UML is not enough...

- Possible number of owners a car can have
- Required age of car owners
- Requirement that a person may own at most one black car
Some OCL examples I

```
Person
name:String
age:int
≪query≫
getName():String
birthday()
setAge(newAge:int):int

Vehicle
colour:Colour

Car
Bike

≪enumeration≫
Colour
black():Colour
white():Colour
red():Colour
```

“A vehicle owner must be at least 18 years old”: 
Some OCL examples I

"A vehicle owner must be at least 18 years old":

context Vehicle
inv: self.owner.age >= 18
Some OCL examples I

“A vehicle owner must be at least 18 years old”:

```ocl
context Vehicle
inv: self.owner.age >= 18
```
Some OCL examples I

“A vehicle owner must be at least 18 years old”:

context Vehicle
inv: self. owner. age >= 18
Some OCL examples I

“A vehicle owner must be at least 18 years old”:
context Vehicle
inv: self. owner. age >= 18
“A vehicle owner must be at least 18 years old”:

context Vehicle
inv: self. owner. age >= 18
“A vehicle owner must be at least 18 years old”:

```
context Vehicle
inv: self. owner. age >= 18
```

What does this mean, instead?

```
context Person
inv: self.age >= 18
```
“A vehicle owner must be at least 18 years old”:
context Vehicle
inv: self.owner.age >= 18

“A car owner must be at least 18 years old”:
context Car
inv: self.owner.age >= 18
Some OCL examples II

```
Person
name:String
age:int
<<query>>
getName():String
birthday()
setAge(newAge:int):int

Vehicle
colour:Colour

Car
Bike

<<enumeration>>
Colour
black():Colour
white():Colour
red():Colour

"Nobody has more than 3 vehicles":
```
Some OCL examples II

"Nobody has more than 3 vehicles":

context Person
inv: self.fleet->size <= 3

or change multiplicity
“All cars of a person are black”: 
Some OCL examples II

“All cars of a person are black”:

c\text{context} \text{Person} \\
\text{inv: } \text{self.fleet}\rightarrow\text{forall}(v \mid v\.\text{colour} = \#\text{black})
“All cars of a person are black”:

```ocl
class Person {
    name: String
    age: int
    get Name(): String
    birthday()
    set Age(newAge: int): int
}

class Vehicle {
    colour: Colour
    black(): Colour
    white(): Colour
    red(): Colour
}

class Car

class Bike
```

“Nobody has more than 3 black vehicles”: 

```ocl
context Person
inv: self.fleet -> forAll(v | v.colour = #black)
```
“All cars of a person are black”:

context Person
inv: self.fleet→forAll(v | v.colour = #black)

“Nobody has more than 3 black vehicles”:

context Person
inv: self.fleet→select(v | v.colour = #black)→size <= 3
What does this mean?

context Person
inv: self.fleet->iterate(v; acc:Integer=0
  | if (v.colour=#black)
  then acc + 1 else acc endif) <=3
context Person
inv: age<18 implies self.fleet→forall(v | not v.oclIsKindOf(Car))
Some OCL examples IV — oclIsKindOf

context Person
inv: age<18 implies self.fleet->forall(v | not v.oclIsKindOf(Car))

“A person younger than 18 owns no cars.”
context Person
inv: age<18 implies self.fleet--»forAll(v | not v.oclIsKindOf(Car))

“A person younger than 18 owns no cars.”

“self” can be omitted.
context Person
inv: age<18 implies self.fleet->forAll(v | not v.oclIsKindOf(Car))

“A person younger than 18 owns no cars.”

“self” can be omitted.

Logical Junctors: and, or, not, implies, if . . . then . . . else . . . endif, =
context Car
inv: Car.allInstances()->exists(c | c.colour=#red)
context Car
inv: Car.allInstances() -> exists(c | c.colour=#red)

“There is a red car.”
OCL pre-/post conditions — Examples

So far only considered class invariants.
OCL pre-/post conditions — Examples

Person

- name: String
- age: int

≪query≫
- getName(): String
- birthday()
- setAge(newAge: int): int

Vehicle

- colour: Colour

Vehicle enumeration

- black(): Colour
- white(): Colour
- red(): Colour

Car

Bike

So far only considered class invariants.

OCL can also specify operations:
So far only considered class invariants.

OCL can also specify operations:

“If setAge(...) is called with a non-negative argument then the argument becomes the new value of the attribute age.”

```
context Person::setAge(newAge:int)
pre:    newAge >= 0
post:   self.age = newAge
```
OCL pre-/post conditions — Examples

So far only considered class invariants.

OCL can also specify operations:

“Calling birthday() increments the age of a person by 1.”

context Person::birthday()
post: self.age = self.age@pre + 1
OCL pre-/post conditions — Examples

So far only considered class invariants.

OCL can also specify operations:

“Calling getName() delivers the value of the attribute name.”

context Person::getName()
post: result = name
Special to OCL are operations with a «query» stereotype:

Only these operations can be used within an OCL expression.
Special to OCL are operations with a `<<query>>` stereotype:

**Only these** operations can be used within an OCL expression.

“Calling `getName()` delivers the value of the attribute `name`.”

```plaintext
class Person {
    name: String
    age: int
    getName(): String
    birthday()
    setAge(newAge: int): int
}
```

```plaintext
class Vehicle {
    colour: Colour
    black(): Colour
    white(): Colour
    red(): Colour
}
```

```plaintext
class Car {
    Vehicle
}
```

```plaintext
class Bike {
    Vehicle
}
```

```plaintext
class Person {
    context Person
    inv: self.getName() = name
}
```