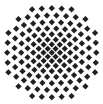

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 3

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 08.11.2019, 23:59 Uhr — Besprechung: KW 46

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studierenden**.
- Dieses Übungsblatt besteht aus zwei Teilen (A+B). Teil A ist in der Präsenzübung zu lösen. Teil B ist in Heimarbeit (Gruppe von 2 Studenten) zu lösen und rechtzeitig abzugeben.
- Für Ihre Abgabe erstellen Sie bitte eine Textdatei für die Aufgaben 1c, 1d und 1e aus Teil B (Endung .txt), in die Sie den Inhalt der Klasse `mysterioushamster` aus BlueJ kopieren. Erstellen Sie zudem EINE weitere PDF-Datei (Endung .pdf) mit Ihrer Lösung aus den Aufgaben 1a, 1b, 1f und 2 aus Teil B.
- Geben Sie zu Beginn der Dateien Ihre Namen (Vor- und Nachname), die Matrikelnummern und die E-Mail-Adressen an. Im Quellcode fügen Sie diese Informationen bitte als Kommentar hinzu.
- Benennen Sie die Dateien nach dem folgenden Schema:
  PSE[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**].[**txt/pdf**].

Lernziel: Dieses Übungsblatt dient dazu, Sie im Finden und Analysieren von Fehlern im Quellcode zu trainieren. Dazu üben Sie u.A. den Umgang mit Abstrakten Syntaxbäumen (ASTs). Sie lernen, ASTs zu erstellen und zu nutzen. Außerdem werden Sie Schnittstellen von Klassen verstehen und erstellen müssen.

Vorbereitung: Stellen Sie vor der Übungsstunde sicher, dass Sie Übungsblatt 1 und 2 absolviert haben, alle Software installiert und funktionsfähig ist (BlueJ und Java 1.11), und Ihr W-Lan richtig konfiguriert ist.

Punkte: Dieses Übungsblatt enthält zwei Teile. Teil A mit 2 Aufgaben und Teil B mit 2 Aufgaben. Im Teil B können Sie bis zu 43 Punkte erzielen.

1 Teil A - Präsenzaufgaben

Erinnerung: Die Dokumentation der API des Hamstersimulators finden Sie unter: `http://caloundra.informatik.uni-stuttgart.de` (nur aus dem Uni-Netz erreichbar). Die Dokumentation der Basisklassen des Java Developer Kits (JDK) finden Sie online unter `https://docs.oracle.com/en/java/javase/11/docs/api/`.

Aufgabe 1 Abstrakter Syntaxbaum (AST)

In dieser Aufgabe geht es darum, aus einem gegebenen Quellcode den AST zu extrahieren, um so den syntaktischen Aufbau des Quellcodes verstehen zu können.

Gegeben ist folgender Code (`myLog` und `tuple` sind Objekte, die in der erweiterten Klasse deklariert und initialisiert wurden. Sie sind vom Typ `Logger` bzw. `Tuple`):

(a) Zeichnen Sie den abstrakten Syntaxbaum für Abbildung 1. Nutzen Sie die Zeichenelemente von Folie 47 aus Vorlesung 3. Für den Rückgabetyp der Operation erstellen Sie einen internen Nachbarknoten des Operationsbezeichners mit der Bezeichnung „RückgabetypDeklaration“ und einem Blatt-Kindknoten mit dem Klassenbezeichner des Rückgabetyps. Für alle Argumentwerte erstellen Sie einen internen Knoten mit der Bezeichnung „ArgumentWerte“ dessen Kinder dann Blatt-Knoten mit der Bezeichnung „ArgumentWert“ sind, an die Sie den jeweiligen Wert

```
class Size extends TupleHolder {

    Integer getColumnCount() {
        myLog.debug("Get Column Count");
        return tuple.get(0);
    }

    Integer getRowCount() {
        myLog.debug("Get Row Count");
        return tuple.get(1);
    }

}
```

Abbildung 1: Java Beispielcode

des Arguments annotieren. Für eine `return`-Instruktion erstellen Sie einen internen Instruktionsknoten mit der Bezeichnung „Instruktion (Rückgabe)“. Dieser bekommt einen internen Kindknoten mit der Bezeichnung „RückgabeWert“. Dieser hat wieder einen internen Kindknoten, der einen „Ausdruck“ darstellt, wobei ein Instruktionsaufruf als solch ein Ausdruck gelten kann, wenn eine Operation mit Rückgabewert (sprich eine Abfrage) aufgerufen wird.

(b) Angenommen `get(...)` des Objekts `tuple` liefert ein String- anstatt eines Integer-Objekts. Welcher der von uns besprochenen Schritte der Programmanalyse (vgl. Foliensatz 3, Folien 51ff.) erkennt diesen Fehler? Handelt es sich um ein syntaktisches, ein semantisches oder ein pragmatisches Problem?

(c) Verändern Sie eine Zeile des gegebenen Programms, um einen syntaktischen Fehler einzubauen. Diskutieren Sie mit ihrem Team, wieso der Fehler syntaktisch ist.

Aufgabe 2 APIs erlernen

In dieser Übung werden Sie mindestens die API des Territoriums unseres Hamstersimulators und die API des Java PrintStream kennenlernen. Ziel ist es dabei, Ihnen beizubringen, wie man sich eine API aneignet und damit arbeitet. Dazu benötigen Sie u.a. die Dokumentation der API der `Territory`-Klasse. Sie finden sie unter `https://tinyurl.com/ybq6otah`.

(a) Öffnen Sie analog zum Vorgehen aus den vorherigen Übungen das git-Repo für Übungsblatt 3. URL: `https://github.com/RSS-PSE-WS1920/exercise-sheet-3`. Programmieren Sie in der `run()` Operation der Klasse `Exercise03`. Lassen Sie Paule folgende Informationen über das Territorium via Paules `write` Operation ausgeben. Das Territorium können Sie über das Objekt `territory` erreichen:

1. Die Anzahl an Körnern, die auf Kacheln liegen
2. Die Gesamtzahl an Hamstern im Territorium
3. Die Größe des Territoriums

Hinweis: In Java können Sie Zeichenketten (Strings) durch das Verwenden des Plus-Operators (”`+`”) zusammenbauen. Beispiel: `"Hallo " + "Paule " + 42`.

(b) Es steht Ihnen auch ein `output` Objekt zur Verfügung, dessen Klasse `PrintStream` aus dem JDK stammt (siehe `https://tinyurl.com/jp5of6d`). Erweitern Sie das Programm aus a) so, dass die Informationen auch auf einer Konsole ausgegeben werden. Erkunden Sie dazu die API von `PrintStream`.
Herausforderung 1: Nutzen Sie dieses Mal nicht den ”+”-Operator!

(c) Lassen Sie Paule die beiden Körner, die in seinem direkten Weg liegen aufsammeln. Geben Sie danach wieder die Informationen aus a) und b) aus und prüfen Sie, dass sich die Informationen passend geändert haben.

(d) Sehen Sie sich die Klasse `LocationVector` an, die sich im BlueJ-Projekt für dieses Übungsblatt befindet. Diskutieren Sie mit Ihrem Gruppenpartner, was die Objekte dieser Klasse wohl machen könnten und wo sie genutzt werden könnten. Leider wurde die Klasse bisher nicht dokumentiert. Schlagen Sie daher eine Dokumentation für die Klasse selbst und die beiden `get`-Abfragen vor. Beschreiben Sie so viel wie möglich. Nutzen Sie natürliche Sprache (Englisch, bzw. wenn nicht anders möglich Deutsch).

(e) Formalisieren Sie Ihre eigene Antwort zum Aufgabenteil d) mittels Javadoc. Ignorieren Sie die Operationen, die nicht Teil der vorherigen Teilaufgabe waren. Schauen Sie sich dann Ihre Dokumentation in BlueJ an, indem Sie in dem Dropdown-Element rechts oben im Quellcodeeditor von Quellcode auf Dokumentation umstellen.

(f) **Herausforderung 2:** Der Hamstersimulator benutzt JavaFX (`https://openjfx.io/`) als UI Bibliothek. Finden Sie die API heraus (siehe `https://openjfx.io/javadoc/11/`), mit der Sie ein Nachrichtenfenster in JavaFX anzeigen können. Zeigen Sie ein solches Fenster an und geben die Informationen aus a) und b) darin aus. Erarbeiten Sie sich dazu die nötige API selbständig. Fügen Sie ihren Code in die vorgesehenen Stellen in der Datei aus unserem BlueJ-Projekt ein.

2 Teil B - Heimarbeit

Aufgabe 1 Der mysteriöse Hamster

In dieser Übung betrachten Sie die Klasse `mysterioushamster`. Die Aufgabe dient dazu, das Erkunden von Programmen, die andere Entwickler geschrieben haben, zu üben.

Leider hat der Autor sich nicht sehr viel Mühe bei der Erstellung der Klasse gegeben. Daher sollen Sie im Folgenden – durch Beobachten des Verhaltens des Hamsters – Schritt für Schritt herausfinden, was der `mysterioushamster` eigentlich alles kann. Es ist nicht notwendig, die Implementierung der Klasse zu lesen oder zu verstehen.

(a) *(3 Punkte)* Ohne den Code im Detail zu betrachten: Finden Sie einen schwerwiegenden Stylefehler, der dem Autor der Klasse unterlaufen ist.

Beschreiben Sie kurz den Fehler, warum er vermieden werden sollte und wie er korrigiert werden könnte. Sie müssen hierzu weder den Code verändern, noch ihn verstehen.

(b) *(4 Punkte)* Erzeugen Sie in BlueJ (wie in Übungsblatt 1 gelernt) ein neues `HomeworkHamster`-Objekt und rufen Sie die Operation `run()` auf. Die Operation `run()` ruft bei der Ausführung Operationen aus dem `mysterioushamster` auf.

Beschreiben Sie ganz grundsätzlich in 2-3 Sätzen, was der Hamster tut, ohne dabei auf jede einzelne Aktion einzugehen.

(c) *(10 Punkte)* Nun sollen Sie herausfinden, was die einzelnen Operationen `doSomething1()` bis `doSomething9()` des Hamsters eigentlich machen.

Hinweis: Manche Operationen setzen voraus, dass sich der Hamster an einer bestimmten Position befindet. Sie können sich hierzu auch die Operation `run()` aus dem vorigen Aufgabenteil genauer ansehen.

Hinweis: Starten Sie vor der Ausführung der ersten Operation den Hamstersimulator neu.

Sie können hierzu (wie in Übungsblatt 2) den Hamster von `HomeworkHamster` über die Operation `getHamster()` erhalten und anschließend seine Operationen aufrufen.

Wenn Sie schon etwas besser programmieren können, können Sie auch den Quellcode der Operationen in `mysterioushamster` an sich betrachten. Allerdings sollten Sie vorher versuchen, ohne den Code und nur durch Studieren des Verhaltens die Aufgabe zu lösen.

Erstellen Sie für jede der 9 Operationen einen Javadoc-Kommentar, in dem Sie kurz und präzise beschreiben, was die jeweilige Operation macht. In diesem Fall müssen Sie noch keine Vor- und Nachbedingungen angeben.

(d) *(8 Punkte)* Finden Sie geeignete Bezeichner für die Operationen `doSomething1()` und `doSomething3()` bis `doSomething9()` und benennen Sie die Operationen entsprechend um.

Denken Sie daran, dass Sie auch die Aufrufe in `run()` von `HomeworkHamster` abändern müssen.

Nach der Änderung ist es notwendig, das Programm wieder zu übersetzen, falls Sie den Code erneut ausführen möchten.

(e) *(1 Punkt)* (Bonus) Finden Sie, analog zum vorigen Aufgabenteil, einen geeigneten Bezeichner für die Operation `doSomething2()`.

(f) *(9 Punkte)* Entscheiden Sie für jede der 9 Operationen, ob es sich um ein Kommando oder eine Abfrage handelt. Begründen Sie kurz!

Aufgabe 2 Objekte verstehen

In dieser Aufgabe sollen Sie zeigen, dass Sie ein grundlegendes Verständnis für Objekte erlangt haben.

(a) *(2 Punkte)* Erzeugen Sie in BlueJ (wie in Übungsblatt 1 gelernt) ein neues `HomeworkHamster`-Objekt. Beschreiben Sie kurz, was auf inhaltlicher und technischer Ebene passiert.

(b) *(2 Punkte)* Die Klasse `HomeworkHamster` bietet die Operation `getHamster()` an. Handelt es sich hierbei um eine Abfrage oder ein Kommando? Begründen Sie kurz.

(c) *(2 Punkte)* (Bonus) Beschreiben Sie, was inhaltlich passiert, wenn Sie `getHamster()` aufrufen und dann `Hole` wählen. Überlegen Sie sich hierzu, was der Unterschied zum Erzeugen des Objekts aus Aufgabenteil (a) ist.

(d) *(2 Punkte)* (Bonus) Beschreiben Sie, was passiert, wenn Sie `getHamster()` zweimal aufrufen und jeweils `Hole` wählen. Rufen Sie hierzu verschiedene Operationen auf und schauen Sie, was im Spiel passiert. Erklären Sie kurz, warum sich das Spiel so verhält.

3 Teil L - Lehramtsteil

Dieser Teil enthält eine Präsenzaufgabe die speziell für Lehramtsstudierende konzipiert ist. Das bedeutet allerdings nicht, dass "Nicht-Lehramtsstudierende" diese nicht auch bearbeiten dürfen. Sie bezieht sich auf den Vorlesungsstoff und die Übungsaufgaben, betrachtet diese aber aus einer didaktischen Perspektive.

Aufgabe 1 Die kleine Hamsterwelt

Sie haben in der Vorlesung und den bisherigen Übungen den Hamster Paule und seine kleine Hamsterwelt kennen gelernt. Diese Welten nennt man auch "Miniwelten" (oder "Minisprachen"). Ziel dieser Aufgabe ist es, dass Sie sich mit Vor- und Nachteilen solcher Miniwelten im Bezug auf den Informatikunterricht auseinandersetzen.

(a) Überlegen Sie sich, was eine "Miniwelt" (oder auch "Minisprache") als solche auszeichnet.

(b) Diskutieren Sie mit ihrem Team, warum der Einsatz einer Miniwelt wie die des Hamsters im Unterricht für den Anfang sinnvoll sein könnte.

(c) Diskutieren Sie mit ihrem Team, welche Nachteile der Einsatz einer Miniwelt wie die des Hamsters im Unterricht haben könnte.

Hinweise zu den Übungen:

- Abgaben erfolgen übers ILIAS.
- Durch die Teilnahme am Übungsbetrieb können Sie sich für die Teilnahme an der Klausur qualifizieren.
  - Bestehen von min. 80% aller Übungsblätter.
  - Ein Übungsblatt gilt als bestanden wenn 50% der Punkte erreicht wurden.
  - Teilnahme an min. 80% der Übungen.
  - Bestehen der Scheinklausur.

Viel Erfolg!