Aufgabe 9 OCL

Annahmen:
- es wird = anstatt == verwendet
- mehrere Invarianten oder Conditions werden benannt und jeweils mit inv, pre oder post gekennzeichnet

Methoden:

fuegeZuVerkaufsgruppeHinzu(Verkaufsgruppe vg, Mitarbeiter ma){}
Funktion:
einen Mitarbeiter zu einer Verkaufsgruppe hinzufügen
Vorbedingung:
Mitarbeiter existiert | gegeben, da Mitarbeiterobjekt übergeben wird --> automatisch erfüllt
Verkaufsgruppe existiert | gegeben, da Vekaufsgruppenobjekt übergeben wird --> automatisch erfüllt
Verkaufsgruppe ist noch nicht voll
Mitarbeiter ist nicht in Verkaufsgruppe
Mitarbeiter ist nicht Verkaufsgruppenleiter
Nachbedingung:
Mitarbeiter ist in Verkaufsgruppe

OCL:
```
context Mitarbeiterverwaltung::fuegeZuVerkaufsgruppeHinzu(vg: Verkaufsgruppe, ma: Mitarbeiter)
  pre verkaufsGruppeNichtVoll:
  vg.mitglieder->size()<5

  pre nichtInVerkaufsgruppe:
  vg.mitglieder->
    forall(mg  | mg.id <> ma.id))

  pre nichtLeiter:
  vg.leiter->
    forall(l | l.id <> ma.id )

  post:
  vg.mitglieder->
    includes(ma)

# Annahme: sowohl mitglieder und mitgliedVon werden als Attribute gespeichert
# Mitarbeiter-Verkaufsgruppen Zuordnung ist somit doppelt vorhanden
# Um Probleme zu vermeiden, wird Bedingung gesetzt, dass ein zu löschender Mitarbeiter nicht Mitglied einer Verk
aufsgruppe sein darf
# Bei der Leitung wird gleich vorgegangen
```

loescheMitarbeiter(long id){}
Funktion:
Löscht einen Mitarbeiter aus dem System
Vorbedingung:
Mitarbeiter existiert
Mitarbeiter ist nicht Mitglied einer Verkaufsgruppe
Mitarbeiter ist nicht Verkaufsgruppenleiter
Nachbedingung:
Mitarbeiter ist weg (keine ID mehr vorhanden)

```
context Mitarbeiterverwaltung::loescheMitarbeiter(id: long)
```

```
   pre existiert:
   Mitarbeiter.allInstances->
      exists(ma | ma.id = id)

   pre nichtInVerkaufsgruppe:
   Verkaufsgruppe.allInstances->
      forall(vg | vg.mitglieder->
         forall(mg | mg.id <> id))

   pre nichtLeiter:
   Verkaufsgruppe.allInstances->
      forall(vg | vg.leiter.id <> id)

   post:
   Mitarbeiter.allInstances->
      forall(ma | ma.id <> id)


# Annahme: es werden sowohl Verkaufsgruppenleiter, als auch Verkaufsgruppe erzeugt
erstelleVerkaufsgruppenleiter(String n, String ad, LocalDate gb){ return Mitarbeiter}
 Funktion:
       Erstellt einen neuen Mitarbeiter
       Erzeugt eine ID
       Erstellt neue Verkaufsgruppe und macht den neuen Mitarbeiter Leiter
 Vorbedingung:
       #Annahme: Mitarbeiter dürfen die exakt gleichen Namen, Adressen und Geburtstaden haben, da sie durch die i
d unterschieden werden
       # Daher keine Vorbedingung
 Nachbedingung:
     Mitarbeiter existiert
       Mitarbeiter ist Leiter einer Verkaufsgruppe
       Anzahl der Verkaufsgruppen um 1 gestiegen

#Alter wird berechnet <-- unklar wie darzustellen
context Mitarbeiterverwaltung::erstelleVerkaufsgruppenleiter(n: String, ad: String, gb:LocalDate) Mitarbeiter
   post existiert:
   Mitarbeiter.allInstances->
      exists(ma | ma.name = n, ma.adresse = ad, ma.geburtstag = gb)

   post istLeiter:
   result.leiterVon <> null # result greift auf den neu erzeugten Mitarbeiter zu

   post neueVerkaufsgruppe:
   Verkaufsgruppe.allInstances->size() = Verkaufsgruppe.allInstances->size()@pre + 1



Klasse Mitarbeiter
 Invariante:
     Nicht gleichzeitig Verkaufsgruppenleiter und Mitglied der selben Gruppe

#Annahme: man kann ganze Objekte einfach vergleichen. Sonst schwierig, da es keine GruppenIds gibt
context Mitarbeiter
   inv: leiterVon <> mitgliedVon
```

Klasse Verkaufsgruppe
Invariante:
Mitglieder muss >= 5 sein (Auf Blatt 1 wurde noch gesagt, dass es mindestens ein Mitglied geben muss. Hier sin
d es 0 bis 5)
Muss immer einen Leiter haben

context Verkaufsgruppe
inv maxGroesse: mitglieder->size() <= 5
inv hatLeiter: leiter <> null