Übungsblatt 1: Threads in Java

Vorstellung in den Tutorien am 29. und 30. April 2019

Themen:

- Mandelbrot-Menge
- for-each in Java
- CodeConventions
- Unterschied Nebenläufigkeit und Parallelität
- Threads und Threadzustände

Laden Sie die Vorlage zu diesem Blatt bei ISIS herunter, dann importieren Sie das Projekt in Ihre Entwicklungsumgebung.

1.1 Mandelbrot-Menge (4 Punkte)

In dieser Aufgabe berechnen wir eine Visualisierung der Mandelbrot-Menge[1], benannt nach dem französisch-US-amerikanischen Mathematiker Benoît Mandelbrot. Die Darstellung, bekannt als „Apfelmännchen“, ist ein sogenanntes Fraktal und hat die erstaunliche Eigenschaft, dass man in den Randbereich der Figur praktisch unendlich weit hineinzoomen kann und immerzu neue komplexe geometrische Figuren sichtbar werden. Ihre Aufgabe besteht darin, das vorliegende Programm zu parallelisieren.

Die Vorlage zur Aufgabe befindet sich im Ordner

`src/main/java/mandelbrot/Mandelbrot.java`

und kann wie folgt gestartet werden:

- **IntelliJ IDEA**: Wählen Sie *Run -> Run*, im sich öffnenden Kontextmenü wählen Sie `Mandelbrot`. Es wurde eine *Run Configration* erzeugt (siehe Drop-Down Box oben rechts). In Zukunft können sie also einfach den „Play“-Button oben rechts klicken.
- **Maven**: Sie können zum Starten auch *Maven* auf der Shell verwenden. Wechseln Sie auf der Kommandozeile in das Projektverzeichnis (mit `pom.xml`) und führen Sie

  ```
  mvn compile exec:java
  ```

  aus.

[1] `https://de.wikipedia.org/wiki/Mandelbrot-Menge`

Vergewissern Sie sich, dass ein 1920x1080 Pixel groSSes Bild namens `mandelbrot.png` im Projektverzeichnis erzeugt wurde, sowie dass das Programm

```
Running on 1 Thread/s
Took 1337 milliseconds
```

auf der Shell ausgegeben hat. Auf Ihrer Maschine kann die Berechnungszeit natürlich variieren.

Aufgabe: Frischen Sie Ihre Kenntnisse über Threads in Java aus der Veranstaltung *Prog1* auf und erweitern Sie die Vorlage:

1. Parallelisieren Sie das Programm mithilfe von Threads!
2. Die Anzahl der Threads soll als erstes Kommandozeilenargument übergeben werden. Achten Sie beim Implementieren der Entgegennahme der Argumente von der Kommandozeile darauf, dass Sie etwaige Fehler abfangen! Falls Sie Maven zum Starten verwenden, können Sie die Kommandozeilenargumente in der Datei `pom.xml` bequem setzen.

   ```
   <configuration>
       <executable>maven</executable>
       <mainClass>mandelbrot.Mandelbrot</mainClass>
       <arguments>
           <argument>HIER</argument>
       </arguments>
   </configuration>
   ```

   (natürlich ein Integer, statt dem „HIER!“). Falls Sie das Programm aus Ihrer Entwicklungsumgebung heraus starten, müssen Sie ihre *Run Configuration* anpassen (*Program Arguments*). Bei **IntelliJ IDEA** befindet sich die Option unter *Run -> Edit Configurations* und funktioniert analog.

Hinweise:

- Parallelisieren Sie nur die Berechnung der Menge selbst, nicht das Übertragen in das `BufferedImage` durch die Methode `render` oder das Erzeugen der Threads selbst (teuer)!
- Die Methoden `pixel`, `mandelbrot` und `render` müssen für die Lösung nicht verändert werden.

Test: Testen Sie nun die Ausführungszeit mit $1, 2, 4, 8, 16, 32, 64$ Threads jeweils etwa dreimal mit derselben Konfiguration. Notieren Sie den Mittelwert, sobald sich die Ergebnisse „eingependelt“ haben. Diskutieren Sie im Tutorium:

- Was fällt Ihnen bezüglich der Anzahl der Threads, der Berechnungszeit und der Anzahl der Kerne in Ihrem Rechner auf?
- Warum ist dies eine vergleichsweise „angenehme“ Parallelisierungsaufgabe?

1.2 *Foreach*-Schleife (1 Punkte)

Informieren Sie sich über die foreach-Schleife in Java. Wann und wie wird sie angewendet? (siehe auch: Homepage von Oracle[2])

Im Verzeichnis `src/main/java/foreach/ForEach.java` der Vorlage befinden sich mehrere Klassenmethoden der Form:

```
static void forLoop1A(String[] names) {
    for(int i = 0; i < names.length; i++) {
        System.out.println(names[i]);
    }
}

static void forLoop1B(String[] names) {
    // TODO implement me!
}
```

Gegeben ist also eine normale for-Schleife bzw. eine foreach-Schleife. Geben Sie jeweils den Code unter Verwendung der anderen Schleife an!

Test: Testen Sie einmal, bevor Sie die Aufgabe lösen. Sie werden sehen, dass alle Tests negativ verlaufen. Starten Sie die Tests erneut, nachdem Sie die Aufgabe gelöst haben, sollten alle Tests positiv verlaufen.

- **IntelliJ IDEA**: Wählen Sie *Run -> Run -> ForeachTest*
- **Maven**: Von der Kommandozeile verwenden Sie zum Testen das Kommando:

  ```
  mvn -DskipTests=false test
  ```

Abgabe

Erzeugen Sie nun Ihre Abgabe mit *Maven*. Öffnen Sie die Datei `pom.xml` und ändern Sie den Eintrag `<version>MISSING-TUTORIAL-NUMBER</version>` auf `T` gefolgt von der Nummer des Tutoriums, an dem Sie teilnehmen und Ihrem Namen. Falls Sie am Tutorium 8 teilnehmen also in den Wert `<version>T08-MaxMustermann</version>`. Danach lassen Sie das *Maven* goal `package` laufen. Dies erzeugt normalerweise ein Java Archiv (`.jar` File) mit dem kompilierten Programm im Ordner `target`. In unserem Fall ist *Maven* so konfiguriert, dass auch der Quelltext Ihrer Abgabe in ein separates Archiv verpackt wird:

- **IntelliJ IDEA**: Drücken Sie zweimal kurz die Shift-Taste („Search everywhere“). Geben Sie in das Textfeld `Maven` ein, warten Sie kurz und wählen Sie unter `Tool Window` den Punkt `Maven Projects`. In dem erscheinenden Fenster wählen Sie das sechste Icon von links (*Execute Maven Goal*), dort wählen Sie aus der Drop-Down Liste `package`.

[2] `http://docs.oracle.com/javase/1.5.0/docs/guide/language/foreach.html`

Wenn der Build also erfolgreich war, befindet sich im Unterverzeichnis `target` nun ein File mit dem Namen `assignment01-TXX-MaxMustermann-src.zip` (wobei `TXX` dann die Nummer Ihres Tutoriums sein sollte). Vergewissern Sie sich, dass das `.zip`-File Ihre Lösung für 1.1 und 1.2 enthält. Laden Sie das File dann bei ISIS hoch.

Hinweis: Wenn Ihr Programm nicht kompiliert, dann wird *Maven* das Archiv nicht erzeugen. Geben Sie nur Lösungen ab, die auch kompilieren!

Alternativ: Sollte das Erzeugen des Abgabe-ZIP-Archivs über `mvn package` nicht funktionieren, erzeugen Sie bitte das ZIP-Archiv über Ihre Entwicklungsumgebung:

- **IntelliJ IDEA**: *File -> Export to Zip File...*

Bitte beachten Sie das Schema für die Benennung des ZIP-Archivs (s.o.)!

Java CodeConventions (Tut)

Welche CodeConventions kennen Sie in Java? Wozu sollte man diese beachten? (siehe auch: Homepage von Oracle[3])

Tragen Sie die wichtigsten CodeConventions für Java zusammen. Geben Sie ein einfaches Beispiel für die Verwendung von JavaDoc an.

Nebenläufigkeit vs. Parallelität (Tut)

Was ist Nebenläufigkeit und was ist Parallelität? Definieren Sie die beiden Begriffe und machen Sie die Zusammenhänge zwischen ihnen klar.

Threads theoretisch (Tut)

Threads Basics Grenzen Sie folgende Begriffe und ihre Beziehungen zueinander ab: Namensraum, kritischer Abschnitt/Bereich, gemeinsam genutzte Daten, immutable Objekte, `synchronized`.

Zustände Wie ist der Thread-Lebenszyklus aufgebaut? Welche unterschiedlichen Zustände eines Threads haben Sie kennen gelernt und welche Eigenschaften haben die jeweiligen Zustände?

[3] `http://www.oracle.com/technetwork/java/codeconvtoc-136057.html`