Übungen zur Vorlesung

Softwaretechnologie

-Wintersemester 2010/2011-

Dr. Günter Kniesel

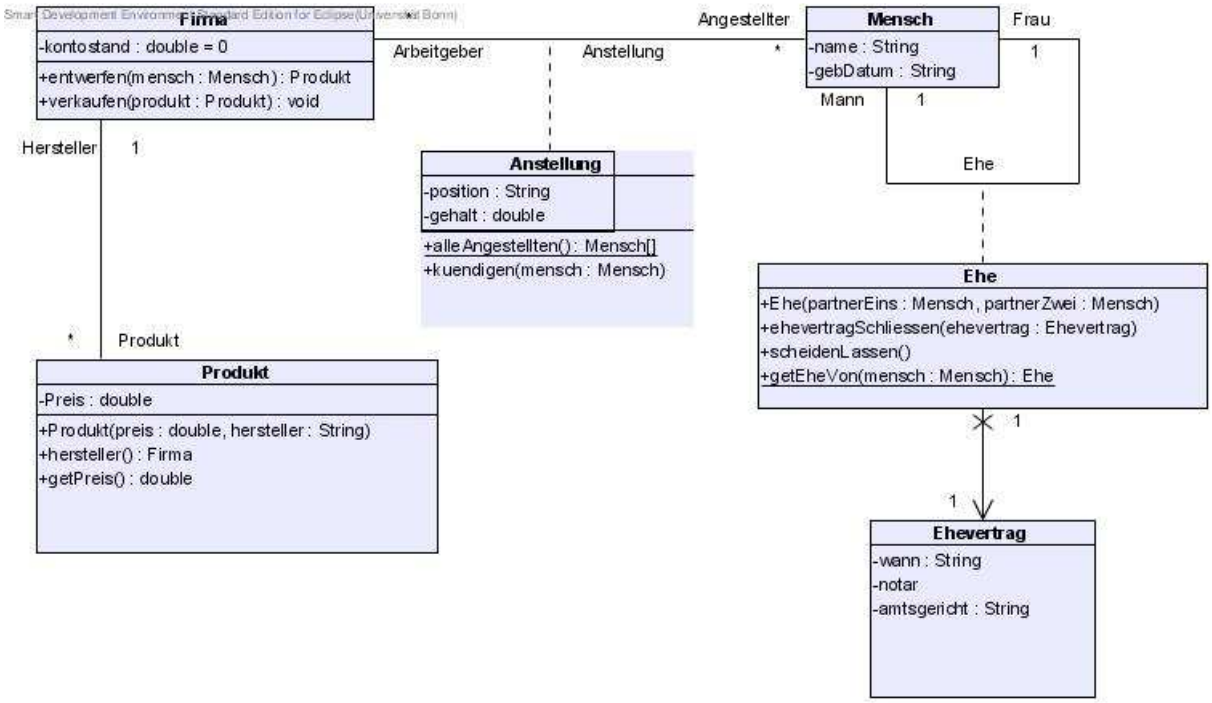
Übungsblatt 12

Zu bearbeiten bis: 23.01.2011

Bitte fangen Sie **frühzeitig** mit der Bearbeitung an, damit wir Ihnen bei Bedarf helfen können. Checken Sie die Lösungen zu den Aufgaben in Ihr SVN-Repository ein, Texte als Textdatei.

Aufgabe 1. *Implementierung von Assoziationen* (13 Punkte)

Ihr Teamleiter hat Ihnen folgenden Entwurf in die Hand gedrückt mit dem Auftrag ihn zu implementieren. Netterweise hat er Ihnen auch ein komplettes UML-Modell dazu ins SVN-Repository hochgeladen, unter gruppe/share/blatt12/blatt12-aufgabe1-modell.vpp.



a) (*2 Punkte*) Nutzen Sie Paradigm, um aus dem UML-Modell ein Code-Gerüst zu generieren (siehe Menuepunkt „InstantGenerate“).

b) (*3 Punkte*) Schauen Sie sich den von Paradigm generierten Code genau an, vergleichen Sie die von Paradigm verwendeten Ansatz mit den im Vorlesungsskript vorgestellten Optionen und diskutieren Sie die Qualität der Umsetzung der Assoziationen in Paradigm:

   a. Was hätten Sie genauso gemacht?

   b. Was hätten Sie anders gemacht?

   Begründen Sie jeweils Ihre Meinung.

c) (*3 Punkte*) Wandeln Sie die Implementierung so ab wie Sie es unter b) vorgeschlagen haben. Falls Sie unter b) keinen Vorschlag hatten, schauen Sie im Vorlesungsskript nach einer alternativen Umsetzungsmöglichkeit nach und implementieren Sie diese. Wenn das Ganze funktioniert checken Sie es ein.

d) (*2 Punkte*) Versuchen Sie, aus Ihrem manuell bearbeiteten Code mit Paradigm wieder ein Modell zu erzeugen (Reverse Engineering). Was geschieht?

e) (*3 Punkte*) Recherchieren Sie im Internet (max. ½ Stunde pro Person): Wie steht Paradigm hinsichtlich des Reverse Engineering von automatisch generiertem und anschließend manuell veränderten Code besser im Vergleich zu anderen Werkzeugen da? Identifizieren Sie auf Basis Ihrer Recherche Unterscheidungsmerkmale und bewerten Sie Paradigm und die von Ihnen verglichenen Werkzeuge anhand dieser Merkmale.

Aufgabe 2. *Blackbox Test* (12 Punkte)

Gegeben sei folgende Schnittstelle:

```
interface DateParser {
     java.util.Date parseDate(String input);
}
```

und folgender Dokumentation:

> Interpretiert gutmütig einen Datumsstring der Form *Tag-Monat-Jahr*
>
> *Tag* und *Monat* können ein- order zweistellig sein, evtl. mit führender 0
> *Jahr* kann zwei- oder vierstellig sein, zweistellige Angaben beziehen sich auf das 21. Jahrhundert
>
> Wenn *Monat* kleiner *1* ist, wird Januar angenommen, bei Werten über *12* Dezember.
> Wenn *Tag* kleiner *1* ist wird der Monatserste angenommen, bei Werten größer dem zuletzt gültigen Tag der Monatsletzte.
>
> **Spezifiziert durch:** parseDate(...) in DateParser
> **Parameter:**
> **input** Der Datumsstring, der interpretiert werden soll
> **Liefert zurück:**
> Das interpretierte Datum (mit dem Uhrzeitwert 12:00:00) oder *null* bei ungültiger Eingabe

a) Überlegen Sie, welche Äquivalenzklassen auftreten können. Geben Sie für fünf typische korrekte Eingaben und Fehlerarten einen kritischen Eingabestring und das korrekte Methoden-Ergebnis an.

b) Im SVN-Repository finden Sie das Archiv„gruppe/share/blatt12/DateParser.zip“, das eine Klasse enthält, die obiges Interface implementiert. Vervollständigen Sie mit *JUnit 4* den `GemaltoDateParserTest`, mit der Sie die Methode `parseDate` in der Klasse `GemaltoParser` testen. Setzten Sie jede in (a) identifizierte Fehlerart in einen Testfall um.
Hinweis: *Sie können in den Tests die Hilfs-Methode `makeDate(…)` verwenden.*
Bonus: Welchen Fehler hat die getestete Klasse?

c) Schreiben Sie nun eine Klasse `MyDateParserTest`, welche von `GemaltoDateParserTest` erbt, aber die `setUp`-Methode so überschreibt, dass in `systemUnderTest` eine Instanz der Klasse `MyDateParser` instanziiert wird. Entwickeln Sie nun die Klasse `MyDateParser`, indem Sie nach jedem Entwicklungsschritt den JUnit-Test `MyDateParser` ausführen, bis alle Testmethoden erfolgreich sind (Zur Dokumenta-

tion Ihrer Vorgehensweise sollen Sie nach jeder Modifikation Ihre Klasse in Ihr SVN-Repository einchecken).