PES
Technische Universität Berlin
Software Engineering for Embedded Systems Group – Prof. Dr. Sabine Glesner
www.pes.tu-berlin.de Secr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin

Softwaretechnik und Programmierparadigmen WiSe 2014/2015

Prof. Dr. Sabine Glesner
Joachim Fellmuth joachim.fellmuth@tu-berlin.de
Dr. Thomas Göthel thomas.goethel@tu-berlin.de
Lydia Mattick lydia.mattick@tu-berlin.de
Tutoren

Achtung: Denkt bitte daran, euch für die Prüfung anzumelden! Falls das über QISPOS für euch nicht möglich ist, meldet das Modul im Prüfungsamt an und gebt den gelben Zettel bei uns im Sekretariat ab.

Übungsblatt 6 **Ausgabe: 20.11. (Besprechung: 24.11. und 25.11.)**

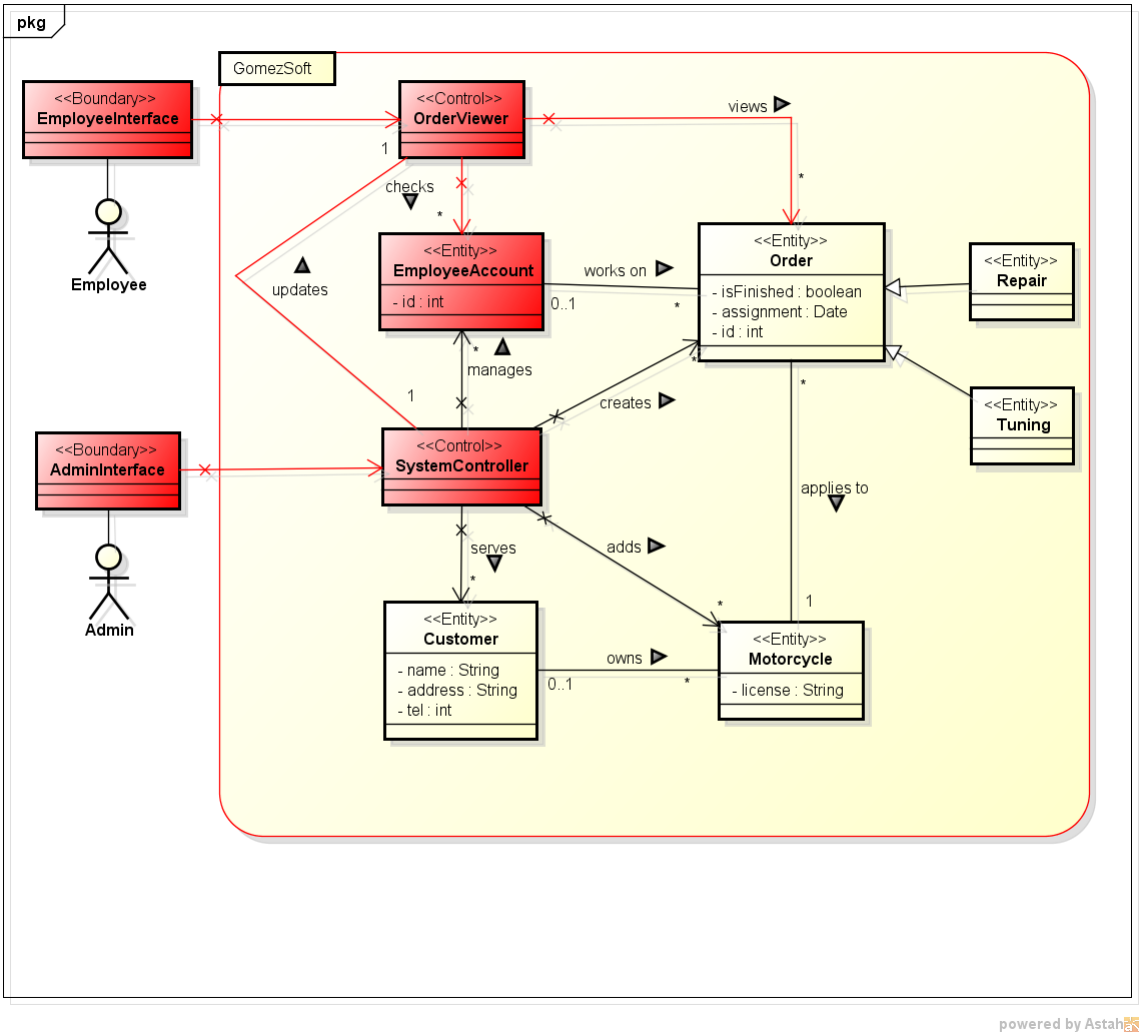
Nachtrag zu OCL

Die folgenden Mengenausdrücke sind für die Bearbeitung von Übungsblatt 4 und 5 hilfreich:

a) `asSet()` formt einen Bag (Multimenge) zu einem Set um, d.h. eliminiert doppelt vorkommende Elemente. Beispiel: Der Ausdruck `collect()` gibt einen Bag zurück.

b) `flatten()` formt eine Menge von Mengen zu einer einfachen Menge um. Bei der Operation `collect()` wird dies automatisch getan. Wenn man dies nicht will, kann `collectNested()` verwendet werden: `collect() = collectNested() -> flatten()`

c) `isInstanceOf(t: Type)` vergleicht den Typ einer Variable mit dem übergebenen Typ.



1. OCL: Invarianten

Zur Erinnerung: Invarianten sind Bedingungen von Klassen, die zu jeder Zeit gelten sollen. Formalisiert die folgenden Invarianten oder denkt euch eigene aus. Benutzt hierzu das gegebene Systemklassenmodell.

a) Ein Auftrag ist immer vom Typ Tuning oder Repair.

b) Für ein Motorrad kann immer nur höchstens ein Reparaturauftrag und ein Tuningauftrag existieren.

2. OCL: Contracts

Vor- und Nachbedingungen von Operationen beschreiben den Systemzustand und die Eingabe- bzw. Ausgabeparameter. Zusammen mit Invarianten lässt sich dadurch formal feststellen, ob z.B. eine bestimmte Abfolge von Operationen möglich ist. Formalisiert die folgenden Vor- und Nachbedingungen oder denkt euch eigene aus. Benutzt hierzu das gegebene Systemklassenmodell.
Hinweis:Ihr könnt davon ausgehen, dass alle Parameter mit Werten belegt sind.

a) Die Operation `finishOrder` erhält eine Auftrags-ID und Mitarbeiter-ID und beendet den angegeben Auftrag. Wichtig ist, dass die Order auch dem Employee zugeordnet wird sofern sie denn existiert.

b) Die Operation `createOrder` erhält einen Kundennamen, ein Motorradkennzeichen und einen Boolean zur Unterscheidung zwischen Tuning und Repair. Nach Ausführung der Operation soll das Motorrad und der Auftrag im System existieren; das Motorrad soll dem Kunden und dem Auftrag zugeordnet sein.

c) Die Operation `acceptOrder` erhält eine Auftrags-ID und Mitarbeiter-ID. Der Mitarbeiter und der Auftrag müssen existieren und der Auftrag darf noch nicht einem Mitarbeiter zugeteilt sein. Nach der Operation ist der Auftrag dem Mitarbeiter zugewiesen.

d) Die neue Operation `deleteCustomer` erhält einen Kundennamen. Sie kann nur durchgeführt werden wenn keine offenen Bestellungen für Motorräder dieses Kunden vorliegen. Es sollen auch alle diesem Kunden zugehörigen Motorräder und Bestellungen aus dem System entfernt werden.