Karlsruher Institut für Technologie

11. Tutorium
5. ÜB, Debugging, Modern Java, Tipps

Tutorium 14

Péter Bohner | 01.02.2023

Inhaltsverzeichnis

Wiederholung

Wahr oder Falsch: JUnit: Eine Methode mit Annotation @BeforeClass wird vor jedem Testfall ausgeführt

Falsch

Wie heißt die Annotation dann?

`@BeforeEach`

Wie überprüft man Werte in Testfällen

Mit **assertEquals(expected, actual)**, assertThrows, assertFalse, etc.

Warum sollte man Unit-Tests verwenden?

Um die Anforderungen zu modellieren; Um sicher zu gehen, dass Änderungen Funktionalität nicht kaputt machen

Was für Arten von Test gibt es?

Unit Tests, Integrationstests, Systemtest, Akzeptanztest, uvm

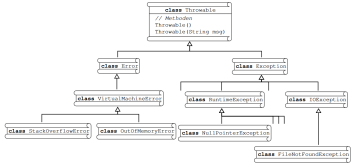
Besprechung Übungsblatt 5

- Nur 2 Abgaben, ich halte es kurz
- Konstanten verwenden
- Geeignete Datentypen verwenden. String ist kein guter Datentyp für einen Charakter
- Code-Duplikate durch Vererbung und Hilfsmethoden vermeiden
- Datenkapselung beachten, falls Referenz nicht unbedingt nötig, kopiere Collections, Arrays und Objekte beim zurückgeben/annehmen
- Ein erweiterbares Befehlssystem bauen. If/else ist das nicht (Für die Abschlussaufgaben)
- Befehlhandler sollen keine Programmlogik enthalten, sondern nur Programmlogik aufrufen und das Ergebnis formatieren und ausgeben

Command Handling

- Ein Befehlssystem, wie in den ÜB wird in den Abschlussaufgaben drankommen.
- Ziel, Commands erweiterbar implementieren
- Idee: abstrakter Command, mit Methoden **match** und **execute**
- Handler Klasse für einzelne Commands, die Argumente parst, die Logik aufruft, dessen output formatiert und diesen zurückgibt.
- IO-Klasse prüft welcher Handler der richtige ist, und ruft in dann auf

Eigene Exceptions schreiben



- Eigene Hierarchie aufbauen
- Von der Klasse Exception erben
- Erste Klasse ist **`ProgrammNameException`**
- Bspw. erben `InvalidInputException` und `LogicException` davon
- Erleichtert Abfangen in der main-Methode: Nur noch **`ProgrammNameException`** muss gefangen werden

Suchen und Sortieren

Karlsruher Institut für Technologie

Was solltet ihr Wissen?

Suchen

- Lineare Suche
- Binäre Suche
  - `Arrays.binarySearch(Datentyp[] array, Datentyp key)`

Sortieralgorithmen

- Sortieralgorithmen nicht notwendig für die Abschlussaufgaben
- `Arrays.sort(Datentyp[] array)`
- `Collections.sort(List<T> list)`

Parsen

Wichtig

- Nutzt `java.util.regex` oder die `matches(String regex)` Methode der Klasse `String`
- Sinnvolle Trennung von Nutzerinteraktion und Logik verwenden
- Daran denken auch falsche Eingaben (bspw. Argumentanzahl falsch) abzufangen
- Bestenfalls mit dem Command Pattern arbeiten

Datenkapselung

Shallow Copy

- Attribute des Objektes werden 1 zu 1 kopiert.
- d.h. für nicht elemetare Attribute, werden Referenzen kopiert
- Kopie nicht komplett unabhängig vom orignal Objekt

$\Rightarrow$ Seiteneffekte möglich

Deep Copy

- nur Attribute mit elementaren Datentypen des Objektes werden 1 zu 1 kopiert.
- d.h. für nicht elemetare Attribute, werden auch Kopien mit neuen Referenzen erstellt
- Kopie komplett unabhängig vom orignal Objekt

$\Rightarrow$ keine Seiteneffekte möglich

Alternative: Objekte als *immutables* (unveränderbar) designen.

Tipps & Tricks

Erst denken - dann programmieren!

- Entwickle einen Plan vor der Implementierung
- „Divide and conquer“
- Stift und Papier verwenden (Aufbaudiagramme & Pseudocode)
- In welcher Beziehung stehen Klassen?
- durch welche Eigenschaften sind diese modelliert
- gibt es eine sinnvolle Vererbungshierachie
- Linguistische Analyse
  - Nomen (Klasse), Verb „etwas tun“ (Operation), Verb „sein“ (Vererbung), „müssen, können“ (Einschränkung), Adjektiv (Attribut)
- Finden von Schnittstellen (Sichtbarkeit)
- Finden geeigneter Datenstrukturen

Tipps & Tricks

„Fail fast“

- Mit Fehlern rechnen
- Fehler früh erkennen
- Konsistenten Zustand einer Klasse behalten
- Exceptions verwenden
- Exceptions sind Ausnahmesituationen, nicht Kontrollfluss!
- Eigene Exception schreiben

Tipps & Tricks

Datenkapselung

- Attribute immer `private`
- Sinnvolle Getter/Setter
- Überprüfen der übergebenen Werte
- Nur `public` wenn notwendig
- Nie Implementierungsdetails preisgeben
- Perspektive eines Angreifers einnehmen

Tipps & Tricks

- Bewertungsrichtlinien, ILIAS WIKI durchlesen
- regelmäßig Blatt aktualisieren und Forum durchlesen
- macht Backups (am besten git nutzen)
- Testen, testen, testen! (Schrittweise Testen)
- Debugger nutzen!
- Nutzt Enums, Vererbung, Polymorphie, Interfaces, abstrakte Klassen, ...
- so wenig wie möglich statisch
- Regex zum parsen der Befehle
- keine zusätzlichen Funktionen und nur genau wie spezifiziert
- Paketstruktur

Tipps & Tricks

- je Aufgabe: 10P Funktionalität, 10P Methodik
- Funktionalität vgl. ÜB5 aus WS20/21:
  - 1 Public Test: 0,5 Punkte
  - Private Test 01 (22 Tests): 0,25 Punkte pro richtigem Test
    - Tests if the program works correctly with correct input.
  - Private Test 02 (15 Tests): 0,2 Punkte pro richtigem Test
    - Tests if semantic errors are handled correctly.
  - Private Test 03 (09 Tests): 0,2 Punkte pro richtigem Test
    - Tests if syntactic errors are handled correctly.
  - Private Test 04 (08 Tests): 0,2 Punkte pro richtigem Test
    - Tests if errors within arguments are handled correctly.
- Methodik:
  - Nicht alle Methodikverletzungen gleich viel Abzug
  - vermeidet unnötige Abzüge für fehlenden/trivialen JavaDoc und MagicNumbers/Strings

Bis zum nächsten Tutorium am 08.02.2023.