W02H01 - Meet and Greet in der Antarktis

Hausaufgabe Easy

Points: 8 of 8 Assessment: manual

Submission due: 9 months ago

Complaint due: 9 months ago

Resume practice in exercise

100% (9 months ago) GRADED

Recent results:

75% (10 months ago) GRADED

75% (10 months ago) GRADED

100% (10 months ago) GRADED

100% (9 months ago) GRADED

100% (9 months ago) GRADED

Show all results

Tasks:

Meet and Greet in der Antarktis

Hinweis: In der ganzen Aufgabe ist das Benutzen der `java.lang.Math` Library (oder anderer ähnlicher Bibliotheken, die den von dir zu leistenden Anteil der Implementierung ersetzen wie `BigInteger`, `BigDecimal`, `StrictMath`, ...) untersagt. Wir behalten uns vor, Abgaben, die diesem Kriterium widersprechen, entweder komplett oder einzelne Teilaufgaben davon mit 0 Punkten zu bewerten.

Die Pinguine haben dich in ihr Zuhause in der Antarktis eingeladen. Sie wollen dir stolz präsentieren, wie sie leben und du willst so viel du kannst über sie erfahren. Da Pinguine intelligent und kompetitiv sind, lassen sie es sich natürlich nicht nehmen, dir ein paar knifflige Aufgaben zu stellen. Als angehedes TUM-Mitglied bist du dieser Situation selbstverständlich bestens gewachsen. Los! Zeig ihnen, was du drauf hast! Viel Spaß!

Hinweis: Du darfst in dieser Aufgabe davon ausgehen, dass alle Integer-Parameter der Methoden in den Teilaufgaben 2 bis 5 nicht-negativ sind.

1. **Pinguin Informationen** No results

   Implementiere die Methode `penguInfoOut(int)`. Der Methode wird ein Pinguin vom Typ `int` übergeben. Weil Pinguine nie depressiv sind, kennen wir nur *nicht negative* Pinguine. Sollte der Methode ein unbekannter Pinguin übergeben werden, soll in der Konsole die Nachricht "`Penguin_`<penguin>`_is_not_a_known_penguin!`" in einer eigenen Zeile ausgegeben werden. Andernfalls soll die Methode eine kurze Beschreibung des Pinguins ausgeben (ebenfalls direkt in der Konsole). Die erste Zeile der Beschreibung enthält dabei folgenden String: "`Penguin:_`<penguin>". Da selbstverständlich sowohl weibliche als auch männliche Pinguine existieren, möchten wir diese Unterscheiden. Zum Glück ist das sehr einfach, denn während *männliche* Pinguine immer *geradzahlig* sind, lassen sich *weibliche* Pinguine eindeutig als *ungerade Zahlen* identifizieren. Diese Information soll nun in der zweiten Zeile wie folgt ausgegeben werden: "`This_penguin_is_a_female.`" (f) oder "`This_penguin_is_a_male.`" (m) angepasst an das entsprechende Geschlecht des Pinguins. Hier hast du noch ein paar Beispiele, an denen du dich orientieren kannst:

```
Beispiel 1:
IN: penguin = -99
OUT 1: "Penguin -99 is not a known penguin!"

Beispiel 2:
IN:penguin = 1
OUT 1: "Penguin: 1"
OUT 2: "This penguin is a female."

Beispiel 3:
IN: penguin = 2
OUT 1: "Penguin: 2"
OUT 2: "This penguin is a male."
```

2. **Evolution der Pinguine** No results

   Implementiere die Methode `penguEvolution(int, int)`. Der Methode wird ein Pinguin vom Typ `int` und eine Anzahl von Jahren (`int`) übergeben. In dieser Aufgabe möchten wir die Evolution eines Pinguins aus einer ganz speziellen Kolonie simulieren (`penguin` > 0). Jedes Jahr verändert sich ein Pinguin nach den folgenden Regeln: Da *Pinguin-Männer* immer gestresst sind und sich so viel beschweren, *halbieren* männliche Pinguine ihre Zahl. Besonders gestresste *männliche Pinguine* sind Zweierpotenzen ($2^x, x \in \mathbb{N} \setminus \{0\}$). Anstatt sich nur zu halbieren, werden diese direkt zu einer `1`. *Weibliche Pinguine* hingegen sind so gut gelaunt und tanzen deshalb so viel, dass sie ihre Zahl *verdreifachen und `1` hinzu addieren*. Aus traditionellen Gründen sind die *Pinguin-Damen* am glücklichsten, wenn sie ein Vielfaches von `7` sind, deshalb verbleiben sie in diesem Zustand auch für genau `7` Jahre, ehe sie ihre Evolution fortsetzen. Achte darauf, dass die Pinguine aus unserer speziellen Kolonie dadurch von Jahr zu Jahr ihr Geschlecht ändern könnten. Die Methode soll den Zustand/die Zahl des übergebenen Pinguins (`penguin`) nach der übergebenen Anzahl von

Jahren (`years`) zurückgeben. In den Beispielen sind immer nach "IN: " die Parameter, nach "OUT: " der Rückgabewert, der bei diesen Parametern berechnet werden sollte, sowie nach "EVO: " die gesamte Evolution über den übergebenen Zeitraum angegeben. Die Evolution ist zur Verdeutlichung dabei und soll **nicht** extra ausgegeben werden. Der korrekte Rückgabewert reicht.

```
Beispiel 1:
IN: penguin = 128, years = 2
OUT: 4
EVO: Jahr 0: 128
     Jahr 1: 1
     Jahr 2: 4

Beispiel 2:
IN: penguin = 9, years = 9
OUT: 7
EVO: Jahr 0: 9
     Jahr 1: 28
     Jahr 2: 14
     Jahr 3: 7
     Jahr 4: 7
     Jahr 5: 7
     Jahr 6: 7
     Jahr 7: 7
     Jahr 8: 7
     Jahr 9: 7

Beispiel 3:
IN: penguin = 9, years = 10
OUT: 22
EVO: Jahr  0: 9
     Jahr  1: 28
     Jahr  2: 14
     Jahr  3: 7
     Jahr  4: 7
     Jahr  5: 7
     Jahr  6: 7
     Jahr  7: 7
     Jahr  8: 7
     Jahr  9: 7
     Jahr 10: 22
```

3. ? **Pinguine, die sich selbst testen** No results

   Pinguine sind grundsätzlich sehr vergesslich und müssen sich alles auf einen kleinen Zettel schreiben, den sie unter ihrem linken Flügel aufbewahren. Dort führen sie natürlich auch eine Liste all ihrer Pinguin-Freunde. Zusätzlich zur Zahl ihres Freundes berechnen sie auch noch die Quersumme dieser Zahl, um testen zu können, dass sie die Nummer richtig notiert haben. Die Quersumme einer Zahl ist die Summe aus den einzelnen Ziffern einer natürlichen Zahl. Implementiere nun die Methode `penguSum(int)`. Die Methode soll die Quersumme der übergebenen Pinguin-Zahl (`penguin`) berechnen und zurückgeben.

```
Beispiel 1:
IN: penguin = 128
OUT: 11

Beispiel 2:
IN: penguin = 1337
OUT: 14
```

4. ? **Permutierende Pinguine** No results

   Pinguine lieben Ordnung und Steine. Am liebsten ordnen sie ihre Steine. Die Kolonie hat eine Menge von `n` vielen Steinen gesammelt. `k` dieser Steine sind identisch, da sie diese von uns geschenkt bekommen haben (d.h. $n \geq k \geq 1$). Die anderen sind alle zu diesen `k` und zueinander unterschiedlich. Die Pinguine möchten nun herausfinden wie viele Möglichkeiten es gibt, die Steine in eine Reihe zu legen. Als exzellenter TUMuin fällt dir natürlich sofort eine simple Formel ein, mit der du die Lösung berechnen kannst: $\frac{n!}{k!}$, mit $i! = 1 \cdot 2 \cdots i = \prod_{k=1}^{i} k$ (Fakultät).
   Implementiere die Methode `penguPermutation(long, long)`, der die beiden `long`s `n` und `k` übergeben werden. Achte dabei darauf, dass deine Implementierung **alle** Eingaben richtig bearbeiten muss, deren finales Ergebnis weiterhin in einen `long` passt. Ggf. solltest du dazu noch einmal einen Blick in die Vorlesung werfen, um noch einmal zu wiederholen, welche Werte ein `long` speichern kann.

```
Beispiel 1:
IN: n = 6, k = 3
OUT: 120

Beispiel 2:
IN: n = 21, k = 19
OUT: 420

Beispiel 3:
```

5. ? **Mächtige Pinguine** No results

WICHTIG: *In dieser Teilaufgabe ist das Benutzen des `*`-Operators strikt verboten! Benutzt du ihn, bekommst du 0P auf diese Teilaufgabe.*
Pinguine sind sehr intelligent. Sie fordern dich zu eine Mathe-Challenge heraus: Gegeben `x` und `i`, berechne x^i. "Haha, wie einfach! :D" Aber die Pinguine sind noch nicht fertig! Du sollst eine Lösung finden, ohne dabei den `*`-Operator zu benutzen. Schnell erinnerst du dich daran, dass Multiplikation ganz leicht durch Addition berechnet werden kann, denn $a \cdot b$ ist nichts anderes als a viele male b zu addieren (z.B. $3 \cdot 2 = 2 + 2 + 2$). x^i lässt sich genau so simpel berechnen, indem man i-mal x multipliziert. Implementiere nun die Methode `penguPowers(int, int)`, ohne den `*`-Operator zu benutzen. Der Methode werden die beiden `int`s `x` und `i` übergeben. Achte auch hier darauf, dass deine Lösung alle Eingaben, deren Ergebnis in einen `long` passt, richtig berechnet. *Hinweis:* es reicht aus, wenn du dir über *nicht negative* Eingaben gedanken machst. Negative Eingaben können als optional betrachtet werden.

```
Beispiel 1:
IN: x = 1337, i = 2
OUT: 1787569

Beispiel 2:
IN: x = 3, i = 4
OUT: 81
```

Tests

Du bekommst bei dieser Aufgabe von den PublicTests (den Tests, die direkt nach jedem Push ausgeführt werden), zwar kein Feedback darüber, was du falsch gemacht hast, allerdings teilen sie dir nach jedem Push mit, ob du die jeweilige Teilaufgabe korrekt gelöst hast oder nicht. Daher kannst du dir auch sicher sein, dass du alle Punkte bekommst, wenn du alle PublicTests bestehst (ausgenommen du hältst dich nicht an eine der in der Aufgabenstellung geforderten Einschränkungen). Die HiddenTests sind deckungsgleich mit den PublicTests und geben dir nach der Deadline ein

Exercise details	
Release date:	Oct 27, 2022 18:30
Submission due:	Nov 6, 2022 18:00
Assessment due:	Nov 13, 2022 18:00
Complaint due:	Nov 20, 2022 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left.

How useful is this feedback to you?

★ ★ ★ ★ ★