Aufgabenstellung für das *erste* Übungsbeispiel der LVA "Objektorientiertes Programmieren"

Abgabeschluss für den Source Code: Mittwoch 24.04.2019

WICHTIG:
Wird das Beispiel nicht fristgerecht abgegeben, haben Sie in Folge nicht mehr die Möglichkeit, die LVA im laufenden Semester erfolgreich abzuschließen!

Abgabegespräch:

Zwischen ***29.04.2019*** *und* ***02.05.2019*** (Jeweils zwischen 13:00 und 18:00 Uhr: Den Tag und die genaue Uhrzeit wählen Sie selbst in TUWEL.)

Allgemeines

Jeder Teilnehmer muss die Beispiele **eigenständig** ausarbeiten. Bei der Abgabe jedes Beispiels gibt es ein Abgabegespräch mit einem Tutor oder Assistenten. Diese Abgabegespräche bestehen aus den folgenden Punkten:

- Im Zuge der Abgaben müssen Sie eine Codeänderung vornehmen können.

  Achtung: Bei der Abgabe werden Autovervollständigung sowie automatische Fehlerkennzeichnung, Fehlerbehebung und Codevorschläge in Eclipse ausgeschaltet sein! Bereiten Sie sich entsprechend vor.

  Eine Anleitung, welche Eclipse Einstellungen dazu anzupassen sind, finden Sie in dem Dokument **`Disabling Code Assistance in Eclipse`** in TUWEL.
- Beantworten von Fragen zu Ihrem Programm.
- Ihr Programm wird mit automatisierten Tests geprüft.
- Beantwortung theoretischer Fragen zu den jeweils in der Übung behandelten Konzepten.

Umsetzung und Abgabe

- Halten Sie sich exakt an die Beschreibungen der Klassen in der **JavaDoc**.
- Geben Sie Ihrem Eclipse-Projekt einen eindeutigen Namen. Halten Sie sich dabei an folgende Benennungsvorschrift:

  Eclipse-Projekt: *matrNr_Beispiel1_Nachname*

  Beispiel: *00123456_Beispiel1_Mustermann*

  ACHTUNG: Wenn Sie sich nicht an die Benennungsvorschrift halten, kann Ihre Abgabe nicht gewertet werden!
- Name und Matrikelnummer muss zu Beginn jeder Klasse in einem Kommentarblock angeführt werden!
- Exportieren Sie das Projekt (inklusive aller für Eclipse notwendigen Dateien, z.B.: .classpath, .project) als ZIP-Datei. Eine Anleitung dazu finden Sie in TUWEL.

Kontakt:

Inhaltliche Fragen: Diskussionsforum in TUWEL

Organisatorische Fragen: oop@ict.tuwien.ac.at

Aufgabe: Restaurant Bestellungsverwaltungssystem (RBVS)

Ziel:

Erwerb konkreter Erfahrung und Kompetenz in der Umsetzung folgender Konzepte:
Vererbung, Casting, Polymorphismus, Overloading, Overriding, Exceptions, Enumerations.

Einleitung:

Für das Erreichen der Minimalanforderung im Praxis-Teil ist Voraussetzung, dass für alle Klassen die in der JavaDoc definierten Anforderungen umgesetzt wurden. Für die Extrapunkte muss die weiter unten angeführte Bonusaufgabe gelöst werden. Die Funktionalität Ihres Programms muss im eigenen Ermessen getestet werden.

Mittels des Restaurant Bestellungsverwaltungssystems (RBVS) ist es möglich in einem Restaurant Bestellungen durchzuführen. RBVS kann für ein Restaurant mehrere Tische und ein Produktsortiment verwalten. Das System bietet die Möglichkeit neue Tische und Produkte hinzuzufügen. RBVS erlaubt Bestellungen von im Produktsortiment enthaltenen Produkten aufzunehmen. Diese Bestellungen werden einem Tisch zugeordnet und haben einen Bestellstatus. Abhängig vom Bestellstatus können Bestellungen geändert werden.

Die Minimalanforderung besteht darin das RBVS entsprechend der Vorgaben zu implementieren.

RBVS teilt sich in mehrere Java-Packages auf. Eine grobe Übersicht der im Gesamtsystem enthaltenen Packages finden Sie im Folgenden. Die detaillierte Beschreibung der Klassen und Interfaces entnehmen Sie der beiliegenden JavaDoc:

Packages

- **rbvs:** In diesem Package werden die Klassen für das Restaurant, die Tische und Bestellungen implementiert.
- **rbvs.product:** Dieses Package bietet die Möglichkeit Produkte zu erstellen und Schnittstellen um auf diese zuzugreifen. Es gibt drei Arten von Produkten: *SimpleProduct*, *ExtendedProduct*, Composite*Product*. Ein *CompositeProduct* ist ein aus mehreren Produkten zusammengesetztes Produkt (z.B. ein Menü in einem Restaurant).
- **rbvs.record:** Deklariert abstrakte Typen für Protokolleinträge.
- **ict.basics:** Dieses Package definiert ein Interface das für Klassen festlegt, dass Kopien Ihrer Objekte erstellt werden können.

BONUSAUFGABE

Erweitern Sie das RBVS so, dass auch Restaurantketten verwaltet werden können. Eine Restaurantkette hat mehr als eine Filiale (Restaurant). Alle Filialen haben dasselbe Produktsortiment wie die Restaurantkette und diese soll über die Restaurantkette geladen und allen Filialen mitgeteilt werden. Änderungen im Produktsortiment der Restaurantkette werden auch an die Filialen versendet und das Produktsortiment der Filiale wird aktualisiert.

Die zu einer Restaurantkette zugehörigen Restaurants sollen dynamisch zum Programmstart anhand einer Konfigurationsdatei erstellt werden. In dieser Konfigurationsdatei ist der Name der Filiale sowie alle Tische mit deren IDs und Anzahl an Sitzplätzen festgelegt. Das Format dieser Einträge ist wie folgt:

```
<<Name der Filiale>>;<<ID des Tisches>>;<<Sitzplätze>>
```

Da jede Filiale mehrere Tische haben kann, können auch mehrere Einträge zu einer Filiale in dieser Konfigurationsdatei existieren.

Implementieren Sie eine Methode die diese Konfigurationsdatei bei der Initialisierung der Restaurantkette einliest und die Restaurants mit ihren Tischen entsprechend anlegt.

Die Basisfunktionen der Klassen entsprechend der JavaDoc muss dabei unverändert bleiben. Eigene Klassen sowie neue Methoden und Attribute in bestehenden Klassen dürfen hinzugefügt werden. Erstellen Sie die Klassen für die Restaurantkette in einem eigenen Package.

Beispielhafte Konfigurationsdatei:

```
Hauptbahnhof;Tisch 1;4
Hauptbahnhof;Tisch 2;2
Mariahilfer Strasse;T1;6
Mariahilfer Strasse;T2;8
Mariahilfer Strasse;T3;4
Palais Luxus;Lounge Mozart;4
Palais Luxus;Steffelblick;6
Palais Luxus;Lipizzanerbox;8
```

Diese Konfigurationsdatei legt drei Filialen (Hauptbahnhof, Mariahilfer Strasse, Palais Luxus) fest. Die einzelnen Filialen haben dabei zwei beziehungsweise drei Tische.