Karlsruher Institut für Technologie

Programmieren – Sommersemester 2017

Architecture-driven Requirements Engineering (ARE)
`https://sdqweb.ipd.kit.edu/wiki/Programmieren`
Jun.-Prof. Dr.-Ing. Anne Koziolek

Übungsblatt 2

Ausgabe: 08.05.2017 – 13:00
Abgabe: 23.05.2017 – 06:00

Allgemeine Hinweise

- Achten Sie darauf nicht zu lange Zeilen, Methoden und Dateien zu erstellen
- Programmcode muss in englischer Sprache verfasst sein
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete `java.lang` und `java.io`, es sei denn die Aufgabenstellung erlaubt ausdrücklich weitere Pakete[1]
- Achten Sie auf fehlerfrei kompilierenden Programmcode[1]
- Halten Sie alle Whitespace-Regeln ein
- Halten Sie die Regeln zu Variablen-, Methoden und Paketbenennung ein und wählen Sie aussagekräftige Namen

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Montag, den 15. Mai um 13:00 Uhr**, freigeschaltet.

- Geben Sie die Java-Klassen zu Aufgabe A als `*.java`-Dateien ab.
- Geben Sie die Java-Klassen zu Aufgabe B als `*.java`-Dateien ab.

Achten Sie unbedingt darauf, diese Dateien bei der richtigen Aufgabe hochzuladen.

Einführung von Checkstyle

Beginnend mit diesem Übungsblatt überprüft der Praktomat Ihre Quelltexte während der Abgabe automatisiert auf die Einhaltung obenstehender Regeln. Falls eine Regel entsprechend markiert ist, wird der Praktomat die **Abgabe zurückweisen**, wenn die Regel verletzt ist. Andere Regelverletzungen führen zu **Punktabzug**.
Sie können (und sollten) Ihre Quelltexte bereits während der Entwicklung auf die Regeleinhaltung überprüfen. Das Programmieren-Wiki beschreibt, wie das funktioniert.

Öffentliche Tests

Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe dieses Blattes nötig ist. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

[1] Der Praktomat wird die Abgabe zurückweisen, falls diese Regel verletzt ist.

Objektorientierte Modellierung

Achten Sie darauf, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung, als auch Funktionalität bewertet werden.

A Arrays (11 Punkte)

Laden Sie die Klasse `NaturalNumberTuple`[2] von unserer Homepage herunter und schreiben Sie die Klasse entsprechend der nachfolgenden Beschreibung fertig.

Eine Instanz von `NaturalNumberTuple` repräsentiert ein Tupel von positiven ganzen Zahlen (0 ist ausgeschlossen). Ein Tupel ist eine Liste von Objekten (bei uns: Zahlen). Im Gegensatz zu einer Menge können Objekte mehrfach in einem Tupel vorkommen. Die Reihenfolge der Objekte in einem Tupel ist relevant. Nutzen Sie als interne Datenstruktur ein **Array** zur Repräsentation eines Tupels.

`NaturalNumberTuple` soll die unten angegebenen Methoden bzw. Konstruktoren implementieren. Vermeiden Sie Code-Duplikate, indem Sie bei der Implementierung einer Methode gegebenenfalls andere Methoden der `NaturalNumberTuple`-Klasse aufrufen. Denken Sie daran, keine Klassen aus `java.util` zu verwenden.

- `public NaturalNumberTuple(int[] numbers)` initialisiert die Zahlen des Tupels mit den übergebenen `numbers`. Sollte `numbers` negative Zahlen oder 0 enthalten, werden diese bei der Initialisierung nicht berücksichtigt.
- `public int min()` Gibt die kleinste Zahl innerhalb dieses Tupels zurück, falls vorhanden, ansonsten -1.
- `public int max()` Gibt die größe Zahl innerhalb dieses Tupels zurück, falls vorhanden, ansonsten -1.
- `public NaturalNumberTuple insert(int number)` Fügt `number` am Ende des Tupels ein, falls `number` größer 0 ist, und gibt das resultierende Tupel zurück. Das ursprüngliche Tupel wird nicht verändert.
- `public NaturalNumberTuple remove(int number)` Löscht alle Vorkommen von `number`, falls vorhanden, und gibt das resultierende Tupel zurück. Das ursprüngliche Tupel wird nicht verändert.
- `public int indexOf(int number)` Gibt den Index des ersten Vorkommens von `number` innerhalb dieses Tupels zurück, falls vorhanden, ansonsten -1. Die erste Zahl des Tupels besitzt den Index 0.
- `public int countNumbers(int number)` Gibt aus, wie oft `number` in diesem Tupel enthalten ist.
- `public NaturalNumberTuple swap(int firstPosition, int lastPosition)` Tauscht die beiden Zahlen an den vorgegebenen Indizes aus und gibt das resultierende Tupel zurück. Das ursprüngliche Tupel wird nicht verändert. Kommt einer oder beide der Indizes im Tupel nicht vor, wird das initiale Tupel zurückgegeben.
- `public NaturalNumberTuple toSet()` Gibt eine Kopie dieses Tupels zurück, bei dem alle mehrfachen Vorkommen einer Zahl entfernt wurden, so dass jede Zahl maximal einmal vorkommt. Diese Methode hat keine Auswirkung auf den Zustand dieses Tupels. Das resultierende Tupel selbst ist aufsteigend sortiert.
- `public boolean equals(NaturalNumberTuple tuple)` Gibt `true` zurück, falls dieses Tupel und das übergebene `tuple` identisch sind; ansonsten `false`. Zwei Tupel sind identisch, wenn beide die gleiche Zahlenfolge enthalten.
- `public void print()`[3] Gibt die Elemente dieses Tupels auf der Konsole aus. (Hinweis, beachten Sie strikt folgendes Ausgabeformat: Alle Zahlen werden in einer einzigen Zeile ausgegeben und sind durch ein Komma separiert. Vermeiden Sie Leerzeichen vor oder nach dem Komma.)

[2] https://sdqweb.ipd.kit.edu/lehre/SS17-Programmieren/NaturalNumberTuple.java
[3] Sie dürfen zusätzlich die toString-Methode implementieren und diese in der print-Methode aufrufen.

B Supermarkt (9 Punkte)

In dieser Aufgabe soll ein Supermarkt modelliert werden. Es geht hierbei alleinig um die objektorientierte Modellierung. Die Bewertung erfolgt aufgrund der Struktur ihrer Modellierung und der Dokumentation Ihrer getroffenen Entwurfsentscheidung und nicht anhand der Funktionalität. Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute und kommentieren Sie Ihren Code angemessen. Überlegen Sie sich eigenständig sinnvolle Benennungen der einzelnen Klassen, Methoden und Attribute. Halten Sie dabei die Regeln zu Klassen-, Methoden- und Attributebenennung ein und wählen Sie aussagekräftige englischsprachige Bezeichner. Fügen Sie allen modellierten Klassen Konstruktoren mit passenden Parametern hinzu, sodass ein Objekt nach seiner Erzeugung einen arbeitsfähigen und nach den üblichen Praktiken adäquaten Anfangszustand (bzw. gesetzte default-Werte) aufweisen kann. Ihre Methoden dürfen null zurückgeben, damit diese fehlerfrei kompilieren. Mit Blick auf Aspekte der Datenkapselung, erweitern Sie Ihre Klassen um geeignete getter- und/oder setter-Methoden. Achten Sie darauf diese entsprechend zu benennen. Wenn sie einen getter/setter/beides weglassen, begründen Sie dies kurz mit Hilfe eines Kommentars. Auch sollen alle modellierten Klassen die *equals*- und *toString*-Methode von *Object* überschreiben. Fehlerfrei kompilierenden Programmcode bildet aber auch hier die Mindestanforderung für eine gültige Abgabe. Da keine Funktionalität getestet wird, wird auch keine *main*-Methode und somit auch keine Ausgabe gefordert. Sie dürfen in dieser Aufgabe alle Klassen der Java-Standardbibliothek einsetzten, insbesondere Klassen des *java.util*-Pakets.

Der folgende Text beinhaltet allgemeine Informationen darüber, welche Elemente und welche Szenarien in einem Supermarkt vorkommen könnten. Es müssen keinesfalls alle Elemente exakt wie beschrieben modelliert werden. Deshalb werden ausnahmsweise Rückfragen, welche sich auf den folgenden Text beziehen, nicht beantwortet.

Die folgenden Elemente sollen für die Modellierung des Supermarktes mindestens berücksichtigt werden: Gebäude, Angestellte, Artikel, Kassen und Kunden.

Ein Supermarkt ist in einem Gebäude untergebracht und hat mehrere Kassen und Angestellte. Jedes Gebäude hat eine Anschrift, welche sich aus einem Straßennamen, einer Hausnummer, einer Postleitzahl und einer Stadt zusammensetzt. Auch hat ein Supermarkt mehrere verschiedene Artikel in unterschiedlicher Anzahl, welche an Kunden verkauft werden können.

Ein Angestellter kann ein Filialleiter oder Mitarbeiter sein. Beide haben einen Namen und eine Anschrift, sowie eine Personalnummer. Jede Rolle „Angestellter“ hat zusätzlich gewisse Privilegien. Ein Filialleiter kann zum Beispiel zusätzlich die Filiale auf- und abschließen oder auch neue Artikel für seinen Supermarkt bestellen. Der Mitarbeiter kann zum Beispiel Regale bestücken oder Kunden beraten.

Artikel haben neben einem Namen und einem Preis noch eine Angabe zu Gewicht und Füllmenge. Der Supermarkt unterscheidet drei Artikelkategorien: Regalartikel, Tiefkühlartikel und Tagesartikel. Regalartikel und Tiefkühlartikel haben ein Ablaufdatum. Tiefkühlartikel haben unterschiedliche Haltbarkeit bei unterschiedlichen Temperaturen, beispielsweise ist eine Pizza im Kühlschrank einen Tag und im Gefrierschrank zwei Wochen haltbar. Tagesartikel haben kein Ablaufdatum, jedoch ein Herstellungsdatum und müssen nach zwei Tagen nach ihrer Herstellung aus dem Verkauf genommen werden.

Kassen können von allen Angestellten eines Supermarkts besetzt werden. Wenn eine Kasse besetzt ist, können Kunden dort Artikel bezahlen oder eine kostenlose Kundenkarte erwerben. Kunden können in einem Supermarkt Artikel in ihren Einkaufswagen legen und an der Kasse bezahlen. Jeder Kunde kann zusätzlich noch die Kundenkarte haben, die er an der Kasse erwerben kann. Durch den Besitz der Kundenkarte räumt ihm der Supermarkt einen gewissen Prozentsatz Rabatt ein.