Projektblatt P2 (20 P)

Abgabe: Montag 29. September 2023, 12:00h

Entpacken Sie zunächst die Archiv-Datei `vorgaben-p02.zip`, in der sich neben mehreren Rahmendateien für die zu lösenden Aufgaben auch die Hilfsklasse `IOTools` befindet. Ergänzen Sie alle Dateien mit Ausnahme von `IOTools.java` zunächst durch einen Kommentar, der Ihren Namen beinhaltet. Ergänzen Sie die Dateien dann durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der abgeänderte bzw. hinzugefügte Java-Sourcecode in den Dateien `Aufgabe_2_1.java` (nur Kommentare), `Aufgabe_2_2.java`, `Aufgabe_2_3.java` und
`Aufgabe_2_4.java` sollte syntaktisch richtig und vollständig formatiert sein. Antworten in Textform müssen stets in Form von (Block-)Kommentaren erfolgen. Alle Dateien mit der Endung `.java` sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P02.zip`, die Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P02.zip *.java
```

Aufgabe 1: Code nachvollziehen 5 P

In jedem der folgenden Codestücke werden zunächst ein bzw. zwei ganzzahlige Werte eingelesen. Anschließend wird in Abhängigkeit von diesen Variablen ein Ergebniswert `erg1`, `erg2` bzw. `erg3` berechnet und ausgegeben:
Analysieren Sie die Codestücke und ergänzen Sie die Datei `Aufgabe_2_1.java`, an den (mit `TODO`) markierten Stellen um eine Beschreibung oder, falls möglich, durch eine Formel, die den Ergebniswert in Abhängigkeit von der Eingabe beschreibt. Begründen Sie Ihre Antwort, indem Sie beschreiben, was in den einzelnen Kontrollanweisungen jeweils geschieht und welche Bedeutung dies für das Gesamtergebnis hat.
In Teil (c) gilt das Ergebnis nur, wenn die Eingabewerte positiv bzw 0 sind. Daher wird das Einlesen solange wiederholt, bis der/die User:in positive Werte eingibt.

Beispiel: Für das Codestück

```
1 int m = IOTools.readInteger("m = ");
2 int n = IOTools.readInteger("n = ");
3 int r = 0;
4 for (int i=1;i<=m; i++){
5   r += 2;
6 }
7 for (int i=1;i<=n; i++){
8   r += 3;
9 }
```

sollte die Erläuterung etwa wie folgt aussehen:

```
f(m,n) = 2*m + 3*n
Beginnend mit r = 0 wird in der ersten Schleife m Mal der Wert
2 zu r addiert.
Danach hat r den Wert 2*m
In der zweiten Schleife wird n Mal der Wert 3 zu r addiert.
Daher hat r am Ende den Wert 2*m + 3*n
```

(a)

```
1   int n = IOTools.readInteger("n = ");
2
3   boolean erg1;
4
5   n = (n < 0) ? -n : n;
6
7   while (n > 0) {
8       n--;
9       n--;
10  }
11
12  erg1 = (n == 0);
```

(b)

```
1   int a = IOTools.readInteger("a = ");
2   int A = a;      // Originalwert von a merken
3   int b = IOTools.readInteger("a = ");
4   int B = b;      // Originalwert von a merken
5   int erg2;
6
7   if ((a <= 0) && (b <= 0)) {
8       while ((a != 0) && (b != 0)) {
9           a++;
10          b++;
11      }
12      erg2 = (a == 0) ? A : B;
13  }
14  else if ((a >= 0) && (b >= 0)) {
15      while ((a != 0) && (b != 0)) {
16          a--;
17          b--;
18      }
19      erg2 = (a == 0) ? B : A;
20  }
```

```
21          else if (a < 0) {
22              erg2 = B;
23          }
24          else {
25              erg2 = A;
26          }
```

(c)

```
1           /* Hier werden zunaechst zwei Werte a, b mit
2              a >= 0 und b >= 0 eingelesen
3            */
4           do {
5               a = IOTools.readInteger("a = ");
6           } while (a <= 0);
7           do {
8               b = IOTools.readInteger("b = ");
9           } while (b <=0);
10
11
12
13          int x=0,y=0,z=0;
14
15          for (int i=1;i<=a;i++){
16              y += b;
17              for (int j=1;j<=a;j++,x++){}
18          }
19
20          for (int i=1;i<=b;i++){
21              y += a;
22              for (int j=1;j<=b;j++,z++){}
23          }
24
25          int erg3 = x + y + z;
```

Anmerkung: Die Codestücke stellen nicht unbedingt die sinnvollste Art und Weise dar, das jeweilige Ergebnis zu berechnen.

Aufgabe 2: `while`- Schleifen, Iteration 4 P

Ergänzen Sie die `main`-Methode der Datei `Aufgabe_2_2.java` so, dass für eine beliebige ganze Zahl vom Typ `long` die Anzahl der geraden und der ungeraden Ziffern der Zahl berechnet wird. Die Zahl 0 gilt als gerade. Sie dürfen dabei davon ausgehen, dass der eingegebene Wert größer als 0 ist. Gehen Sie hierzu wie folgt vor:

(1) Deklarieren Sie eine Variable `zahl` vom Typ `long`, die Sie mit einem Wert initialisieren, den Sie mit Hilfe einer passenden Methode aus der Klasse `IOTools` einlesen. Wiederholen Sie das Einlesen solange, bis der/die User*in einen Wert eingegeben hat, der größer oder gleich 10 ist.

(2) Legen Sie nun zwei ganzzahlige Variablen an, die die Anzahl der geraden sowie der ungeraden Ziffern speichern sollen und initialisieren Sie diese Variablen geeignet. Verwenden Sie dann eine `while`-Schleife, in der Sie in jedem Durchlauf prüfen, ob die letzte Ziffer der Variablen `zahl` gerade oder ungerade ist und aktualisieren Sie die zugehörige Zählervariable entsprechend. Verändern Sie dann die Variable `zahl`, indem Sie die letzte Ziffer entfernen. Führen Sie die Schleife solange fort, bis `zahl` den Wert 0 hat.

(3) Geben Sie das Ergebnis wie in folgendem Beispiel aus:

```
Zahl =  122333444455555
Anzahl gerader Ziffern:   6
Anzahl ungerader Ziffern: 9
```

Aufgabe 3: `for`- Schleifen, Iteration 5 P

Eine (unendliche) Zahlenfolge ist eine Folge von Werten $a_1, a_2, a_3, \dots$ (mit $a_i \in \mathbb{R}$), deren Abfolge einer bestimmten Gesetzmäßigkeit unterliegt. Diese Gesetzmäßigkeit definiert den Wert des n-ten Folgengliedes a_n in Abhängigkeit von n und/oder von ein oder mehreren $a_j (1 \leq j < n)$. Um die Folge berechnen zu können, wird das erste Folgenglied a_1 bzw. ggf. weitere Anfangsglieder (z.B. a_2, a_3) vorgegeben.
Beispiele:

- $a_n = n^2$ ist beispielsweise die Folge der Quadratzahlen 1,4,9,16, $\cdots$
- $a_n = 2 * a_{n-1}$ definiert eine Folge, bei der jedes Folgenglied den doppelten Wert des vorhergehenden Wertes hat: $1, 2, 4, 8, 16, 32, \cdots$ In diesem Beispiel lautet der Anfangswert $a_1 = 1$
- die Folge 5, 10, 6, 12, 8, 16, 12, 24, $\cdots$ wird durch folgende Gesetzmäßigkeit beschrieben: $a_1 = 5$ und für $n > 1 :\ a_n = a_{n-1} * 2$, falls n gerade und $a_n = a_{n-1} - 4$, falls n ungerade
- die Folge der Fibonaccizahlen 1,1,2,3,5,8,13,21, $\cdots$ wird wie folgt gebildet: $a_1 = 1, a_2 = 1$, $a_n = a_{n-1} + a_{n-2} \forall n > 2$

Die Datei `Aufgabe_2_3.java` enthält drei Methoden `folge_a`, `folge_b` und `folge_c`, die jeweils die ersten $n = 15$ Elemente der in den Aufgabenteilen (a) - (c) angegebenen Folgen iterativ berechnen und in der Konsole ausgeben sollen.
Ergänzen Sie nun diese Datei an den mit `Aufgabenteil (a)`, `Aufgabenteil (b)` bzw. `Aufgabenteil (c)` markierten Stellen wie im Folgenden beschrieben:

- Analysieren Sie die jeweilige Zahlenfolge und ergänzen Sie den Kommentar oberhalb der zugehörigen Methode durch eine kurze Beschreibung des Schemas, nach dem die Folgenglieder berechnet werden. Geben Sie, wenn möglich, eine Formel an, ansonsten verwenden Sie eine verbale Beschreibung.
- Überlegen Sie sich die Anzahl der nötigen initialen Elemente $(a_1, a_2, \dots)$, deklarieren Sie in der jeweiligen Methode eine entsprechende Anzahl von Hilfsvariablen und initialisieren Sie diese entsprechend. Geben Sie diese Elemente anschließend in der Konsole aus.

- Verwenden Sie nun eine `for`-Schleife, mit der Sie die restlichen Elemente (15 - der Anzahl der initialen Elemente) in der Methode berechnen und ausgeben können. Definieren Sie sich zunächst (vor der Schleife) eine Variable, die das nächste Folgenglied repräsentiert. Darüberhinaus dürfen Sie sich ggf. auch weitere Hilfsvariablen definieren. Berechnen Sie dann in jedem Schleifendurchlauf das nächste Element und geben Sie es in der Konsole aus. Aktualisieren Sie ggf. die Hilfsvariablen.

(a) 0 1 2 3 6 11 20 37 68 ...

(b) 1 6 4 9 7 12 10 15 ...

(c) 2 2 5 11 20 32 47 65 ...

Die Ausgabe sollte am Ende dann wie folgt aussehen:

```
Aufgabenteil (a)
0 1 2 3 6 11 20 37 68 125 230 423 778 1431 2632
Aufgabenteil (b)
1 6 4 9 7 12 10 15 13 18 16 21 19 24 22
Aufgabenteil (c)
2 2 5 11 20 32 47 65 86 110 137 167 200 236 275
```

Aufgabe 4: Geschachtelte Schleifen 6P

Ergänzen Sie die Datei `Aufgabe_2_4.java` an den beiden mit `// TODO` markierten Stellen wie im Folgenden beschrieben:

(a) Zunächst sollen Sie das folgende Gleichungssystem mit den Variablen a, b, c durch "Ausprobieren" möglicher Werte lösen. Dabei können Sie davon ausgehen, dass die Lösung(en) positive, ganze Zahlen aus dem Intervall $[1, 1000]$ sind.

$$\begin{aligned} a * b + c &= 2023 \\ a + b * c &= 2024 \end{aligned}$$

Gehen Sie wie folgt vor:

(1) Berechnen Sie mit Hilfe einer verschachtelten Schleife für alle möglichen Zahlenpaare (a, b) mit $1 \leq a, b \leq 1000$ den Wert von `c`, wie er sich für die erste und die zweite Gleichung ergibt.

(2) Wenn die beiden Werte für c übereinstimmen, geben Sie die gefundene Lösung, wie unten angegeben, in der Konsole aus.

(3) Achten Sie dabei darauf, dass Sie beim Berechnen von `c` in der zweiten Gleichung durch eine Integer-Division manchmal einen falschen Wert erhalten. Kompensieren Sie dies dadurch, indem Sie eine zusätzliche Bedingung prüfen, bevor Sie eine Lösung ausgeben (Hinweis: Rückrechnen).

Bei korrekter Implementierung sollte die Ausgabe wie folgt aussehen:

```
a = 674  b = 2  c = 675
```

(b) Bestimmen Sie alle Lösungstripel (x, y, z) der Gleichung

$$x^2 + y^2 + z^2 = 3 * x * y * z$$

Hierbei sollen die Werte von x, y und z auf ganzzahlige Werte aus dem Intervall $[1, 500]$ beschränkt werden. Bestimmen Sie nur die Lösungen für die gilt: $x \leq y \leq z$.

Gehen Sie analog zu Aufgabenteil (a) so vor, dass Sie mit Hilfe einer geschachtelten Schleife für alle alle Werte-Tripel (x, y, z) mit $0 < x, y, z \leq 500$ und $x \leq y \leq z$ prüfen, ob die Werte auf der linken und der rechten Seite der Gleichung übereinstimmen.

Hinweis: Um die Bedingung $x \leq y \leq z$ einzuhalten, müssen Sie die initialen Werte für die Variablen in den inneren Schleifen an die Werte der jeweils übergeordneten Schleife anpassen.

Bei korrekter Implementierung sollte die Ausgabe wie folgt aussehen:

```
(1, 1, 1)
(1, 1, 2)
(1, 2, 5)
(1, 5, 13)
(1, 13, 34)
(1, 34, 89)
(1, 89, 233)
(2, 5, 29)
(2, 29, 169)
(5, 13, 194)
(5, 29, 433)
```