Dokumentation der Praktischen Arbeit zur Prüfung zum Mathematisch-technischen Softwareentwickler

30. März 2023

Nico Meyer

Prüfungs-Nummer: TODO

Programmiersprache: Java

Inhaltsverzeichnis

1 Aufgabenanalyse **4**

1.1 Interpretation der Aufgabe 4

1.2 Fehlerarten 4

1.2.1 Technische Fehler 5

1.2.2 Syntaktische Fehler 5

1.2.3 Semantische Fehler 5

1.3 Fehlerbehandlung 5

1.3.1 Technische Fehler 5

1.3.2 Syntaktische Fehler 5

1.3.3 Semantische Fehler 5

1.3.4 Sonderfälle 5

2 Verfahrensbeschreibung **6**

2.1 Gesamtsystem 6

2.1.1 Eingabe 6

2.1.2 Verarbeitung 6

2.1.3 Ausgabe 6

2.1.4 Datenstrukturen 6

3 Programmbeschreibung **7**

3.1 Struktogramm 7

3.2 Entwicklungsdokumentation 7

4 Testdokumentation **8**

4.1 Normalfälle 8

4.2 Fehlerfälle 8

4.3 Grenzfälle 8

4.4 Sonderfälle 8

5 Zusammenfassung und Ausblick 9
5.1 Zusammenfassung 9
5.2 Ausblick 9

A Abweichung und Ergänzung 10

B Benutzeranleitung 11
B.1 Vorbereiten des Systems 11
B.1.1 Systemvoraussetzungen 11
B.1.2 Installation 11
B.2 Programmaufruf 11
B.3 Testen der Beispiele 11
B.4 Kompilieren 12

C Entwicklungsumgebung 13

D Verwendete Hilfsmittel 14

E Erklärung 15

1 Aufgabenanalyse

1.1 Interpretation der Aufgabe

Gefordert ist ein Programm, welches TODO

Als Datenstruktur wird .
Bevorzugt wird float, da 32-Bit mehr als ausreichend für Rechenoperationen mit Zahlen von bis zu 6.1. Zudem kann durch Hardwarebeschleunigung, oder auch ohne, eine kürzere Laufzeit erreicht werden, im Gegensatz zu Double-Operationen.

Die Eingabedatei wird zeilenweise eingelesen. Beginnt eine Zeile mit einem Semicolon, wird diese als Kommentar beziehungsweise als Beschreibung interpretiert.

Abbildung 1.1: Input-Restriktionen

- Restriktion 1.

Die Lösung des Problems wird mittels TODO realisiert. TODO Beschreibe Algorithmus

1.2 Fehlerarten

Die Eingabedatei kann verschiedene Integritätsbedingungen verletzen. Das Programm muss diese Fehlerarten identifizieren und den Nutzer darüber informieren.

1.2.1 Technische Fehler

1.2.2 Syntaktische Fehler

Die Eingabedatei muss der Struktur aus Input-Restriktionen entsprechen. So kann zum Beispiel ein syntaktischer Fehler provoziert werden, indem TODO

1.2.3 Semantische Fehler

1.3 Fehlerbehandlung

1.3.1 Technische Fehler

1.3.2 Syntaktische Fehler

1.3.3 Semantische Fehler

1.3.4 Sonderfälle

2 Verfahrensbeschreibung

2.1 Gesamtsystem

Es wurde eine Mischung aus EVA und MVC genutzt.

2.1.1 Eingabe

2.1.2 Verarbeitung

2.1.3 Ausgabe

2.1.4 Datenstrukturen

3 Programmbeschreibung

3.1 Struktogramm

3.2 Entwicklungsdokumentation

Die Dokumentation des Programms wurde in Javadoc vorgenommen und kann im Ordner javadoc eingesehen werden. Hierzu kann die index.html aufgerufen werden

4 Testdokumentation

4.1 Normalfälle

4.2 Fehlerfälle

4.3 Grenzfälle

4.4 Sonderfälle

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

5.2 Ausblick

A Abweichung und Ergänzung

B Benutzeranleitung

B.1 Vorbereiten des Systems

B.1.1 Systemvoraussetzungen

Um sicherzustellen, dass das Programm lauffähig ist, sollte Linux (EndeavourOS) als Betriebssystem genutzt werden. Es ist außerdem eine JRE oder JDK in der Version 17 oder höher vonnöten, um das Programm auszuführen.
Falls eine erneute Kompilierung des Programms gewünscht ist, empfiehlt es sich die JDK anstelle der JRE zu installieren. Um diese JRE/JDK anschließend zu nutzen, muss das bin-Verzeichnis dieser in die PATH-Umgebungsvariable hinzugefügt werden.

B.1.2 Installation

Es ist keine Installation nötig, es reicht das .zip-Verzeichnis zu entpacken.

B.2 Programmaufruf

Nach dem Entpacken kann das Programm über den Befehl

```
java -jar GroPro-1.0.jar "Testbeispiele/Test1IHK.txt"
```

in der Eingabeaufforderung (CMD) oder beliebiger Bash ausgeführt werden.

B.3 Testen der Beispiele

Das Ausführen der automatischen Tests erfolgt über die Datei „RunTestBeispiele.cmd“. In der Konsole wird nun das Programm für jede Datei im Ordner „Test-

beispiele“ ausgeführt. Alle Dateiausgaben befinden sich im Anschluss im Ordner „Testbeispiele/out“.

B.4 Kompilieren

Zum Erzeugen der .jar-Datei sollte Maven genutzt werden. Der Einfachheit halber muss das bin-Verzeichnis der Maveninstallation ebenfalls in der PATH-Umgebungsvariable aufgenommen werden. Anschließend lässt sich das Programm kompilieren, indem der Befehl

```
mvn package
```

im Root-Verzeichnis des Quellcodes (das ist der Ordner, indem sich die „pom.xml“ befindet) ausgeführt wird. Eine JAR-Datei befindet sich dann im Ordner „target“.

C Entwicklungsumgebung

Betriebssystem	Linux (EndeavourOS)
Hardware	Lenovo E14 Gen4 AMD Ryzen 7 5825U with Radeon Graphics (16) @ 2.000GHz 32GB RAM
Compiler	Java Development-Kit 17

D Verwendete Hilfsmittel

- IntelliJ IDE 2022.1 (Ultimate Edition)
  Entwicklungsumgebung und Editor für Java und andere Programmiersprachen
  https://www.jetbrains.com/de-de/idea/
- Maven
  Build-Tool für Java
  https://maven.apache.org
- TexLive 2022
  Softwarepaket für LaTeX
  https://miktex.org/
- Structorizer
  Programm zur Erstellung von Nassi-Shneiderman-Diagrammen
  https://structorizer.fisch.lu/
- Visual Paradigm
  Programm zur Modellierung von Software-(Diagrammen)
  https://www.visual-paradigm.com/
- Git
  Versionsverwaltungssystem
  https://git-scm.com/

E Erklärung

Ich erkläre verbindlich, dass das vorliegende Prüfprodukt von mir selbstständig erstellt wurde. Die als Arbeitshilfe genutzten Unterlagen sind der Arbeit vollständig aufgeführt. Ich versichere, dass der vorgelegte Ausdruck mit dem Inhalt des von mir erstellten Datenträgers identisch ist. Weder ganz noch in Teilen wurde die Arbeit bereits als Prüfungsleistung vorgelegt. Mir ist bewusst, dass jedes Zuwiderhandeln als Täuschungsversuch zu gelten hat, der die Anerkennung des Prüfprodukts als Prüfungsleistung ausschließt.

Köln, den 30. März 2023

_____

Nico Meyer