Hausaufgabe 1: Auffrischung Prog I - I/O-Streams, Threads und Quicksort

Abgabe bis 16. Mai 2022, 23:59 Uhr
(maximal erreichbare Punktzahl: 20 Punkte)

Hinweise zur Bewertung

Im Rahmen der Lehrveranstaltung *Programmieren II* werden Hausaufgaben mit jeweils 20 Punkten ausgegeben. Um zur Modul-Abschlussklausur zugelassen zu werden, müssen Sie mindestens 90 der insgesamt 120 erreichbaren Punkte (75%) gesammelt haben. Sie müssen nicht in jeder einzelnen Hausaufgabe auf 75% kommen, es zählt nur die Gesamtpunktzahl aller Hausaufgaben.

Die Modulnote wird allein durch das Ergebnis der Klausur bestimmt. Die Hausaufgaben fließen nicht mit ein, sondern bilden das Zulassungskriterium für die Klausur.

Die Hausaufgaben werden auf der Grundlage von automatischen Tests bewertet. Ihnen wird mit jeder Hausaufgabe eine Auswahl der für die Bewertung genutzten Tests zur Verfügung gestellt. Sie können diese nutzen, um schon bei der Bearbeitung die Richtigkeit Ihrer Lösung überprüfen zu können.

Achtung: Entwickeln Sie Ihre Lösungen nicht "nur für die Tests"! Die von uns zur Bewertung genutzten Tests werden in der Regel mit anderen Parametern ausgeführt als die Ihnen zur Verfügung gestellten. Vermeiden Sie daher hartkodierte Ausgaben, die exakt auf die veröffentlichten Tests passen.

Beispiel: Sie sollen eine Methode implementieren, die zwei Argumente `a` und `b` addiert. Unser veröffentlichter Test prüft diese Methode mit den Parametern `a=2` und `b=3`, mit dem erwarteten Ergebnis `5`. Sollten Sie hier Ihre Methode einfach mit `return 5` statt `return a + b` implementieren, zeigt Ihnen der Test zwar an, dass Ihre Lösung richtig ist, in unserer Bewertung werden Sie aber 0 Punkten erhalten, da wir andere Werte für `a` und `b` einsetzen.

Vorbereitung und Software

Im Rahmen der Hausaufgaben verwenden wir in diesem Modul:

- Java Development Kit version 18[1]
- IntelliJ IDEA[2] als Entwicklungsumgebung
- Das Build-Tool Gradle[3]
- Scala[4]
- Das Scala-Plugin[5] für IntelliJ

Wir empfehlen als Entwicklungsumgebung die aktuellste Version der kostenlosen IntelliJ IDEA Community Edition. Es steht Ihnen frei, eine andere IDE zu verwenden, wir beschränken unseren Support aber auf IntelliJ. Wir können nicht auf Konfigurationsschwierigkeiten und Inkompatibilitäten mit der automatischen Bewertung Rücksicht nehmen, die durch die Verwendung einer anderen Entwicklugnsumgebung entstehen.

Sie benötigen für dieses Modul mindestens Java Version 8, da die behandelten Konzepte mit dieser Version eingeführt wurden. **Wir empfehlen Ihnen jedoch, die derzeit aktuellste Version, Java 18, zu nutzen.** Sie können sich das JDK entweder manuell herunterladen und installieren oder es über die Projektkonfiguration in IntelliJ automatisch einbinden lassen. Sie werden dafür beim ersten Öffnen der Vorgabe aufgefordert, ein JDK für das Projekt zu spezifizieren.

Eine Installationsanleitung für Scala und das Scala-Plugin für IntelliJ finden Sie in unserem ISIS-Kurs. Das Gradle Build-Tool ist bereits in IntelliJ integriert.

Um die Hausaufgaben zu bearbeiten, laden Sie sich die jeweilige Vorgabe herunter, entpacken Sie das ZIP-Archiv in einen leeren Ordner und öffnen Sie den Ordner als IntelliJ-Projekt. Der Gradle Synchronisierungsprozess sollte automatisch starten und alle notwendigen Dependencies automatisch herunterladen und konfigurieren. Ggf. werden Sie aufgefordert, ein JDK für das Projekt auszuwählen bzw. herunterzuladen.

Verändern Sie ausschließlich die Klassen im `impl` Package! Für die Bewertung werden diese Dateien in eine gesonderte Ausführungsumgebung geladen. Sämtliche Änderung, die Sie an Dateien außerhalb des `impl` Packages vornehmen, werden dabei verworfen.

Als Abgabe laden Sie Ihre Lösung als ZIP-Archiv bei ISIS hoch. Achten Sie darauf, dass Ihre Abgabe exakt die gleiche Ordner- und Dateistruktur wie die Vorgabe hat! Abgaben im falschen Format können ggf. nicht bewertet werden!

[1] `https://jdk.java.net/18/`
[2] `https://www.jetbrains.com/de-de/idea/download/`
[3] `https://gradle.org/`
[4] `https://www.scala-lang.org/`
[5] `https://plugins.jetbrains.com/plugin/1347-scala`

1.1 Atomare Inkrementierung (7 Punkte)

Ihnen ist die Klasse `AtomicIncrementer` gegeben, welche das Interface `util.Incrementer` implementiert. Diese Klasse soll einen Zähler implementieren, welcher mittels der vom Interface vorgegebenen Methoden ausgelesen, inkrementiert und zurückgesetzt werden kann. Dabei soll gewährleistet sein, dass beim gleichzeitigen Zugriff durch mehrere parallele Threads keine unerwünschten Effekte, wie z.B. Race Conditions, auftreten.

- Ergänzen Sie zunächst die fehlenden Methoden, die vom Interface definiert werden.
- Implementieren Sie in der Dokumentation der Interface-Methoden angegebene Funktionalität. Sie dürfen die `AtomicIncrementer`-Klasse hierfür beliebig um Variablen und Methoden ergänzen, solange die Signaturen des Konstruktors und der Interface-Methoden unverändert bleiben.
- Ihre Implementierung soll **Thread-Sicherheit** gewährleisten. Sorgen Sie daher dafür, dass alle Operationen (Methoden) **atomar** sind. Wie Sie dies erreichen, ist Ihnen freigestellt, solange die Signaturen des Konstruktors und der Interface-Methoden unverändert bleiben.

Sie können Ihre Implementierung mithilfe der in der Testklasse `TestAtomicIncrementer` gegebenen Tests überprüfen.

1.2 InputStreams und Multithreaded Quicksort (13 Punkte)

Im Projektunterordner `resources` finden Sie die Datei `numbers.txt`, welche eine Sammlung von Zahlen enthält. Ziel dieser Aufgabe ist es, dass Sie diese Datei auslesen, die enthaltenen Zahlen in ein Integer-Array übertragen und das resultierende Array mithilfe des Ihnen aus *Programmieren I* bekannten Quicksort-Algorithmus sortieren. Ihre Quicksort-Implementierung soll dabei allerdings nicht einfach rekursiv sein, sondern jeden Rekursionsschritt an frische Threads übergeben.

Ihnen sind dazu die Klassen `Main`, `FileInput` und `QuicksortRunnable` gegeben. Verändern Sie für Ihre Lösung ausschließlich diese drei Klassen!

1.2.1 Main.java (3 Punkte)

- Verändern Sie ausschließlich die Methode `readAndSort()`! Die `main`-Methode ist vorgegeben und ruft `readAndSort()` auf. Bei der Bewertung wird die `main`-Methode ignoriert, da `readAndSort()` direkt von den Tests aufgerufen wird.
- Lesen Sie das Integer-Array aus der gegebenen Datei aus, indem sie die Methode `FileInput.readIntsFromFile()` mit dem übergebenen Dateipfad aufrufen.
- Sortieren Sie das Array, indem Sie einen `Thread` starten, der ein `QuicksortRunnable` mit dem Array übergeben bekommt.
- Warten Sie darauf, dass der gestartete `Thread` fertig wird und geben Sie anschließend das sortierte Array zurück.

1.2.2 FileInput.java (4 Punkte)

- Implementieren Sie die Methode `readIntsFromFile()`.
- Diese Methode erhält als Argument den Pfad zu einer Datei, welche sich im Ordner `resources` befindet. Solche Ressourcen können mittels der Methode `ClassLoader.getResourceAsStream()`[6] gelesen werden.
- Lesen Sie die Datei aus und schreiben Sie den Inhalt in ein Integer-Array (`int[]`). Sie können davon ausgehen, dass jede Zeile der Datei exakt einen Integer enthält und nicht leer ist.
- Geben sie das resultierende Array zurück.
- Wird ein ungültiger Dateipfad übergeben, so soll ein leeres Integer-Array zurückgegeben werden.

1.2.3 QuicksortRunnable.java (6 Punkte)

- Implementieren Sie die Methode `run()` der Klasse `QuicksortRunnable`. Verändern Sie dabei weder die Signatur der Klasse noch die des Konstruktors.
- Die Superklasse `util.Monitor` wird von der automatischen Bewertung verwendet, um Einblicke in die Ausführung Ihres `Runnables` zu gewinnen. Sie hat keinerlei Auswirkungen auf Ihre Implementierung und kann ignoriert werden.
- Das `QuicksortRunnable` soll, wie sein Name verrät, den rekursiven Quicksort-Algorithmus implementieren. Statt jedoch die Rekursion klassisch durch Aufruf der selben Methode zu erreichen, soll jeder Rekursionsschritt einen neuen `Thread` ins Leben rufen, der wiederum rekursiv neue `Threads` startet.
- Zur Veranschaulichung können Sie folgenden Pseucode betrachten:

```
funktion quicksort(Partition (liste, links, rechts))
  falls links < rechts dann
    teiler := teile(links, rechts)
    starte neuen Thread mit QuicksortRunnable (Partition (liste, links, teiler - 1))
    starte neuen Thread mit QuicksortRunnable (Partition (liste, teiler + 1, rechts))
  ende
ende
```

- Dem `QuicksortRunnable` wird ein Objekt vom Typ `Partition` übergeben, welches alle für einen Rekursionsschritt notwendigen Informationen enthält.

Sie können Ihre Lösung mithilfe der in der Testklasse `TestReadAndSort` gegebenen Tests überprüfen.

[6] `https://docs.oracle.com/javase/8/docs/api/java/lang/ClassLoader.html#getResourceAsStream-java.lang.String-`