Klausur B-AI2 PM 2 SS 2017, Prof. Dr. Bernd Kahlbrandt

Matrikelnummer	Vorname	Name

Aufgabe	1.1	1.2	1.3	2	3					
Punkte (ca.)	3	4	4	15	6				32	
Erreicht										
Aufgabe	4.1	4.2	4.3	4.4	5				$\sum$	Note
Punkte (ca.)	3	2	2	4	7				50	
Erreicht										

Datum: **04.07.2017** Uhrzeit: **09:00**
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, Spezifikation der JVM, Skript, Folien (pdf), Abts Grundkurs Java, Goll Java als erste Programmiersprache (beides pdf).

2. Erlaubte „Mitbringsel“: Vorlesungmitschrift 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 Java-Codekonventionen müssen eingehalten werden!

4. Testfälle für die Aufgaben 2 und 5 finden Sie im Paket *de.hawh.kahlbrandt.exam.ss2017-bai2pm2.test*.

Den Code schreiben Sie bitte im Wurzelpaket des Projekts: *de.hawh.kahlbrandt.exam.ss2017-bai2pm2*.

Viel Erfolg!

1 Datum, Uhrzeit, Zeiträume

Falls Sie im letzten Wintersemester (also WS 2016/17) Ihr Studium begonnen haben, so begann Ihr Studium planmäßig am 01.09.2016. Die Regelstudienzeit beträgt für Vollzeitstudierende 6 Semester. Um diese Daten herum folgen einige Aufgaben, die Sie bitte mittels geeignetem Java-Code beantworten. Geben Sie die Ergebnisse bitte auch nach europäischem Standard formatiert auf der Konsole aus!

1. Wieviele Tage wird Ihr Studium gedauert haben, wenn Sie zum Ende der Regelstudienzeit abgeschlossen haben und zum Ende Ihres sechsten Semesters exmatrikuliert werden?

2. Fallen Sie bei einer Prüfung durch, so müssen Sie spätestens nach einem Jahr einen zweiten Versuch unternehmen. Für viele Prüfungen haben Sie insgesamt drei Versuche. Der Einfachheit halber nehmen wir an, die Prüfung sei jeweils am letzten Semestertag (31.08. bzw. am letzten Februartag). Bis wann müssen Sie eine Prüfung aus dem 3. Semester also bestanden haben?

3. Um wieviele Minuten verlängert sich Ihr Studium gegenüber der Regelstudienzeit, wenn Sie in die Situation aus 2. „hineinstolpern“ und die drei Versuche ausschöpfen müssen?

2 Generische Klasse schreiben

Schreiben Sie bitte eine generische Klasse *Queue*, die das vorgegebene Interface *IQueue* implementiert!

Die Einträge in der *Queue* verwalten Sie bitte über ein Array fester Länge. Beim Erzeugen eines Objekts kann eine ganzzahlige Kapazität der Queue angegeben werden. Wird keine Kapazität angegeben, so wird eine Queue mit der default-Kapazität von 42 angelegt. Wird eine nicht-positive Kapazität beim Erzeugen angegeben, so wird die (einzige) geeignete RuntimeException geworfen. Der erste Eintrag kommt auf Position 0 im Array, der nächste auf Position 1 usw. Ist das Ende des Arrays erreicht, so gibt es zwei Möglichkeiten:

1. Es wurden keine Elemente am Anfang entnommen (*dequeue*). Dann ist die Queue voll und beim nächsten *enqueue* wird eine RuntimeException „QueueFullException“ geworfen.

2. Andernfalls geht es vorne mit 0 weiter. Bis der (sich rollierend verschiebende) Platz doch aufgebraucht ist.

Nun zu den Methoden aus dem Interface und der jeweiligen Fehlerbehandlung, die vielleicht einer weiteren Erläuterung bedürfen:

enqueue Bekommt ein Element übergeben und fügt es am Ende der Queue ein. Ist die Queue bereits voll, so wird eine Runtime Exception „QueueFullException“ geworfen.

dequeue Entfernt das erste Element aus der Queue. Ist die Queue leer, so wird eine RuntimeException „QueueEmptyException“ geworfen.

peek Liefert ein Optional für das erste Element der Queue zurück. Ist die Queue leer, liefert peek ein leeres Optional.

Denken Sie bitte auch an etwaige weitere Methoden, die für diese Java-Klasse notwendig sein könnten! Begründen Sie bitte warum Sie weitere Methoden schreiben oder darauf verzichten.

Tipp: Es mag sinnvoll sein sich sowohl die Position zu merken, an der das nächste *enqueue* erfolgen soll, als auch die, an der das nächste *dequeue* bzw. *peek* zugreifen soll. Es kann einfacher sein zu begründen eine Methode nicht zu schreiben, als sie tatsächlich zu implementieren.

3 Unbegrenzte Streams

Ein sogenanntes compostelanisches *Heiliges Jahr* findet statt, wenn der *25.07.* auf einen Sonntag fällt. Das erste Heilige Jahr wurde im Jahr 1126 begangen. Definieren Sie bitte einen unbegrenzten Stream, der die Heiligen Jahre liefert! Ermitteln Sie bitte aus diesem Stream die Heiligen Jahre zwischen dem Jahr 2000 und dem Jahr 2100, jeweils einschließlich! Geben Sie das Ergebnis bitte als *List* zurück und anschließend auf der Konsole aus!

Welche verschiedenen Abstände gibt es zwischen diesen Jahren? Geben Sie bitte auch diese auf der Konsole aus.

4 Datei einlesen und verarbeiten

Wählen Sie bitte eine geeignete Datenstruktur als Ziel, um die folgenden Aufgaben zu lösen!

1. Verwenden Sie bitte einen Scanner, um die Datei *QueueTest.java* aus dem Testverzeichnis einzulesen.
2. Wieviele „import“ Statements enhält diese Datei?
3. Finden Sie bitte aus dem Source-Code heraus, wieviele Test-Methoden die Klasse hat!
4. Finden Sie bitte reflexiv heraus, welche und wieviele Test-Methoden die Klasse hat!

Geben Sie die Ergebnisse bitte auf der Konsole aus!

5 Und eine letzte Aufgabe: Vermischtes

Hier die Signatur einer Klassenmethode, z. B. aus einer Utility-Klasse:

```
public static int sum(int ...  coll)
```

die die Summe einer variablen Anzahl ganzer Zahlen zurückliefert. Schreiben Sie bitte in einer Utility-Klasse *A05* einige verschiedene Implementierungen $(3-5)$ dieser Methode! Werfen Sie bitte eine *ArithmeticException*, wenn es während der Rechnung zu einem Überlauf kommt! Wie Sie die Methoden nennen, ist egal soweit der Name den java Code-Konventionen entspricht. Ich teste reflexiv alle mit passender Signatur.

Die leere Summe ist 0.