Aufgabenblatt 8

Allgemeine Anmerkungen

Ihre Lösung für dieses Aufgabenblatt ist bis Freitag, 12.6., 13h durch `git commit` und `push` abzugeben. Mit der Angabe werden die Dateien `FileNotFoundException.java` und `FileFormatException.java` sowie `CelestialSystemIndexTreeVariantD.java` mitgeliefert. Wenn Sie zusätzlich zu den gefragten Klassen, weitere Klassen oder Interfaces definieren, achten Sie darauf, dass die Klassennamen mit `My` beginnen, um Konflikte mit späteren Aufgabenblättern zu vermeiden.

Ziel

Ziel der Aufgabe ist die Anwendung der Konzepte: Eingabe mit Validierung, Ausnahmebehandlung, Zusicherungen unterscheiden (siehe Skriptum Seiten 114–143).

Gegebenes Klassen und Interfaces

Gegeben sind mehrere Interfaces und Klassendefinitionen in der Datei `CelestialSystemIndexTreeVariantD.java`. (Diese Datei beinhaltet eine mögliche Lösung für Teilaufgaben aus dem Aufgabenblatt 6. `VariantC` wurde durch `VariantD` ersetzt um Konfikte zu vermeiden).

Aufgaben

1. Validierung von Eingabedaten:
   - Fügen Sie der Klasse `ReadDataUtil.java` aus Aufgabenblatt 3 untenstehende Methode hinzu. Anstelle von `In.java` soll ein gepufferter Stream zum Einlesen genutzt werden (siehe Skriptum Seite 129). Falls die Datei nicht gefunden wird, oder das Format nicht wie erwartet ist, soll die gesamte Sammlung der `bodies` unverändert bleiben.
   - Definieren Sie die beiden angegebenen Exceptionklassen in den entsprechenden mitgelieferten Dateien.
   - Testen Sie die Methode mit der Datei `Configuration.csv` (erstellen Sie auch Varianten mit Formatfehlern).
   - Beantworten Sie als Kommentar in der Datei `ReadDataUtil.java` folgende Frage: Wie kann man durch Einfügen von Zeichen `;` und newlines (`\n`) aus `Configuration.csv` eine "fehlerhafte" Datei machen, die trotzdem von der Methode akzeptiert wird? Kann man solche Probleme verhindern?

     ```
     // Reads the positions of the specified 'bodies' on the
     // specified 'day' in the year 2020 from the file with
     // the specified 'path', and sets position and current
     // velocity of each body in the collection accordingly.
     ```

```
// Each line in the file is required to have the
// following format:
// BODY;YEAR;DAY;RAD_AU;SE_LAT;SE_LON
// where BODY is a string, YEAR and DAY is a positive
// integer, and the last three numbers are interpretable
// as 'double' values. The first line (header) is ignored.
// The character ';' must only be used as field separator.
// If the file is not found, an exception of the class
// 'FileNotFoundException' is thrown. If it does not
// comply with the format described above, the method
// throws an exception of the class 'FileFormatException'.
// Both exceptions are subtypes of 'IOException'.
public static void readConfiguration(String path,
                  CelestialBodyCollection bodies,
                  int day) /* TODO: fix signature */ {

     // TODO: implement method.
}
```

2. Ausnahmebehandlung: Ändern Sie die Klasse `Simulation` so, dass sie zwei Kommandozeilenargumente verarbeitet. Das erste Argument ist der Pfad der einzulesenden Datei, das zweite Argument ist eine Ganzzahl, die den Tag der auszulesenden Position bestimmt. Die Klasse soll beim Auftreten von Problemen bei der Ausführung entsprechende Fehlermeldungen ausgeben und die Ausführung beenden. Beispiele für Aufrufe im Kommandozeileninterpreter mit entsprechenden Fehlermeldungen (Sie können zum Ausführen das Terminal in IntelliJ nutzen):

```
$ javac Simulation.java
$ java Simulation Configuration.csv 60
Running simulation ...
$ java Simulation 60
Error: wrong number of arguments (2 arguments needed).
$ java Simulation blah 60
Error: File "blah" not found.
$ java Simulation Configuration_altered.csv 60
Error: File "Configuration_altered.csv"
does not have required format in line 97 (aborting).
$ java Simulation Configuration.csv -17
Error: Invalid day argument (-17).
```

3. Statisches Programmverstehen:
   - Fügen Sie die Vorbedingungen (V) und Nachbedingungen (N) für die Methoden im Interface `VariantDNode` in den Kommentaren hinzu.
   - Fügen Sie Zusicherungen als Hoare-Tripel `{V}S{N}` an den passenden Stellen in der Klasse `CelestialSystemIndexTreeVariantD` ein. `S` soll dabei eine Anweisung sein, die einen Methodenaufruf von `VariantDNode` enthält. Sie können die entsprechenden Zusicherungen

als Kommentare einfügen, oder - sofern sich die Zusicherungen in Java ausdrücken lassen - entsprechende `assert`-Anweisungen.

- Bestimmen Sie jeweils eine Schleifeninvariante für die Schleifen in der Methode `add` der Klasse `CelestialSystemIndexTreeVariantD`. Fügen Sie an den passenden Stellen Kommentare oder `assert`-Anweisung ein.

4. Zusicherungen und Untertypen:
   - Fügen Sie die Vorbedingungen und Nachbedingungen für die Methoden in den Klassen `VariantDNullNode` und `VariantDNonNullNode` als Kommentare bei den Methoden hinzu.
   - Überprüfen Sie, ob die im Skriptum auf Seite 142 beschriebenen Bedingungen für die Bildung von Untertypen erfüllt sind und weisen Sie ggfs. in den Kommentaren auf Fehler hin.