**UTCN**

**Computer Science Department**

**Software Design 2009/2010**

**OBJECT CONSTRAINT LANGUAGE (OCL)**

**====================================================================**

Considerati diagrama de clase de mai jos:

# 

1. Formulati invarianti in OCL care specifica urmatoarele constrangeri:
   1. Atributul *valoare* a clasei *Comanda* nu poate fi negativ.

**context** *Comanda*

**inv:** valoare >= 0

* 1. Atributul *valoare* a clasei *Comanda* va fi 0 daca nu s-a comandat nici o componenta.

**Context** *Comanda*

**inv:** valoare = 0

**!!! Se va folosi o colectie pentru component comandata (componenteComandate)**

**context** *Comanda*

**inv:** componenteComandate -> isEmpty() implies valoare = 0

* 1. Valoarea atributului *valoare* a clasei *Comanda* este egala cu suma preturilor componentelor comandate.

**context:** *Comanda*

**inv:** componenteComandate **->** collect(pret) -> sum()

* 1. Instante diferite ale clasei *Componenta* au coduri diferite.

**context** Componenta

**inv:** Componenta.allInstances() -> forAll(c1, c2 | c1<> c2 **implies** c1.codComponenta <> c2.codComponenta)

1. Formulati o preconditie in OCL care specifica urmatoarea constrangere:
2. Numarul de zile cu care se poate prelungi termenul de livrare a unei comenzi trebuie sa fie pozitiv.

**context** Comanda:: prelungesteLivrare(durata: Integer)

**pre:** durata > 0

1. Formulati postconditii in OCL care specifica urmatoarele constrangeri:
2. Anularea unei comenzi implica setarea confirmarii pe false.

**Context** Comanda:: anuleaza()

**post:** confirmata = false

1. Rezultatul operatiei *aplicaReducere* a clasei *Comanda* consta in aplicarea unei reduceri specificate sumei totale a comenzii.

**context** Comanda:: aplicaReducere(reducere: Double)

**post**: valoare = valoare@pre + valoare@pre \* reducere

1. Rezultatul operatiei *aprobaPlataRate* a clasei *Client* specifica daca un client poate plati o comanda efectuata in rate. Se considera ca un client isi poate plati componentele comandate in rate daca sunt satisfacute urmatoarele constrangeri:
   * + Clientul a luat un salar in ultimele 12 luni.
     + In fiecare din cele 12 luni, salarul a fost mai mare de 2000 lei.

Sa se formuleze o postconditie in OCL pentru operatia *aprobaPlataRate* care exprima conditiile de mai sus.

**context** Client:: aprobaPlataRate(): Boolean

**post**: result = (venit.size() = 12 and venit.forAll(suma > 2000))

1. Operatia *acordaPlataRate* a clasei *Client* returneaza fie o instanta a clasei *PlataRate* (daca operatia a avut loc cu succes), fie null. Preconditiile operatiei trebuie sa exprime faptul ca argumentele operatiei au valori valide. Efectul operatiei consta din urmatoarele aspect:

* Daca clientului i se aproba plata comenzii in rate (conditiile de la punctul d sunt satisfacute) atunci se creeaza un nou obiect de tipul *PlataRate*, atributele sale vor lua valorile date ca si argument operatiei *acordaPlataRate* si obiectul va fi legat de client.
* In cazul in care clientului nu i se aproba plata comenzii in rate nu se creeaza nici un obiect de tipul *PlataRate.*

Sa se formuleze in OCL o specificatie pentru operatia *acordaPlataRate* care exprima conditiile de mai sus.

**context** Client:: acordaPlataRate(val: Double, nrRate: Integer): PlataRate

**pre:** valoare > 0 and nrRate > 0

**post: if** self.aprobaPlataRate()

1. Extindeti diagrama de clase cu o noua clasa *Sistem.* Un sistem poate fi format din una sau mai multe componente. Scrieti codul Java / C# aferent diagramei de clase; adaugati daca este necesar noi atribute / operatii.

**Bibliografie**

* Software Design Course
* OMG OCL Specification v 2.0, <http://www.omg.org/technology/documents/formal/ocl.htm>