Praktikum „SEP: Java-Programmierung“ SS 2019

Trie: Häufige Fragen

Thomas Bunk und Karlheinz Friedberger

Ab wann kann abgegeben werden?

Seit Beginn der Aufgabe.

Es können beliebig viele Abgaben hochgeladen werden, nur die letzte Abgabe zählt.

Deadline ist **Dienstag, der 14. Mai, 10:00 Uhr**.

Dürfen zusätzliche Klassen implementiert werden?

Ja. Die gegebenen Klassen sollten aber für eine verständliche und funktionelle Implementierung ausreichen.

Die Verwendung vieler Klassen für geringe Funktionalität kann auch (muss aber nicht) ein Hinweis auf eine komplizierte und unverständliche Programmierung sein.

Dürfen zusätzliche public-Methoden implementiert werden?

Nein. D. h., dass auch alle zusätzlichen Klassen und deren Methoden maximal package-private/protected sein dürfen. Zusätzlich müssen die gegebenen Signaturen genau so übernommen werden, wie gegeben.

Bei Fragen zu den Signaturen oder falls euch etwas komisch vorkommt, meldet euch bitte bei einem der Betreuer.

Dürfen externe Bibliotheken (z.B. Guava) verwendet werden?

Nein. In den Einzelabgaben sollen alle Lösungen nur mit den Standardbibliotheken in Java implementiert werden. Als einzige Ausnahme darf JUnit für Unit-Tests verwendet werden.

Muss Node#setChild(char c, Node child) tatsächlich private sein?

Ja. Das ist unsere Anforderung!

Welche Befehle muss die Shell erkennen?

Alle eindeutigen Präfixe der geforderten Kommandos.

Das bedeutet, z. B.: `a`, `ad` und `add` (und nur diese) müssen korrekt als `add` interpretiert werden.

Das Kommando `aa` kann korrekt interpretiert werden, muss es aber nicht. Das wird nicht von uns getestet.

Typische Fehlermeldungen

Public Tests failed:

- Oftmals grobe Bugs oder fehlende Funktionalität
- Falsche Ausgabe, z.B. ausführliche Meldung bei Programstart

Typische Fehlermeldungen

Secret Tests failed:

- Großbuchstaben testen, z.B. `add HANS 12` → Error!
- Ziffern testen, z.B. `add 12 12` → Error!
- Sonderzeichen testen, z.B. `add jürgen 12` → Error!

  → nur Buchstaben `a-z` erlaubt!

- beliebiges Prefix für Kommando sollte ausreichen, z.B.
  - **new**: {n, ne, new, N, NE, NEW, Ne, nE, ...}
  - **delete**: {d, de, del, dele, delet, delete, D, DE, ...}

  → einfachste Lösung: nur ersten Buchstaben prüfen

- Cleanup nicht korrekt, z.B. bei folgender Sequenz:
  `add anton 12 ;  add antonia 12 ;  del anton`

Typische Fehlermeldungen

CheckStyle failed:

- aussagekräftige Fehlermeldung im Praktomat
- 95 % der Fehler durch ungünstige Formatierung, z.B.
  - Tabs vs Spaces
  - `incorrect indentation level`: 4 statt 2 Spaces
  - `Method name 'CHANGE' must match pattern '[a-z][a-z0-9][a-zA-Z0-9_]*'`:
    Bennenung von Methoden, Variablen, Konstanten

→ Richtlinie: Google-Java-Format umsetzen

→ Empfehlung: Auto-Formatter (Plugin) verwenden

Typische Fehlermeldungen

SpotBugs failed:

- `Correctness Warnings`: meistens ein Bug (bei euch!)
- `Malicious code vulnerability Warnings`:
  `field isn't final but should be`
  → `final` verwenden!
- `Performance Warnings`:
  `method concatenates strings using + in a loop`
  → StringBuilder verwenden!
- `Internationalization Warnings`: kann ignoriert werden

Typische Verletzungen der Anforderung

> **Darf *System.out.println* außerhalb der Shell verwendet werden?**

Nein. Die Ein- und Ausgabe für den Benutzer ist Geheimnis der Shell-Klasse.

> **Darf *System.out.println* in der Trie-Klasse verwendet werden?**

Nein, Nein. Siehe oben.

> **Darf *System.out.println* in der Node-Klasse verwendet werden?**

Nein, Nein, Nein. Siehe oben.