Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Institut für Programmstrukturen und Datenorganisation (IPD)

Prof. Dr. Walter F. Tichy

Vorlesung Softwaretechnik 1

Übungsblatt 5
Sommersemester 2009

Ausgabe: 18.06.2009

Abgabe: 02.07.2009, 11:00 Uhr

Entwurfsmuster und Parallelität

Aufgabe 1: Entwurfsmuster in Java IO (6 Punkte)

Betrachten Sie die Schnittstellendokumentation der Klasse **`BufferedOutputStream`** aus dem Paket **`java.io`** (http://java.sun.com/javase/6/docs/api/java/io/BufferedOutputStream.html).

a) Zeichnen Sie die komplette Vererbungshierarchie der Klasse **`BufferedOutputStream`** als UML-Klassendiagramm. Die Klasse **`Object`** können Sie hierbei weglassen. Geben Sie in Ihrem UML-Klassendiagramm öffentliche Methoden immer dann an, wenn sie zum ersten Mal deklariert werden oder (neu) implementiert werden. Eventuell geworfene Ausnahmen können Sie vernachlässigen. Attribute (Fields), Konstruktoren und nicht öffentliche Methoden brauchen Sie ebenfalls nicht einzuzeichnen. (3 Punkte)

b) Welches Entwurfsmuster ist in dem UML-Klassendiagramm zu finden? Kennzeichnen alle zum Entwurfsmuster gehörenden Klassen/Schnittstellen im UML-Klassendiagramm. Weisen Sie diesen die entsprechenden Rollen aus dem Entwurfsmuster zu. Ordnen Sie den Methoden im UML-Klassendiagramm – soweit möglich – ihre Entsprechungen im Entwurfsmuster zu. (3 Punkte)

Aufgabe 2: Gebietszerlegung und Synchronisation (8 Punkte)

a) In der Vorlesung wurde besprochen, wie Matrizen und Vektoren für die Parallelisierung von Algorithmen aufgeteilt werden können. Der folgende Ausschnitt stammt aus dem Quelltext zur parallelen Multiplikation einer Matrix **`a`** mit einem Vektor **`b`** unter Verwendung von **`t`** Fäden. Dabei wird die Matrix zeilenweise in mehrere Blöcke der Größe **`s`** aufgeteilt.

```
int s = (int) Math.ceil((double) a.length / t);
for (int i = 0; i < t; i++) {
  start = i * s;
  end = Math.min((i + 1) * s, a.length);
  // ... Arbeiter erstellen und Threads starten
}
```

Angenommen, die Berechnung der Blöcke würde wie folgt stattfinden:

```
int s = a.length / t;
int r = a.length % t; // Rest bestimmen
for (int i = 0; i < t; i++) {
  start = i * s;
  end = (i + 1) * s;
  if (i == t - 1) {
    // Letzter Faden übernimmt
    // zusätzlich die übrigen Elemente
    end += r;
  }
  // ... Arbeiter erstellen und Threads starten
}
```

Wieso ist diese alternative Implementierung zur Berechnung der Blockgrößen nicht geeignet? Begründen Sie Ihre Aussage und geben Sie ein nicht triviales Beispiel an, welches Ihre Aussage belegt (Dimensionen der Matrix `a`, Anzahl der Fäden `t`, Größe der Blöcke `s` für jeden Faden, den Rest `r`, sowie den Anfang `start` und das Ende `end` jedes Blockes `i`). Berechnen Sie für dieselben Werte `a` und `t` die Werte `s`, `r`, `start` und `end` nach dem in der Vorlesung gezeigten Algorithmus zur Gebietszerlegung (siehe erster Quelltextausschnitt dieser Aufgabe). Unter welchen Annahmen führt auch dieser Algorithmus zu einer ungünstigen Aufteilung? (4 Punkte)

b) Betrachten Sie die folgende Implementierung einer Barriere:

```
public class Barrier {

  final int cnt;    // Anzahl teilnehmender Fäden
  int arrived = 0;  // Anzahl der angekommenen Fäden

  public synchronized void sync() throws InterruptedException {
    arrived++;
    if (arrived == cnt) {
      arrived = 0;
      notifyAll();
    } else {
      while (arrived > 0) {
        wait();
      }
    }
  }
}
```

Erklären Sie in wenigen Sätzen, warum diese Barriere nicht korrekt funktioniert. Geben Sie dazu ein möglichst einfaches Beispiel an, welches das Problem aufdeckt. Beschreiben Sie, wie das Problem behoben werden kann und passen Sie den Quelltext so an, dass die Barriere korrekt funktioniert. (4 Punkte)

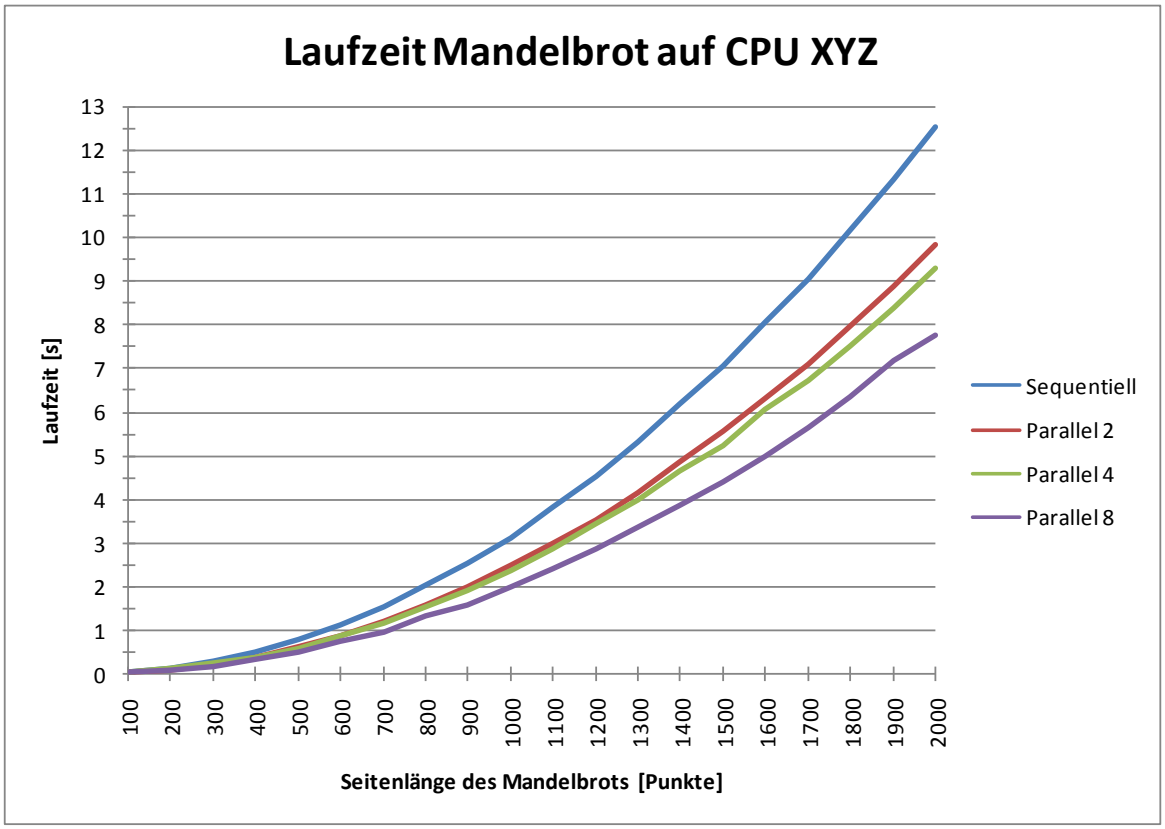
Aufgabe 3: Mandelbrot (16 Punkte)

In der Datei Mandelbrot.zip finden Sie das komplette Eclipse-Projekt eines Programms zur Berechnung der Mandelbrot-Menge. In der Klasse `mandelbrot.Start` sind fünf Parameter (siehe `@Option` Annotation) definiert und dokumentiert. Lesen Sie sich genau durch, was diese tun bzw. später tun sollen. Die Deklarationen dieser Parameter verletzen die Checkstyle-Regeln. Darüber brauchen Sie sich bei der Abgabe keine Gedanken zu machen.

a) Die zur Verfügung gestellte Implementierung verwendet das Entwurfsmuster Strategie. Lokalisieren Sie die relevanten Klassen/Schnittstellen und weisen Sie ihnen die entsprechenden Rollen aus dem Entwurfsmuster zu. Ordnen Sie den Methoden im Quelltext – soweit möglich – ihre Entsprechungen im Entwurfsmuster zu. (3 Punkte)

b) Erweitern Sie das Programm derart, dass es möglich ist, die Mandelbrot-Menge parallel zu berechnen. Die parallele Berechnung der Mandelbrot-Menge soll ausgeführt werden, wenn das Programm mit dem Parameter `--parallel` gestartet wird und die Anzahl der Threads (`--threads`) größer eins ist. In allen anderen Fällen soll die Berechnung der Mandelbrot-Menge weiterhin sequentiell erfolgen. Das Einfärben des Bildes der Mandelbrot-Menge müssen Sie nicht parallelisieren. (9 Punkte)

c) Führen Sie Laufzeitmessungen sowohl für den sequentiellen, als auch für den parallelen Algorithmus mit Seitenlängen von i × 100, i = 1,2,3, …, 20 durch. Sorgen Sie für eine möglichst genaue Messung durch das Anhalten nicht benötigter Prozesse. Messen Sie im parallelen Fall für 2^n, n = 1,2,3, ..., 9 Threads. Teilen Sie die neun Messreihen zum parallelen Algorithmus in drei Blöcke mit jeweils drei aufeinanderfolgenden Messreihen ein. Stellen Sie einen Block mit drei der parallelen Messreihen immer zusammen mit der sequentiellen Messreihe in jeweils einem eigenen Diagramm dar. Auf der x-Achse tragen Sie die Seitenlängen auf, auf der y-Achse die Laufzeiten in Sekunden. Beschriften Sie alle Achsen geben Sie einen Titel an und fügen Sie eine geeignete Legende ein. Achten Sie darauf, dass der gezeigte Bereich der Laufzeiten in allen Diagrammen gleich bleibt. Sie erhalten auf diese Weise drei Diagramme, die in etwa wie folgt aussehen:



Stellen Sie in drei weiteren Diagrammen die entsprechenden Kurven für die Beschleunigung (T(1)/T(p)) dar. Geben Sie zudem folgende Kennzahlen des verwendeten Rechners an: Prozessorname, Anzahl der installierten Prozessoren, Taktfrequenz, Anzahl und Größen der Prozessor-Caches sowie die Größe des Hauptspeichers. (4 Punkte)

Hinweis: **Bitte lesen Sie sich den folgenden Text aufmerksam durch. Missachtung der Anweisungen wird Sie Punkte kosten!**

Die Lösung geben Sie bitte nicht nur als E-Mail sondern auch über den Praktomaten ab. Die Anmeldung am Praktomaten erfolgt unter folgender Adresse:

https://swt1.ipd.uka.de/

Die Oberfläche zum Einreichen der Lösungen finden Sie unter

https://swt1.ipd.uka.de/praktomat/

Den Quelltext Ihres Programmes drucken Sie bitte aus und geben ihn zusammen mit dem Übungsblatt ab. Zusätzlich müssen Sie sowohl den Quelltext als auch die Java-Anwendung an folgende E-Post-Adresse senden:

swt1@ipd.uka.de

Als Betreff geben Sie bitte **[Uebungsblatt 5] <TUTNR> <MATRNR> <VORNAME> <NACHNAME>** an.

Hängen Sie den Java-Quelltext als ZIP-Archiv an die E-Mail an. Der Name des ZIP-Archivs muss folgendes Format haben: **<TUTNR>_<MATRNR>_<VORNAME>_<NACHNAME>.zip**. Das ZIP-Archiv darf nur den Java-Quelltext (d.h. Dateien mit der Endung .java) und evtl. benötigte Ressourcen (wie bspw. Icons) enthalten. Sofern Sie mit Paketen gearbeitet haben, muss diese Ordnerstruktur ebenfalls in Ihrem ZIP-Archiv vorhanden sein.

Verwenden Sie in Ihrem Java-Quelltext nur ASCII-Zeichen und vermeiden Sie IDE-spezifische Java-Annotationen. Die lauffähige Java-Anwendung hängen Sie als **ausführbares JAR-Archiv** zusätzlich zum ZIP-Archiv an die E-Mail an. Es werden nur Java-Anwendungen akzeptiert, welche als ausführbares JAR-Archiv an die E-Mail angehängt sind! Das JAR-Archiv muss als separater Anhang gesendet werden. Der Name des JAR-Archivs muss wie folgt lauten: **<TUTNR>_<MATRNR>_<VORNAME>_<NACHNAME>.jar**

Beispiel: Sie heißen Hans Müller, gehen in das Tutorium 7 und Ihre Matrikelnummer ist 123456. Ihr Betreff sieht dann wie folgt aus:

[Uebungsblatt 5] 07 123456 Hans Mueller

Die Anhänge heißen dann:

07_123456_Hans_Mueller.zip / 07_123456_Hans_Mueller.jar

Die E-Mail mit beiden Anhängen muss bis zum offiziellen Abgabetermin bei uns eingegangen sein. Vergewissern Sie sich, dass Sie alle Code-Dateien, die zur Übersetzung der Java-Anwendung notwendig sind, mitschicken und der Quelltext übersetzt werden kann. Prüfen Sie, dass die Java-Anwendung aus dem JAR-Archiv lauffähig ist. Andere Dateiformate werden nicht akzeptiert.

Bei Missachtung der obigen Anforderungen oder Teilen davon wird die gesamte Aufgabe mit 0 Punkte bewertet.