Praktikum Programmieren
Studiengang Angewandte Informatik
Prof. Dr. Bernd Kahlbrandt
Dipl.-Ing. Gerhard Oelker
Dipl.-Wirtschaftsinformatiker (FH) Sven Berding
Hochschule für Angewandte Wissenschaften Hamburg
Department für Informatik
24. Mai 2020

Aufgabenblatt A08: Parallele, sequentielle Streams, IO

Prognosen sind schwierig, insbesondere, wenn sie von der Zukunft handeln, aber dies wird voraussichtlich das letzte Aufgabenblatt zu dieser Veranstaltung in diesem Semester sein.

Folgende Lernziele verfolge ich mit diesen Aufgaben:

- Den Umgang mit parallelen Streams (besser) beherrschen.
- Die Grenzen der Parallelisierbarkeit kennen.
- Konsequent mit *AutoCloseable* Resourcen umgehen können.
- Professionell mit Serialisierung umgehen können, wenn es denn sein muss.
- Weiter aktiv in der Java-Programmierung sein, um bis zu einer Prüfung nicht zu viel vergessen zu haben.

1 Java Denksportaufgabe: Grenzen der Parallelisierung

Was passiert hier bei Parallelisierung? Vergleichen Sie bitte für Parameter von *limit* bis maximal 14 (bei diesem Wert war meine Geduld zu Ende) die Laufzeiten für diese und die serielle Variante (z. B. . .limit auskommentieren). Sequentiell funktioniert auch für höhere Parameterwerte für *limit*. Versuchen Sie sich bitte an einer Erklärung! Mir reicht es schon, wenn Ihnen der Unterschied auffällt und Sie ihn artikulieren können.

```
import java.math.BigInteger;
import java.util.stream.Stream;

import static java.math.BigInteger.*;

/**
 * Parallel stream-based program to generate the first 20 Mersenne
 * primes - HANGS!!! (Page 222)
```

```
 * @author Joshua Bloch
 */
public class ParallelMersennePrimes {
    public static void main(String[] args) {
        primes().map(p -> TWO.pow(p.intValueExact()).subtract(ONE))
                .parallel()
                .filter(mersenne -> mersenne.isProbablePrime(50))
                .limit(20)
                .forEach(System.out::println);
    }

    static Stream<BigInteger> primes() {
        return Stream.iterate(TWO, BigInteger::nextProbablePrime);
    }
}
```

2 Parallele und Serielle Streams und IO

1. Im Foliensatz 1 zu Streams finden Sie ein Beispiel zu numerischer Integration. Schreiben Sie bitte zwei Methoden, die die Grenzen eines Intervalls, ein Schrittweite und eine Funktion übergeben bekommen und das Integral näherungsweise berechnen:

2. Eine Methode, die dies seriell erledigt!

3. Eine Methode, die dies parallel erledigt!

4. Vergleichen Sie bitte die Laufzeiten der beiden vorstehenden Methoden für einige Methoden und Schrittweiten (10^{-n}) für einige n.

5. Erzeugen Sie bitte durch Einlesen einer hinreichend großen Textdatei einen Stream aus Worten und generieren Sie daraus einmal jeweils ein *Array*, eine *ArrayList* eine *LinkedList*. Erzeugen Sie aus diesen „Collections“ bitte einmal einen seriellen und einmal einen parallelen Stream in dem Sie die Worte verarbeiten, z. B. Ermittlung der Wort-Häufigkeiten, und vergleichen Sie die Laufzeiten!

   Im folgenden geht es um ein Thema, das ich durchaus geneigt wäre Ihnen vorzuenthalten. Sollten Sie professionell mit Java programmieren, müssen Sie es aber beherrschen: Professionelles vermeiden der schlimmsten Tücken von Serialisierung. Die potenziellen Sicherheitsrisiken und die Möglichkeiten dagegen vorzubauen werde ich Ihnen nicht mehr vermitteln können. Aber die unter Software-Engineering Aspekten wichtigen Dinge können Sie üben. Sie finden die Kernpunkte im Skript und in der Java Collection-Library gute Vorbilder.

Sie sollten bisher zwei generische Klassen schreiben: Eine mit einem Array und eine mit einer statischen inneren Klasse. Von diesen sollen Sie jetzt eine weitere Version schreiben. Was Sie verändern müssen, ist Bestandteil der Aufgabenstellung! Schreiben Sie bitte von beiden Klassen zwei neue Versionen:

6. Mit einer sogannten *custom serialized form*: Sie müssen dazu Methoden implementieren, die in der API-Doc des Interfaces *Serializable* beschrieben werden.

7. Mit einem Serialization Proxy.

   Ich empfehle Ihnen zu diesen beiden Aufgaben die Erläuterungen, die ich dazu in der Vorlesung zu geben plane.

Abgabetermin für alle:

Montag, 08.06.2020, 08:00