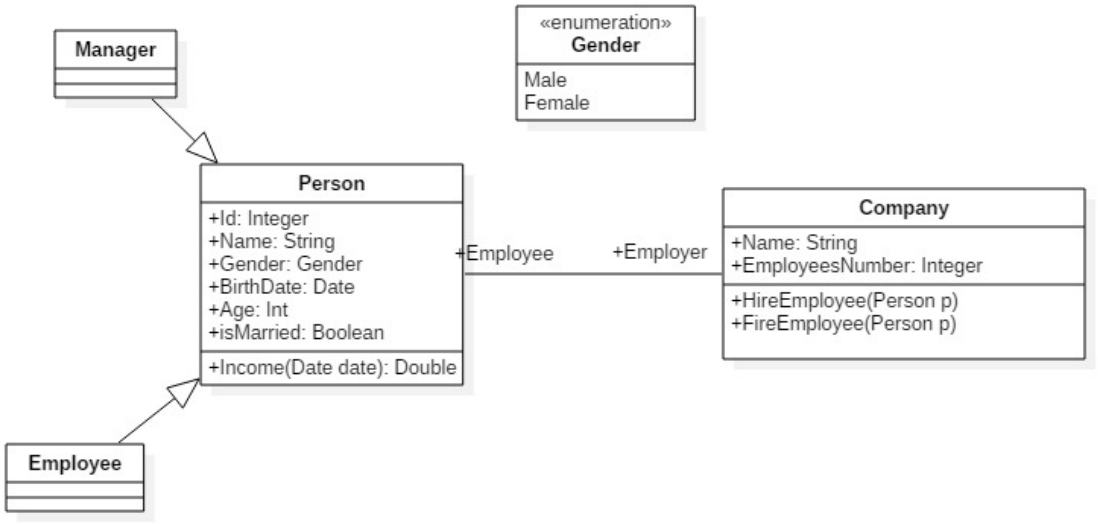
Faculty of Mathematics and Computer Science
Department of Computer Science

Object Analysis and Design (OAD)
Lab Session (TP) 02

Section 1. Object Constraint Language (OCL)

1. **Company:**
   Given the UML class diagram, express in OCL the following constraints:



   - A company must have at least 50 employee .
   - A manager must be at least 45 years old.
   - Two employees must have different IDs.
   - There exists at least two women managers
   - The procedure *FireEmployee* removes an employee from the company, the minimum number of employees is 150

2. **University:**
   Consider the previous class diagram of the university against the following description:

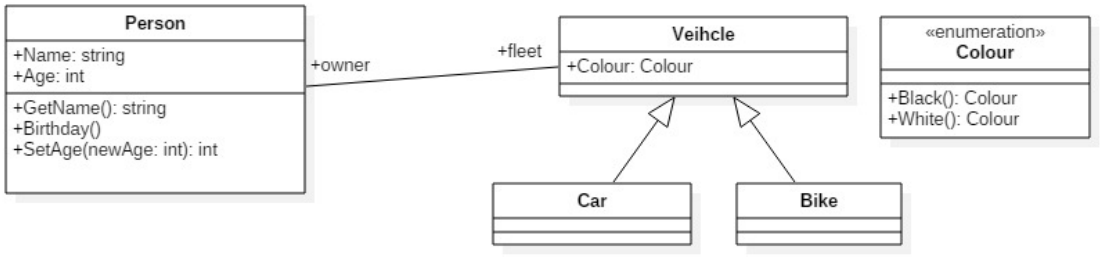
   A student is of age of (>=18). Section SIGL will consist of 20 students at most, adding students is performed by AddStudent function. After adding students, the list is subjected to some sort (SortStudentsList method), in way that despite the sort algorithm, it results in the same students and the same number. A student can choose to drop a course.
   Now, consider that when the Department wants to create a Program it guarantees that a course is not added twice, and it is assigned to one teacher. A teacher must have one of the following grades (Doctor or Professor). A teacher can teach three courses if only if he has grade Professor.

   Propose an OCL specification for this description.

3. **Vehicle Owning:**
   Consider the following class diagram:

   Express in OCL the following constraints.

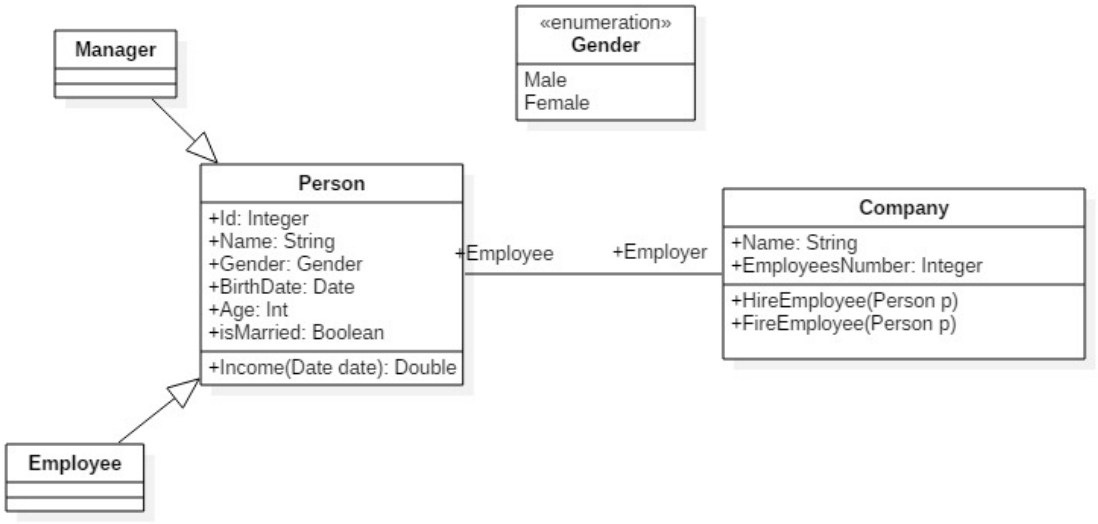
   - A vehicle owner must be at least 18 years old
   - A car owner must be at least 18 years old
   - A person younger than 18 owns no cars
   - Nobody has more than 3 vehicles
   - All cars of a person are black
   - There is a white car



Answers

Section 1. Object Constraint Language (OCL)

1. **Company:**
Given the UML class diagram, express in OCL the following constraints:



- A company must have at least 50 employee .
- A manager must be at least 45 years old.
- Two employees must have different IDs.
- There exists at least two women managers
- The procedure *FireEmployee* removes an employee from the company, the minimum number of employees is 150

Answer: context Company inv:
self.numberOfEmployees > 50
– Question 2
context Company inv:
self.Manager.Age > 45
– Question 3
context Person inv:
Person.allInstances()− >forAll(p1, p2 | p1.Id <> p2.Id)
– Question 4
context Manager
inv: Manager.allInstances()− >exists(m | m.Gender = #Female)˘ >size ≥ 2
context Company: FireEmployee(p: Person) pre: (p<>null) and Employees contains(p))
post: employees not contains(p) and (employees− >size=employees− >size@ pre-1)
inv: employee.size ≤ 150

2. **University:**
Consider the previous class diagram of the university against the following description:

A student is of age of (>=18). Section SIGL will consist of 20 students at most, adding students is performed by AddStudent function. After adding students, the list is subjected to some sort (SortStudentsList method), in way that despite the sort algorithm, it results in the same students and the same number. A student can choose to drop a course.
Now, consider that when the Department wants to create a Program it guarantees that a course is not added twice, and it is assigned to one teacher. A teacher must have one of the following grades (Doctor or Professor). A teacher can teach three courses if only if he has grade Professor.

Propose an OCL specification for this description.

Answer: Solotion 01:

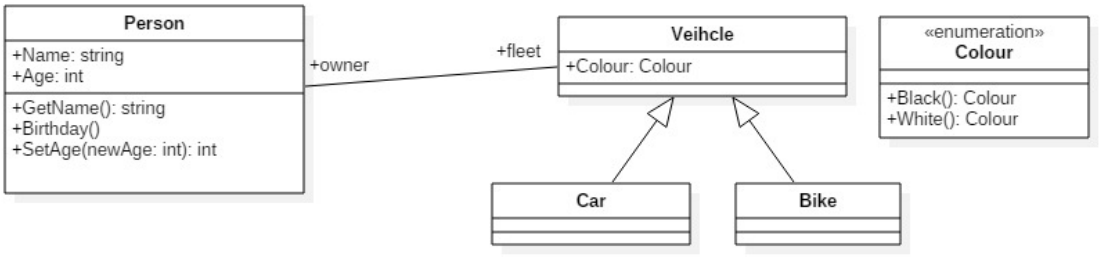
```
- - Age
context Student
inv: self.age >=18
inv: RegistartionNumber>=1
- - AddStudent
context Section::
AddStudent(s: Student)
pre: (s<>null) and not contains(s)
post: contains(s) and (size=size@pre+1)
inv: size<=20
- - SortStudentsList
context Section::
SortStudentsList(s: Sequence(Student)) : Sequence(Student)
post SameSize:
result->size = s->size
post SameElements:
result->forAll(i | result->count(i) = s->count(i))
post IsSorted:
Sequence{1..(result->size-1)}->
forAll(i | result.at(i) <= result.at(i+1))
- - Student Drop Course
context Student:: DropCourse(C: Course) pre: (C<>null) and Courses contains(C))
post: Courses not contains(C) and (Courses->size=Courses->size@pre-1)
- - Teacher
context Teacher inv CoursesToTeach: self.Cources->size <=3
- - AffectCourses
context Department::
AffectCourse(T: Teacher, C: Course)
pre: (T<>null) and T.Courses not contains(C) and (C.Teacher = null)
post: T.Courses contains(C) and (T.Courses->size=T.Courses->size@pre+1)
inv: T.Courses<=3
- - CreateProgram
context Department::
CreateProgram(P: ProgramOfCourses, C: Course)
pre: (P<>null) and (P.Courses not contains(C))
post: P.Courses contains(C) and (P.Courses->size=P.Courses->size@pre+1)
inv: P.Courses<=10
```

3. **Vehicle Owning:**
Consider the following class diagram:

Express in OCL the following constraints.

- A vehicle owner must be at least 18 years old
- A car owner must be at least 18 years old

- A person younger than 18 owns no cars
- Nobody has more than 3 vehicles
- All cars of a person are black
- There is a white car



Answer: Solotion 01:

- - Age
context Vehicle
inv: self. owner. age >= 18
- - Age
context Car
inv: self. owner. age >= 18
- - Age
context Person
inv: age<18 implies self.fleet–>forAll(v | not v.oclIsKindOf(Car))
- - Vehicles number
context Person
inv: self.fleet–>size <= 3
- - Vehicle color
context Person
inv: self.fleet–>forAll(v | v.colour = #black)
- - Vehicle black number
context Person
inv: self.fleet–>select(v | v.colour = #black)–>size <= 3
- - Vehicle black number
context Person
inv: self.fleet–>iterate(v; acc:Integer=0
| if (v.colour=#black)
then acc + 1 else acc endif) <=3
- - White car
context Car inv: Car.allInstances()->exists(c | c.colour=#white)