Klausur B-AI2 PM 2 SS 2018, Prof. Dr. Bernd Kahlbrandt

Matrikelnummer	Vorname	Name

Aufgabe	1.1	1.2	1.3	2	3.1	3.2		$\sum$	Note
Punkte (ca.)	2	3	3	20	5	5		38	
Erreicht									

Datum: **06.07.2018** Uhrzeit: **9:00**
Raum: **BT 7, 11.Stock** Dauer (Minuten): **120**

Hinweise

1. Verfügbare Hilfsmittel: Java API Doc, Java Source Code, Laufwerk M u. a. mit Java Sprachspezifikation (JLS9), Spezifikation der JVM (JVM9), Skript, Folien (pdf),
2. Erlaubte „Mitbringsel“: Vorlesungsmitschrift inklusive Skript/Folien mit Notizen, Wörterbuch Englisch oder Muttersprache, elektronisches Wörterbuch ohne Internetzugang („Flugmodus“), Papier, Stifte.
3. Notwendige Bestandteile:
   - 3.1. In **jeder** .java-Datei steht unmittelbar vor dem „Top-Level Element“ (Klasse, Annotation etc.) ein javadoc-Kommentar mit Ihrem Namen nach einem *@author* tag.
   - 3.2. Ihre Entwurfsentscheidungen sind in javadoc-Kommentaren oder //-Kommentaren knapp und nachvollziehbar begründet.

     Meine Vorstellung dazu: Ein javadoc-Kommentar (s. o. unter 3.1) mit den Entwurfsentscheidungen für die Klasse, pro Attribut bzw. Methode entsprechend.
   - 3.3. Die Verwendung der Annotation *@SupressWarnings* ist nur zulässig, wenn Sie bei Verwendung dieser Annotation in einem javadoc-Kommentar überzeugend darlegen, warum die jeweilige Warnung gefahrlos unterdrückt werden kann!
   - 3.4. Die üblichen Java-Codekonventionen (Einrückung, Namensgebung etc.) müssen eingehalten werden!
4. Achten Sie bitte auf den genauen Text der Aufgaben und etwaige Hinweise!
5. Die Testfälle für die Aufgaben befinden sich im Paket *de.hawh.kahlbrandt.ss2018bai2pm2exam.test*.

Den Code schreiben Sie bitte im Wurzelpaket des Projekts: *de.hawh.kahlbrandt.ss2018bai2pm2exam*.

Viel Erfolg!

1 Datum, Uhrzeit, Zeiträume, Rechnen

Für diese Fragen braucht man keine extra Methode zu schreiben, man verwendet einfach die geeigneten aus der Java-Bibliothek. Sie müssen Ihren Code nur in jeweils eine Klassenmethode „einkapseln“, damit ich einfach testen kann.

Schreiben Sie bitte in einer Utility-Klasse *A01* für die folgenden Zwecke je eine Klassenmethode mit dem angegebenen Namen!

1. Schreiben Sie bitte eine Klassen-Methode *getAnniversary(LocalDate founding, int years)*, die für ein Datum und eine ganzzahlige positive Anzahl Jahre *years* das Datum des entsprechenden Jubiläums zurückliefert. Ich teste u. a. mit 50 Jahre HAW.

2. Schreiben Sie bitte eine Klassen-Methode *switchTimeZone(ZonedDateTime zdt, ZoneId zoneId)*, die ein Datum und eine Zeitzone als Parameter bekommt und das Datum in ein Datum der übergebenen Zeitzone konvertiert!

3. Schreiben Sie bitte eine Methode *long factorial(long n)*, die die Fakultät $n! = 1 \cdot 2 \cdot \cdots \cdot n$ einer nicht negativen ganzen Zahl vom Typ *long* sicher berechnet und zurückgibt sowie eine *ArithmeticException* mit einer entsprechenden Fehlermeldung wirft, wenn der Bereich von *long* verlassen wird oder $n < 0$ ist. Nach Definition ist $0! = 1$.

2 Generische Klasse schreiben

Schreiben Sie bitte eine generische Klasse *Stack*, die das vorgegebene Interface *IStack* implementiert!

Die Einträge im *Stack* verwalten Sie bitte über ein Array. Beim Erzeugen eines Objekts kann eine ganzzahlige Kapazität des Stacks angegeben werden. Wird keine Kapazität angegeben, so wird ein Stack mit der default-Kapazität von 11 angelegt. Wird eine nicht-positive Kapazität beim Erzeugen angegeben, so wird eine *NegativeArraySizeException* geworfen.

Die Klasse soll *serialisierbar* sein. Implementieren Sie dies bitte mit einer geeigneten *custom serialized form*!

Nun zu den Methoden aus dem Interface und der jeweiligen Fehlerbehandlung, die vielleicht einer weiteren Erläuterung bedürfen. Die Signaturen von *peek* und *pop* sind inkonsistent, aber ich wollte Sie beide Varianten üben lassen.

push Bekommt ein Element übergeben und legt es oben auf den Stack.

pop Entfernt das oberste Element aus dem Stack und liefert es zurück. Ist der Stack leer, so wird eine *RuntimeException EmptyStackException* geworfen.

peek Liefert ein Optional für das oberste Element des Stacks zurück. Ist der Stack leer, liefert peek ein leeres Optional.

capacity Liefert die aktuelle Kapazität des Stacks.

size Liefert die aktuelle Anzahl der Elemente auf dem Stack.

Zwei Stacks mit gleichem Typ-Parameter sind gleich, wenn Sie die selbe Größe haben (*size(), nicht* capacity()) und die gleichen Elemente in der gleichen Reihenfolge enthalten.

Dokumentieren Sie bitte alle Elemente mit Javadoc und denken Sie auch an Javadoc tags, wie @param, @throws, @serial etc.

3 Diverses

1. In der Datei *data/quotes.txt* finden Sie einige Zeilen mit „Sprüchen“. Ihre Aufgabe: Bitte schreiben Sie Code, der den „Spruch“ ermittelt, in dem das erste Vorkommen des unpersönlichen Reflexivums „sich“ am spätesten ist, d. h. an der höchsten Position unter allen Sprüchen. Groß- oder Kleinschreibung spielt dabei keine Rolle, Satzzeichen habe ich der Einfachheit halber bereits entfernt.

   Tun Sie das bitte in einer Klassenmethode in der Klasse *A03* mit dem Namen *adornoWinner*, die einen *Path* als Parameter bekommt und die Zeilennummer des Spruchs aus der Datei zurückliefert, bei dem „sich“ am spätesten zum ersten Mal steht.

   Kommt das Wort in keiner Zeile vor, geben Sie bitte -1 zurück.

2. Schreiben Sie bitte einen potentiell unbegrenzten *Stream*, der die im Folgenden definierten ganzen Zahlen ausgehend vom Startwert *start* liefert bis der Wert *1* erreicht ist:

$$c_n = \begin{cases} start, & \text{wenn } n = 0 \\ c_{n-1}/2, & \text{wenn } n > 0 \wedge c_{n-1} \text{ gerade} \\ 3 \cdot c_{n-1}, & \text{wenn } n > 0 \wedge c_{n-1} \text{ ungerade} \end{cases}$$

   Tun Sie das bitte in einer Klassenmethode in der Klasse *A03* mit dem Namen *collatz*, die eine positive ganze Zahl n vom Typ *int* erhält und liefern Sie die Länge des Streams zurück. Wird ein Parameter $n <= 0$ übergeben, so werfen Sie bitte eine *RuntimeException* mit einer aussagefähigen Fehlermeldung.