Index 1 den Wert 0 hat, so hat der Aufruf der Methode `magic` keinen Effekt.
B. Die Werte der Parameter `a[0]` und `a[1]` bleiben innerhalb der Methode `magic` unverändert.
C. Die Werte von `a[0]` und `a[1]` in der `main`-Methode sind nach Aufruf der Methode `magic` vertauscht.
D. Der Aufruf der Methode `magic` wäre für jedes beliebige Array ohne Fehler möglich.
E. Es handelt sich um das Prinzip „pass-by-reference“.
F. Es handelt sich um das Prinzip „pass-by-value“.

c) Betrachten Sie die Klasse `Aufgabe1c`.

```
public class Aufgabe1c {

	public static void main(String[] args) {
		int[] a = { 7, 3, 5, 8, 2 };
		magic(a);
		for (int i = 0; i < a.length; i++) {
			if (i < a.length - 1) {
				System.out.print(a[i] + ", ");
				continue;
			}
			System.out.println(a[i]);
		}
	}

	public static int[] magic(int[] a) {
		int[] b = new int[a.length];
		for (int i = 0; i < b.length; i++) {
			a[i] = 2 * b[i];
		}
		return b;
	}
}
```

Geben Sie an, welche der folgenden Aussagen korrekt sind.

A. Nach Aufruf der `main`-Methode wird auf der Konsole „`14, 6, 10, 16, 4`“ ausgegeben.
B. Nach Aufruf der `main`-Methode besitzt das Array `a` andere Werte.
C. Der Aufruf der Methode `magic` innerhalb der Methode `main` hat keinen Einfluss auf das dort angegebene Array `a`, da der Rückgabewert ignoriert wird.
D. Innerhalb der Methode `magic` werden alle Werte des Arrays `a` auf 0 gesetzt.
E. Es handelt sich um das Prinzip „pass-by-reference“.
F. Es handelt sich um das Prinzip „pass-by-value“.

d) Betrachten Sie die Klasse `Aufgabe1d`.

```
public class Aufgabe1d {

	public static void main(String[] args) {
		int a = 3;
		int[] b = { 7, 3, 5 };
		b = magic(a, b);
		for (int i : b) {
			System.out.print(i + ", ");
		}
	}

	public static int[] magic(int a, int[] b) {
		int[] c = new int[a * b.length];
		for (int i = 0; i < c.length; i++) {
			c[i] = b[i % a];
		}
		return c;
	}
}
```

Geben Sie an, welche der folgenden Aussagen korrekt sind.

A. Nach Aufruf der `main`-Methode bleibt das Array `b` unverändert.
B. Wenn die `magic`-Methode anstatt mit `a = 3` mit `a = 1` aufgerufen wird, dann wird auf der Konsole „`7, 3, 5, `“ ausgegeben.
C. Nach Aufruf der Methode `main` ist das Array `b` größer als vorher.
D. Nach Aufruf der Methode `magic` innerhalb der `main`-Methode stehen im Array `b` genau 3-mal hintereinander die ursprünglichen Werte von `b`.
E. Beim Aufruf der Methode `magic` mit den Variablen `a` und `b` in der `main`-Methode handelt sich bezogen auf die Variable `b` um das Prinzip „pass-by-value“.
F. Beim Aufruf der Methode `magic` mit den Variablen `a` und `b` in der `main`-Methode handelt sich bezogen auf die Variable `b` um das Prinzip „pass-by-reference“.

e) Betrachten Sie die Klasse `Aufgabe1e`.

```
public class Aufgabe1e {

	public static void main(String[] args) {
		int a = 123;
		System.out.println(magic2(magic1(a)));
	}

	public static int[] magic1(int a) {
		int[] b = new int[a + "".length()];
		int i = 0;
		while (a > 0) {
			b[i++] = a % 10;
			a /= 10;
		}
		return b;
	}

	public static int magic2(int[] b) {
		int a = 0;
		for (int i = 0; i < b.length; i++) {
			a += b[i];
		}
		return a;
	}
}
```

Geben Sie an, welche der folgenden Aussagen korrekt sind.

A. Die Methode `magic1` funktioniert für beliebige Wert von `a` ohne Fehler.
B. Der Aufruf `magic2(magic1(magic2(magic1(a))))` liefert für jeden beliebigen Wert von `a` denselben Wert wie der Aufruf `magic2(magic1(a))`.
C. Das Programm berechnet die Quersumme einer Zahl `a > 0`.
D. Das Programm berechnet den Kehrwert einer Zahl `a > 0`.
E. Beim Aufruf der Methode `magic1` mit der Variablen `a` in der `main`-Methode handelt sich um das Prinzip „pass-by-reference“.
F. Beim Aufruf der Methode `magic1` mit der Variablen `a` in der `main`-Methode handelt sich um das Prinzip „pass-by-value“.

Aufgabe 2 (Armstrong-Zahlen) 6 Punkte

Armstrong-Zahlen sind narzisstische Zahlen, deren Summe ihrer Ziffern, jeweils potenziert mit der Stellenanzahl der Zahl, wieder die Zahl selbst ergibt. Beispielsweise ist die Zahl 153 eine Armstrong-Zahl, da $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ gilt.

Gegeben ist die Klasse `ArmstrongNumbers`. Vervollständigen Sie einerseits die Methode `public static boolean isArmstrongNumber(int number)`, welche für den Parameter `number` prüft, ob diese Zahl die Bedingung einer Armstrong-Zahl erfüllt. Vervollständigen Sie weiterhin die Methode `public static int[] giveArmstrongNumbers(int n)`, welche die ersten n Armstrong-Zahlen in einem Array zurückgibt. Geben Sie als Lösung das Programm `ArmstrongNumbers.java` in Moodle ab.

```
javac ArmstrongNumbers.java
java ArmstrongNumbers 15
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634
```

Hinweise zur Bearbeitung

- Nutzen Sie die Methode `private static void printArray(int[] a)` der Klasse `ArmstrongNumbers`, um das Array mit den n Armstrong-Zahlen auszugeben.
- Um die Anzahl der Stellen einer Zahl zu ermitteln, könnten die folgenden Methoden aus der Klasse `java.lang.Math` hilfreich sein[1].
    - `public static double log10(double a)`
    - `public static double floor(double a)`
- Um die b-te Potenz einer Zahl a zu berechnen, können Sie die folgende Methode aus der Klasse `java.lang.Math` verwenden.
    - `public static double pow(double a, double b)`

[1] https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Aufgabe 3 (Pythagoras-Baum) 8 Punkte

Der Pythagoras-Baum ist ein Fraktal, bei welchem auf der oberen Seite eines Quadrats zwei kleinere Quadrate (Ankathetenquadrat und Gegenkathetenquadrat) so angeordnet werden, dass diese ein rechtwinkliges Dreieck umschließen (siehe Abb. 2.1).

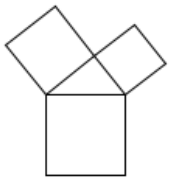

Abb. 2.1: Möglicher Pythagoras-Baum für n = 1

Wird dieser Vorgang der Konstruktion mehrfach durch Rekursion wiederholt, so entsteht anhand dieser Vorschrift ein Gebilde, das einem Baum ähnelt (siehe Abb. 2.2).

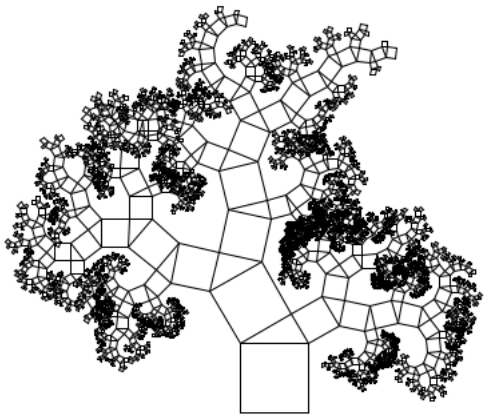

Abb. 2.2: Möglicher Pythagoras-Baum für n = 12

Schreiben Sie ein Programm `PythagorasTree.java`, welches für einen Kommandozeilenparameter n die angegebene Konstruktionsvorschrift n-fach rekursiv wiederholt und somit die rekursiven Strukturen für n = 1, 2, 3, 4, 12 erzeugt (siehe Abb. 2.1, Abb. 2.2 und Abb. 2.3).

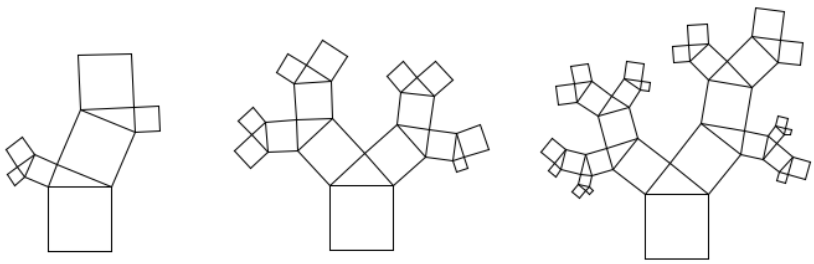

Abb. 2.3: Möglicher Pythagoras-Baum für n = 2, 3, 4

Die Ankatheten und Gegenkatheten sollen für jedes Dreieck zufällig gebildet werden, sodass der Winkel α zwischen Ankathete und Hypotenuse am oberen linken Eckpunkt des großen Quadrats größer gleich 30° (π / 6) und echt kleiner 60° (π / 3) ist.

Fügen Sie außerdem mindestens folgende Kommentare zu Ihrem Quelltext hinzu (Java-Doc):

- Klasse: kurze Beschreibung, `@author` und `@version`
- Methoden (inklusive `main`-Methode): kurze Beschreibung, `@param` und `@return`

Hinweise zur Bearbeitung

- Nutzen Sie die Methode `line(double x0, double y0, double x1, double y1)` aus der Bibliothek `gdp.stdlib.StdDraw`.
- Zur Berechnung der neuen Koordinaten und Seitenlängen benötigen Sie ggf. den Sinus und Kosinus eines Winkels, welche Sie mit Methoden aus der Klasse `java.lang.Math` berechnen können[2].
  - `public static double sin(double a)`
  - `public static double cos(double a)`
- Die abgebildeten Pythagoras-Bäume zeigen nur Beispielausgaben. Ihre Programme sollten sich aufgrund der zufälligen Auswahl der Dreiecke unterscheiden.

Viel Spaß und Erfolg bei der Lösung der Aufgaben!

[2] https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html