Universität Leipzig **WS 14/15**
Institut für Informatik **Dr. Monika Meiler**

Modellierung und Programmierung 1
Übungsserie 5

Abgabetermin: 11.01.2015, 23:55 Uhr

Grundsätzlich sind Nebenrechnungen anzugeben und Antworten zu begründen.
Einzureichen sind, bei mehreren Dateien als .zip-Archiv:
Lösungen und UML-Klassendiagramme als .pdf-Datei,
Programme als Quellcode und Ergebnisdateien.

---

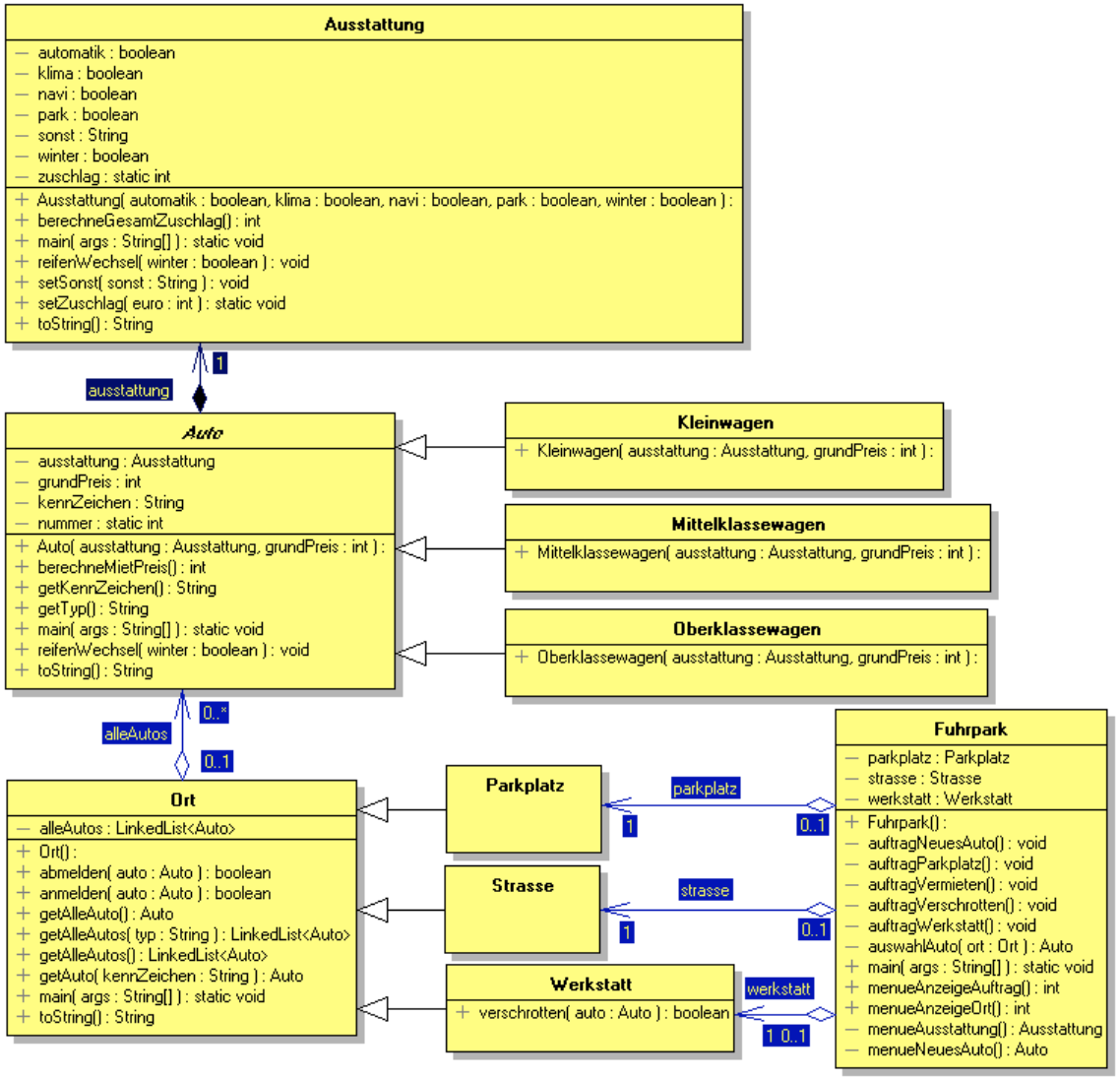
1. **OOP - Implementieren / Collection**
   Der Fuhrpark einer Autovermietung soll künftig mit Hilfe eines Softwaresystems verwaltet werden. Es werden Autos unterschiedlicher Ausstattung und Größe vermietet. Hier einige Beispiele:

Kennzeichen	**L-RA-1001**	**L-RA-1002**	**L-RA-1003**	**L-RA-1004**	**L-RA-1005**	**L-RA-1006**
Grundpreis in Euro	50	40	60	60	90	80
Kleinwagen	x	x				
Mittelklassewagen			x	x		
Oberklassewagen					x	x
Automatik	x			x		x
Klimaanlage	x	x	x	x	x	x
Navigationssystem	x		x	x	x	x
Parksensoren			x			x
Winterreifen						
Besonderheit	neu	Multi-media			Cabrio	

   Der Mietpreis eines Autos pro Tag wird aus dessen Grundpreis und einem Zuschlag von 5 Euro für jedes vorhandene Ausstattungsmerkmal berechnet.
   Der Oberklassewagen L-RA-1005 mit Klimaanlage und Navigationssystems zum Beispiel kostet an einem Tag:

$$Tagesmietpreis = Grundpreis + 2 \cdot Zuschlag = 90 + 2 \cdot 5 = 100 Euro$$

   Ein Softwareentwickler hat mit einer Entwicklungssoftware eine Klassenhierarchie geschaffen, den Quellcode dazu generiert, eine Online-Dokumentation erzeugt und alles im Archiv **AutoVermietung.zip** zusammengefasst. Einem Programmierer wird nun übertragen, alle im Quellcode noch nicht implementierten Bestandteile anhand der Online-Dokumentation zu ergänzen.



a) ***Klasse Ausstattung***: Listen Sie jeweils alle in der Klasse verwendeten Klassenvariablen, Klassenmethoden, Instanzvariablen und Instanzmethoden auf. Erläutern Sie den hier notwendigen Einsatz von Klassenvariablen und Klassenmethoden in maximal »drei Sätzen«.

b) ***Klasse Ausstattung***: Als Ausstattungsmerkmale werden das Vorhandensein von Automatik, Klimaanlage, Navigationssystem, Parksensoren und Winterreifen registriert. Zusätzlich können noch Besonderheiten eingetragen werden.
Ergänzen Sie den Code der Klasse und testen Sie die Klasse mit den obigen sechs Ausstattungsbeispielen (Hauptmethode `main`):
Erzeugen Sie jede dieser Ausstattungen und führen Sie anschließend für die letzte Ausstattung einen Reifenwechsel durch. Geben Sie alle erzeugten Ausstattungen auf der Konsole aus und leiten Sie die Ergebnisse in eine Datei **Ausstattung.out** um.

c) ***Klassenhierarchie Auto***: Die Autovermietung bietet Klein-, Mittelklasse- und Oberklassewagen unterschiedlicher Ausstattung an. Die Kennzeichen der Autos dienen ihrer Identifikation. Sie haben sämtlich den Aufbau L-RA-*nnnn*, wobei die Nummerierung *nnnn* mit 1000 beginnt und fortlaufend erfolgt.
Ergänzen Sie den Code der Klassen Auto, Kleinwagen, Mittelklassewagen und Oberklassewagen und testen Sie in der Klasse Auto die Klassenhierarchie mit den obigen sechs Autobeispielen:
Erzeugen Sie jedes dieser Autos unter Verwendung der Klasse Ausstattung und führen Sie anschließend für das letzte Auto einen Reifenwechsel durch. Geben Sie die erzeugten Autos auf der Konsole aus und leiten Sie die Ergebnisse in eine Datei **Auto.out** um.

d) ***Klassenhierarchie Ort***: Die Mietautos wechseln ihre Aufenthaltsorte. Sie befinden sich auf dem betriebseigenen Parkplatz, auf der Straße oder in einer vertraglich gebundenen Werkstatt.
Ergänzen Sie den noch fehlenden Code der Klassen Ort und Werkstatt und testen Sie in der Klasse Ort die Klassenhierarchie mit den obigen sechs Autobeispielen:
Richten Sie Parkplatz, Straße und Werkstatt als Orte ein. Stellen Sie alle Kleinwagen auf den Parkplatz, alle Mittelklassewagen auf die Straße und alle Oberklassewagen in die Werkstatt. Geben Sie für alle Orte die sich dort befindenten Autos auf der Konsole aus.
Schicken Sie einen Mittelklassewagen Ihrer Wahl von der Strasse auf den Parkplatz und einen Oberklassewagen Ihrer Wahl von der Werkstatt ebenfalls auf den Parkplatz. Der zweite Oberklassewagen wird verschrottet. Geben Sie für alle Orte die sich dort befindenten Autos auf der Konsole aus. Leiten Sie die Ergebnisse in eine Datei **Ort.out** um.

Klasse Fuhrpark: Ein Verwaltungsprogramm (ohne GUI) soll den Dispatcher der Autovermietung bei seiner Tätigkeit unterstützen:
Alle Autos zur Vermietung stehen auf dem Parkplatz bereit. Ist ein Auto vermietet, so befindet es sich auf der Straße. Wird es zurückgegeben, so kommt es in die Werkstatt zur Durchsicht. Dort wird entschieden, ob es weiter vermietet wird und somit wieder auf den Parkplatz kann, oder ob es zu verschrotten ist.
Das Verwaltungsprogramm ermöglicht dem Dispatcher folgende Szenarien:

- **Auftrag "Neues Auto"**:
  Ein neues Auto wird gekauft und auf dem Parkplatz bereitgestellt.
- **Auftrag "Vermieten"**:
  Ein Mieter sucht sich ein Auto aus und fährt damit auf die Straße.
- **Auftrag "Straße -> Werkstatt"**:
  Ein Auto wird zurückgegeben. Es wird zur Durchsicht in die Werkstatt gefahren.
- **Auftrag "Werkstatt -> Parkplatz"**:
  Ein gewartetes Auto wird zurück auf den Parkplatz gefahren.
- **Auftrag "Verschrotten"**:
  Ein Auto wird verschrottet.

Das Verwaltungsprogramm gibt weiterhin dem Dispatcher die Gelegenheit, sich zu jeder Zeit einen **Überblick** über die Aufenhaltorte seiner Autos zu verschaffen:

- Welche Autos befinden sich auf dem **Parkplatz**?
- Welche Autos befinden sich auf der **Straße**?
- Welche Autos befinden sich in der **Werkstatt**?

Implementieren Sie im Verwaltungsprogramm alle Diestleistungen wie folgt:

e) Stellen Sie dem Dispatcher ein Menü für die Aufträge `menueAnzeigeAuftrag()` und ein Menü fuer den Überblick `menueAnzeigeOrt()` zur Verfügung. Implementieren Sie zunächst den Überblick über die Aufenhaltorte seiner Autos.

f) **Auftrag "Neues Auto"** Ergänzen Sie zuerst den fehlenden Code der Klasse Fuhrpark für den Kauf eines Autos und kaufen Sie die obigen sechs Autos zum Test ein. Geben Sie anschließend alle Autos, die sich auf dem Parkplatz befinden, auf der Konsole aus.

g) Ergänzen Sie den noch fehlenden Code der Klasse Fuhrpark und testen Sie das Verwaltungsprogramm mit den obigen sechs Autobeispielen und weiteren Beispielen. Vermieten Sie Autos, lassen Sie welche zurückgeben. Entscheiden Sie, ob zurückgegebene Fahrzeuge verschrottet oder weiter vermietet werden.

2. **OOP - Modellierung / Collection**
Eine Firma möchte sich durch ein Software-System in die Lage versetzen, ihre Telefonanlage

selbst zu verwalten. Die Telefonanlage hat eine *Vorwahl* und eine *Anlagennummer* (zum Beispiel Universität Leipzig: 0341 97). Jedes Telefon der Firma wird von der Telefonanlage gewartet. Es besitzt eine *Nebenstellennummer* aus genau 5 Ziffern (zum Beispiel Prüfungsamt der Fakultät: 32165) und einen *Aufenthaltsort* (zum Beispiel Raum: A 510).
Erstellen Sie ein **UML-Klassendiagramm**, einschließlich der Beziehungen (Vererbung, Assoziation, Aggregation, Komposition, Multiplizitäten) untereinander, für die folgenden beiden Szenarien:

a) Modellieren Sie eine ***Klasse Telefonanlage*** und eine ***Klasse Telefon*** mit den folgenden Funktionalitäten:
Telefone werden bei der Telefonanlage an- bzw. abgemeldet. Die Anlage verwaltet dabei die Nebenstellennummern.

b) Aufgrund akuter Mittelknappheit der Firma wird folgende Einschränkung für das Telefonieren festgelegt:
Jeder Mitarbeiter kann über eine vor der eigentlichen Telefonnummer gestellte dreistellige Identizitätskennung, die durch # abgeschlossen wird, von jedem Telefon aus anrufen (zum Beispiel: 123 # 97 32165). Er erhält ein monatliches Guthaben. Beim Telefonieren werden die vom Telefon registrierten Einheiten vom Guthaben abgebucht. Sind die Einheiten verbraucht, so wird er von der Telefonanlage für weitere Telefonate gesperrt. Am ersten Wochentag des neuen Monats werden alle Guthaben wieder auf einen festgelegten Wert aufgeladen.
Führen Sie eine neue ***Klasse Mitarbeiter*** ein, in welcher der *Namen, Vornamen*, die *Identizitätskennung* und das *aktuelle Guthaben* vermerkt wird. Erweitern Sie die Klassen *Telefonanlage* und *Telefon* entsprechend zur Verwaltung der Mitarbeiter und ihrer Guthaben.

c) Verwenden Sie in der Klasse *Telefonanlage* zur Speicherung der *Telefone* und der *Mitarbeiter* jeweils eine **Collection-Klasse**. Begründen Sie Ihre Auswahl in maximal »zwei Sätzen«.