Projektblatt P2 (23 + 5* Punkte)

Abgabe: Donnerstag 26. September 2024, 12:00h

Entpacken Sie zunächst die Archiv-Datei `vorgaben-P2.zip`, in der sich neben mehreren Rahmendateien für die Aufgaben auch nochmal die Hilfsklasse `IOTools` befindet. Ergänzen Sie die Dateien zunächst durch Ihren Namen. Ergänzen Sie die Dateien dann durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der hinzuzufügende Java-Sourcecode sollte syntaktisch richtig und vollständig formatiert sein. Alle `.java` Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.zip`, welches 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.zip  *.java
```

Aufgabe 1: Code nachvollziehen 6 P

In jedem der folgenden Codestücke werden zunächst ein oder mehrere Variablenwerte eingelesen. Anschließend wird in Abhängigkeit von diesen Variablen ein Funktionswert berechnet und als Ergebnis ausgegeben:

- $r1 = f_1(a, b)$
- $r2 = f_2(n)$
- $r3 = f_3(a, b)$

Analysieren Sie die Codestücke und ergänzen Sie die Dateien `Aufgabe_2_1a.java`, `Aufgabe_2_1b.java` und `Aufgabe_2_1c.java` an den markierten Stellen um eine Beschreibung oder, wenn möglich, eine Formel für die Funktionen f_1, f_2 und f_3. Begründen Sie Ihre Antwort, indem Sie beschreiben, was in den einzelnen Kontrollanweisungen jeweils geschieht und welche Bedeutung dies für das Gesamtergebnis hat.
Sie dürfen voraussetzen, dass alle Eingabewerte positive Zahlen sind
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 a = IOTools.readInteger("a = ");
2   int b = IOTools.readInteger("b = ");
3   int r1 = 0;
4
5   for (int i=a;i<=b;i++){
6       switch(i%4){
7       case 1:
8       case 2:
9       case 3:
10          break;
11      default:
12          r1++;
13          break;
14      }
15  }
16  System.out.println("r1 = " +  r1);
```

(b)

```
1   int n = IOTools.readInteger("n = ");
2   int a = n;
3   int b = 1;
4   int r2 = 0;
5
6   while (a > b){
7       r2 += (a + b);
8       a--;
9       b++;
10  }
11  if (a == b){
12      r2 += a;
13  }
14  System.out.println("r2 = " +  r2);
```

(c)

```
1         int a = IOTools.readInteger("a = ");
2         int b = IOTools.readInteger("b = ");
3         int r3 = 0;
4         int x=0,y=0,z=0;
5
6         for (int i=1;i<=a;i++){
7             y += b;
8             for (int j=1;j<=a;j++,x++){}
9         }
10
11        for (int i=1;i<=b;i++){
12            y += a;
13            for (int j=1;j<=b;j++,z++){}
14        }
15
16        r3 = x + y + z;
17
18        System.out.println("r3 = " +  r3);
```

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

Hinweis: Überlegen Sie sich, welche Funktionen hier implementiert werden und überprüfen Sie mit Hilfe der vorgegebenen Programme in den Dateien `Aufgabe_2_1a.java`, `Aufgabe_2_1b.java` und `Aufgabe_2_1c.java` Ihre Hypothesen, indem Sie die Programme übersetzen und mit verschiedenenen Eingabewerten testen.

Aufgabe 2 — 7 P

Eine (unendliche) Zahlenfolge ist eine Folge von Werten $a_1, a_2, a_3, \ldots$ (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 $a_j (1 < j < n)$. 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_2.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.
- Legen Sie eine Variable für a_n an und initialisieren Sie dieses mit dem ersten Element der Folge. Sie dürfen hier davon ausgehen, dass Sie nur ein initiales Element benötigen um alle weiteren Elemente der Folge zu berechnen.

- 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. Berechnen Sie in jedem Schleifendurchlauf das nächste Element, weisen Sie den Wert der Variable für a_n zu und geben Sie den Wert in der Konsole aus.

  Sie dürfen sich ggf. weitere Hilfsvariablen definieren, die Sie zu Beginn (vor der Schleife) passend initialisieren und in jedem Schleifendurchlauf nach der Ausgabe des nächsten Folgenelements aktualisieren.

(a) 1 2 5 14 41 122 ...

(b) 3 4 2 5 1 6 0 7 ...

(c) 12 6 3 8 4 2 1 6 ...

Die Ausgabe sollte am Ende dann wie folgt aussehen:

```
Aufgabenteil (a)
1 2 5 14 41 122 365 1094 3281 9842 29525 88574 265721 797162 2391485
Aufgabenteil (b)
3 4 2 5 1 6 0 7 -1 8 -2 9 -3 10 -4
Aufgabenteil (c)
12 6 3 8 4 2 1 6 3 8 4 2 1 6 3
```

Hinweis: In Aufgabenteil (c) hängt der Wert des jeweils nächsten Folgenglieds davon ab, ob das aktuelle Element gerade oder ungerade ist.

Aufgabe 3: 5 P

Ergänzen Sie die `main`-Methode in der Klasse `Aufgabe_2_3` so, dass eine Buchstabenpyramide mit einer vom User zu spezifizierenden Höhe h in der Konsole ausgegeben wird. Für $h = 7$ sollte diese wie folgt aussehen:

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
ABCDEFG
ABCDEF
ABCDE
ABCD
ABC
AB
A
```

Gehen Sie dazu wie folgt vor:

- Fordern Sie den User auf, einen Wert für h anzugeben und lesen Sie diesen dann mit Hilfe der `IOTools` Klasse ein. Wiederholen Sie dies ggf. bis der User einen Wert aus dem Intervall $[3, 26]$ eingeben hat.
- Generieren Sie dann in einer Doppelschleife die ersten h Zeilen der Pyramide. In der inneren Schleife soll dabei jeweils eine Zeile mit den ersten k (mit k = 1, 2, ..., h) Buchstaben des Alphabets ausgegeben werden.
- Generieren Sie dann in einer Doppelschleife die fehlenden $h - 1$ Zeilen der Pyramide.In der inneren Schleife soll dabei jeweils eine Zeile mit den ersten k (mit k = h-1, h-2, ..., 1) Buchstaben des Alphabets ausgegeben werden.

Aufgabe 4: 5 P

Mit dem Newton-Verfahren kann man die Wurzel einer Zahl a durch eine Folge $(x_i)_{i=1,2,...}$ wie folgt mit zunehmender Genauigkeit approximieren:

$$x_{n+1} = \frac{x_n}{2}\left(3 - \frac{x_n^2}{a}\right)$$

Dabei kann x_1 wie folgt gewählt werden:

$$x_1 := \frac{(1+a)}{2}$$

Ergänzen Sie die `main`-Methode in der Klasse `Aufgabe_2_4`, so dass dieses Verfahren angewendet wird um die Wurzel einer vom User spezifizierten Zahl x iterativ zu approximieren. Ein Abbruch der Approximation soll erfolgen, wenn der Absolutbetrag der Differenz zweier aufeinanderfolgender Folgenglieder kleiner als 10^{-8} ist. Geben Sie das Resultat zusammen mit der Anzahl der durchgeführten Iterationsschritte in der Konsole aus. Für die beiden Aufrufe des Programms mit den Werten $x = 2$ und $x = 3$ sollte die Ausgabe dann wie folgt aussehen:

```
Wurzel aus 2.0 approx. 1.4142135623730951 (4 Iterationen)
Wurzel aus 3.0 approx. 1.7320508075688772 (5 Iterationen)
```

Verwenden Sie zur Berechnung des Absolutbetrags die Methode `Math.abs` Darüber hinaus dürfen sie keine weiteren Methoden aus der `Math`-Klasse verwenden.

Aufgabe 5: 5* P

Ergänzen Sie die `main`-Methode der Klasse `Aufgabe_2_5` in der Datei `Aufgabe_2_5.java` so, dass das resultierende Programm ein zweidimensionales, regelmäßiges "Molekülgitter" der Größe $n \times n$ (mit $n > 0$) auf der Kommandozeile ausgibt. Die Moleküle sollen dabei durch das Zeichen 'o' repräsentiert werden und die Verbindungen zwischen den Molekülen mit dem Zeichen '-' für horizontale Verbindungen und mit dem Zeichen '|' für senkrechte Verbindungen. Gehen Sie dabei wie folgt vor:

(1) Lesen Sie mit Hilfe der IOTools in einer Schleife solange eine ganze Zahl ein, bis der Benutzer eine ganze, positive Zahl eingibt.

(2) Geben Sie im Wechsel zwei unterschiedliche Zeilen aus:

- die erste Zeile besteht aus $(n-1)$ Wiederholungen der Zeichenkette "o-" und einem einzelnen Zeichen 'o' am Ende
- die zweite Zeile besteht aus n Wiederholungen der Zeichenkette "| ", welche aus dem Zeichen '|' und einem Leerzeichen besteht.

Die erste Zeile muss dabei n Mal wiederholt werden und die zweite $n-1$ Mal. Hinweis: Damit die zweite Zeile nicht einmal zu viel ausgeführt wird, sollten Sie die Schleife, in der die beiden Zeilen nacheinander generiert und ausgegeben werden, rechtzeitig beenden.

Für die Werte $n = 1, 2, 3, 4$ sollten folgende Ausgaben produziert werden:

```
o

o-o
| |
o-o

o-o-o
| | |
o-o-o
| | |
o-o-o

o-o-o-o
| | | |
o-o-o-o
| | | |
o-o-o-o
| | | |
o-o-o-o
```