technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 11

Jonas Ellert
Mart Hagedoorn
Guangping Li

Sommersemester 2022
8. April

DAP2 Praktikum – Blatt R
Regeln und Hinweise

Abgabe: —

Die Regeln und Hinweise können sich im Laufe des Semesters ändern. Die aktuelle Version ist immer im Moodle verfügbar!

0.1 Arbeitsumgebung einrichten

Verwendung von Pool-Rechnern und Privatgeräten. An den Rechnern in den Pool-Räumen der Fakultät können Sie sich mit Ihrem IRB-Account einloggen. Dort haben Sie ihre eigene Arbeitsumgebung, die geräteübergreifend immer für Sie zur Verfügung steht. Es empfiehlt sich, für jede Aufgabe oder jedes Aufgabenblatt (je nach Umfang) ein eigenes Arbeitsverzeichnis anzulegen, z. B. mit:

```
mkdir -p ~/dap2praktikum/aufgaben/blatt01
```

Mit dem folgenden Befehl wechseln Sie in dieses Verzeichnis:

```
cd ~/dap2praktikum/aufgaben/blatt01
```

Auf den Pool-Rechnern ist ein Linux-Betriebssystem installiert, und es steht Java in Version 11 zur Verfügung. **Wir empfehlen Ihnen dringlichst, auch die Bearbeitung der Langaufgaben unter Linux und mit der gleichen Java-Version vorzunehmen.** Dazu haben Sie mehrere Möglichkeiten: Installation einer gängigen Linux-Distribution (z.B. Ubuntu) auf einem Privatgerät; Verwendung des Windows Subsystem for Linux (WSL) auf einem Privatgerät mit Windows; Verwendung der Rechner in den Pool-Räumen außerhalb der Praktikumszeiten.

Verwendung von IDEs. Für das Praktikum wird es ausreichend sein, die Aufgaben mit einem Texteditor und dem Kommandozeilen-Compiler/Interpreter zu lösen (weitere Hinweise dazu stehen unten). Es ist Ihnen zwar erlaubt, eine beliebige IDE (BlueJ, Eclipse, Visual Studio Code, etc.) zu verwenden, allerdings sind Sie dann selbst dafür verantwortlich, diese einzurichten und sicherzustellen, dass diese auch funktioniert.

Die Tutoren helfen ihnen NICHT bei Problemen, die durch die Verwendung von IDEs oder Windows entstehen. Die Testierung der Aufgaben erfolgt auf den Pool-Rechnern. Wenn Ihr Lösung nur auf Ihrem Privatgerät funktioniert, können wir Ihnen keine Punkte geben.

0.2 Kommentare und Codestil

Auf diesem und auf allen folgenden Aufgabenblättern werden bis zu zwei Punkte für die sinnvolle Kommentierung des Quelltextes vergeben. Diese Punkte können folglich nicht ohne Kommentierung erreicht werden. Kommentare sollen das Programm erläutern und nicht einfach nur den Quelltext beschreiben, wie folgendes Beispiel demonstriert:

```
// Beispiel eines NICHT SINNVOLLEN Kommentars:
// Hier wird geprueft, ob a gleich 0 ist
if (a != 0) {
```

Besser ist es, zu erklären, *warum* `a` mit Null verglichen werden muss:

```
// Beispiel eines SINNVOLLEN Kommentars:
// a darf nicht 0 sein, da sonst spaeter durch 0 dividiert wuerde
if (a != 0) {
```

Richten Sie sich nach folgender *Faustregel*:
Stellen Sie sich vor, Sie möchten jemandem mit Ihren Programmierfähigkeiten Ihr Programm erklären. Ihr Gegenüber soll das Programm schnell verstehen können und davon überzeugt werden, dass es *korrekt* ist. Schreiben Sie dazu alle nötigen Hinweise als Kommentare in Ihren Quelltext. Insbesondere *müssen* sinnvolle Schleifeninvarianten als Kommentar angegeben werden. Mithilfe der Kommentare sollten Sie selbst in Monaten noch dazu in der Lage sein, Ihren eigenen Code sofort zu verstehen.

Neben guter Kommentierung führt auch ein guter Codestil zu einem leichteren Verständnis des Codes. In Java haben sich zum Beispiel Konventionen wie CamelCase-Notation durchgesetzt. Ein weiteres Beispiel für guten Codestil ist es, isolierte Berechnungen (z. B. *ggT berechnen*) in eigene Methoden auszulagern. Dadurch steigern Sie unter anderem die Wiederverwendbarkeit und Lesbarkeit Ihres Codes. Es gibt verschiedenste Java-Codestile[1][2][3], die alle in wesentlichen Teilen übereinstimmen. Sollten Sie mit einer IDE arbeiten, ist es empfehlenswert die Autoformatieren-Funktion der IDE zu verwenden. Für viele Texteditoren gibt es Plugins, welche die automatische Formatierung ermöglichen.

Wenn Ihr Tutor nicht dazu in der Lage ist, Ihren Quelltext ohne große Mühe zu verstehen, kann dies negativen Einfluss auf die Bewertung haben.

Hinweis: **Richtiges Einrücken ist eine elementare Voraussetzung!**

0.3 Ausgaben, Fehlermeldungen, Exceptions und Assertions

Ihr Programm sollte immer eine leicht verständliche und bedeutungsvolle Ausgabe erzeugen (also z.B. "`Das Maximum der eingegebenen Werte ist 15`", und nicht einfach nur "`15`"). Außerdem sollten Sie immer mit einer sinnvollen Fehlermeldung auf ungültige Eingaben reagieren. Exceptions müssen abgefangen und behandelt werden. In vielen Fällen macht es Sinn, das Programm mit entsprechenden Assertions auf Korrektheit zu überprüfen.

Wenn Sie sich nicht an diese Vorgaben halten, kommt es zu Punktabzügen!

[1] https://www.oracle.com/technetwork/java/codeconventions-150003.pdf
[2] https://google.github.io/styleguide/javaguide.html
[3] https://www.gnu.org/software/gnu-crypto/code-formatting.html

0.4 Bibliotheken

Sie dürfen **keine Bibliotheken** (weder extern noch Java-Standard) verwenden. Auf jedem Blatt wird explizit angegeben, welche Standard-Klassen oder Bibliotheken erlaubt sind. Sinn der Regelung ist es, dass Sie die Aufgabenstellung nicht aktiv umgehen. Wenn Sie z.B. einen Sortieralgorithmus implementieren sollen, und stattdessen `Collections.sort()` verwenden, so führt dies zu 0 Punkten.

Die Verwendung von nicht explizit erlaubten Bibliotheken kann zu massiven Punktabzügen führen!

0.5 Weiterführende Informationen zu Java

- Java API: `https://docs.oracle.com/en/java/javase/11/docs/api/index.html`