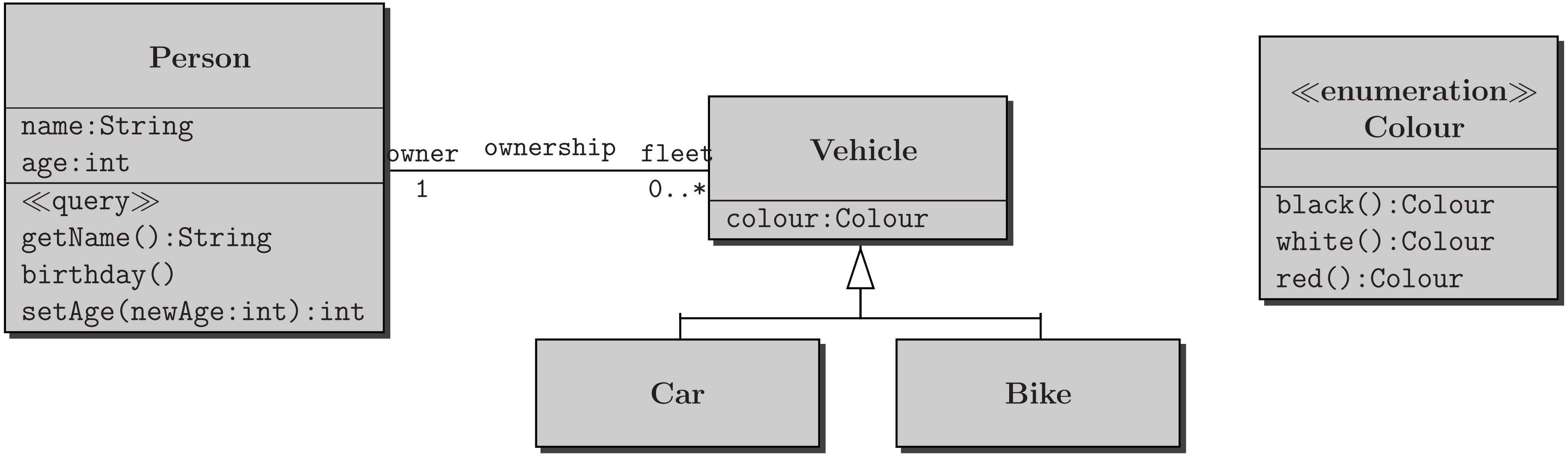
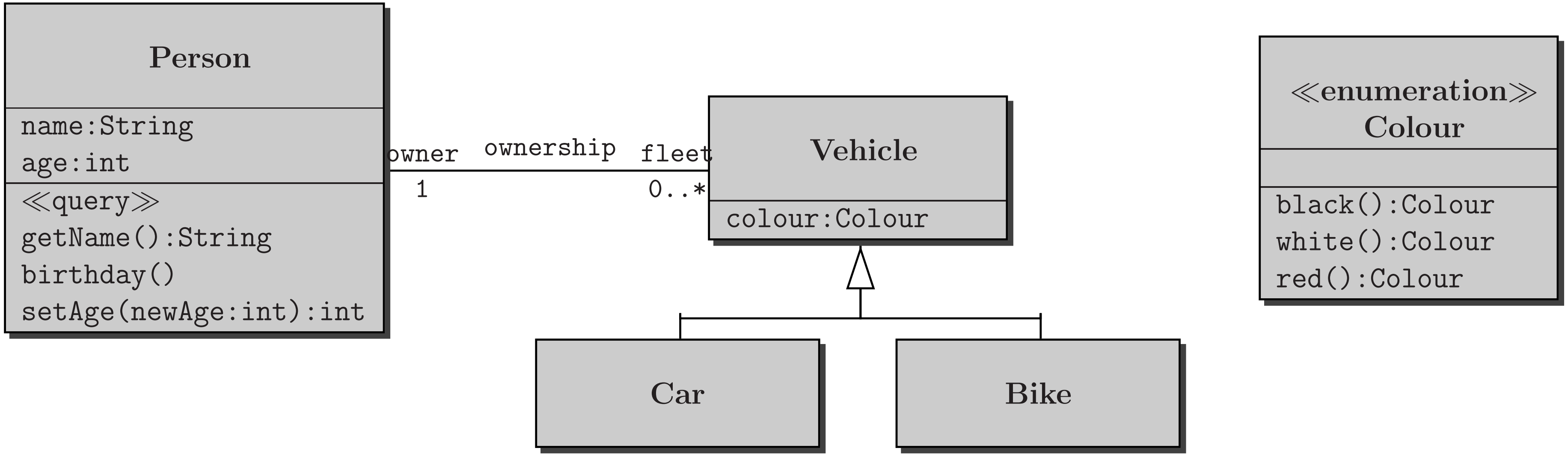
UML is not enough. . .



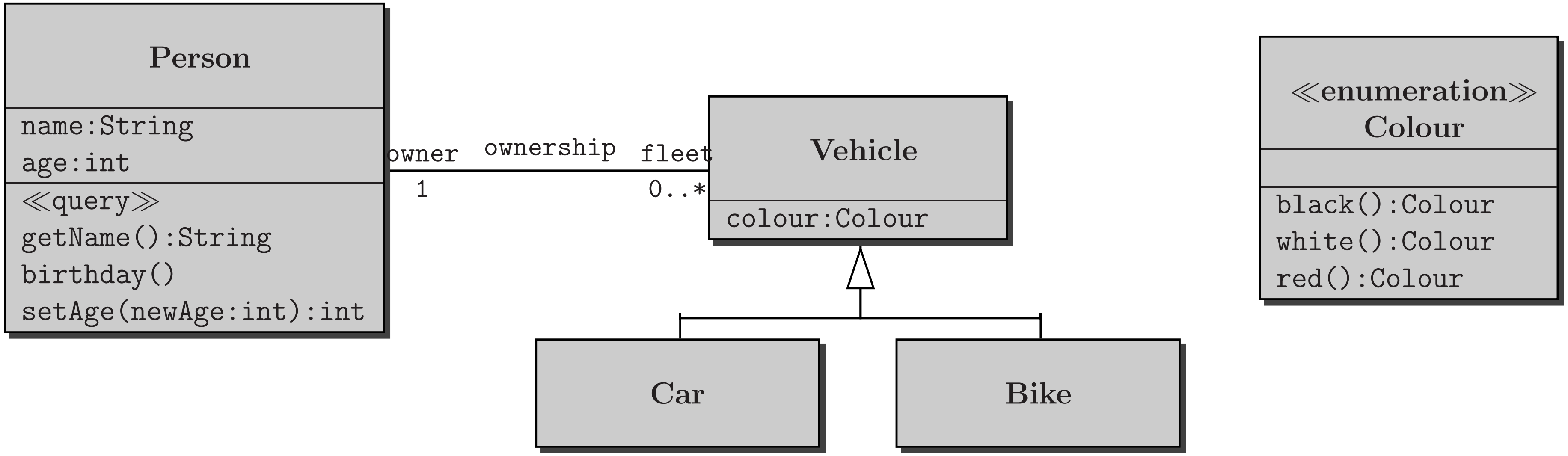
- **Possible number of owners a car can have**
- **Required age of car owners**
- **Requirement that a person may own at most one black car**

Some OCL examples I



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

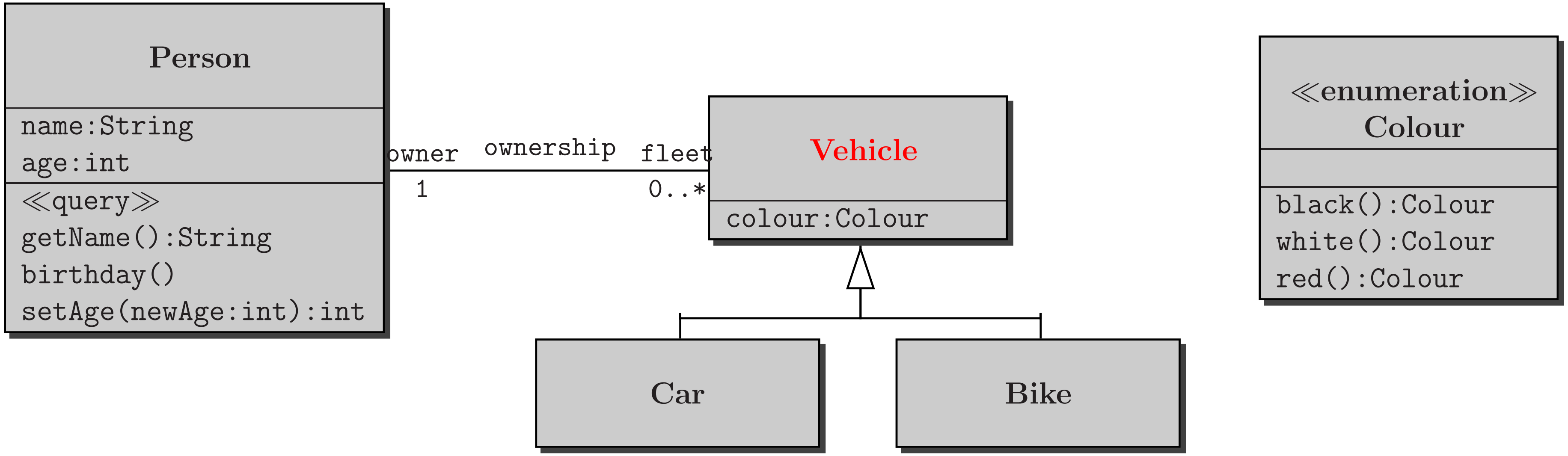
Some OCL examples I



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

context **Vehicle**
inv: **self. owner. age** $>=$ **18**

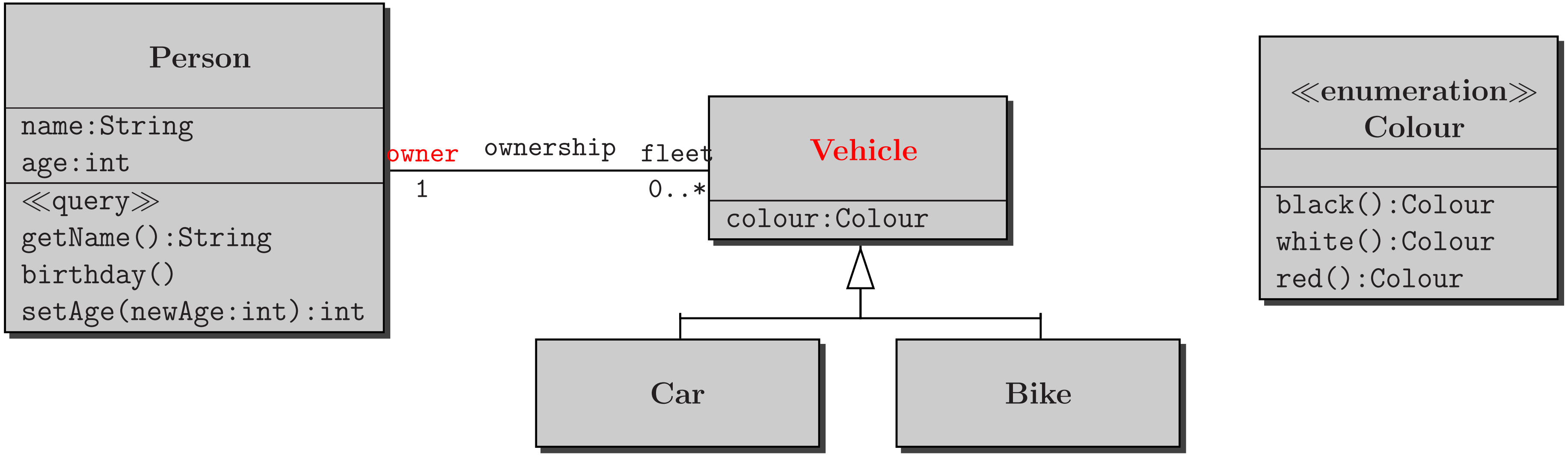
Some OCL examples I



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

context **Vehicle**
inv: **self. owner. age $>=$ 18**

Some OCL examples I

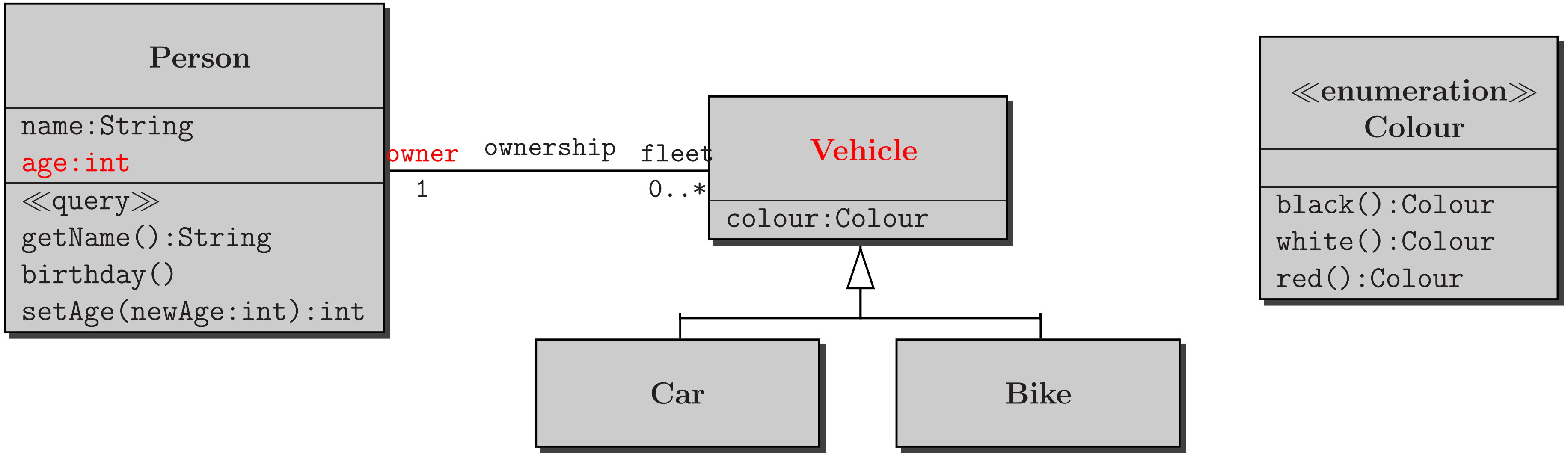


“A vehicle owner must be at least 18 years old”:

context **Vehicle**

inv: **self. owner. age $>=$ 18**

Some OCL examples I

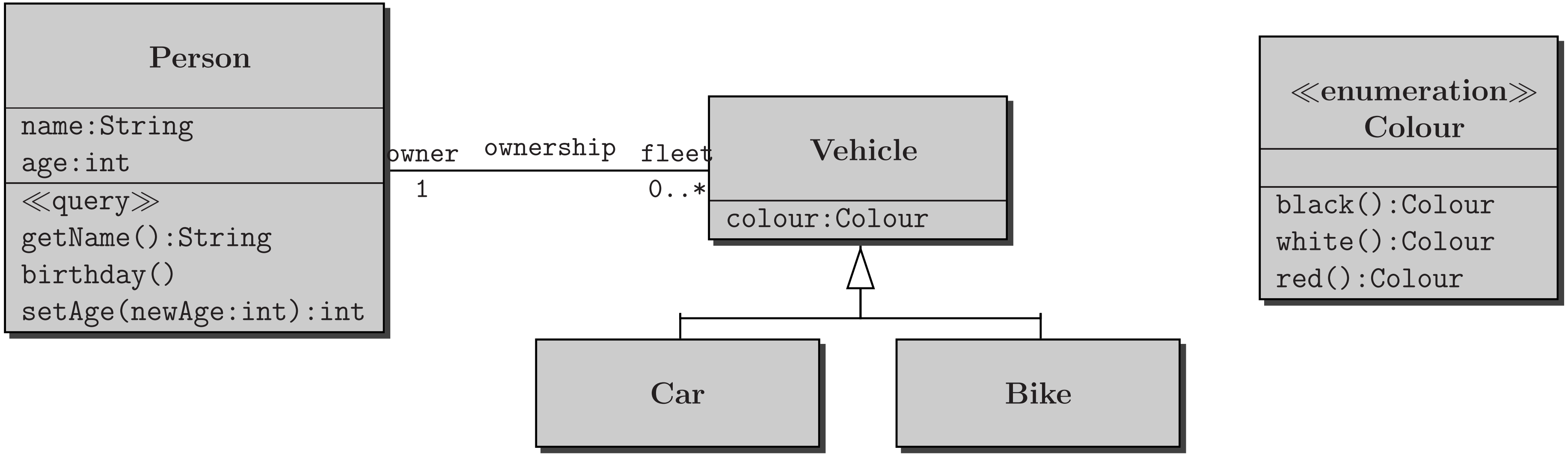


“A vehicle owner must be at least 18 years old”:

context **Vehicle**

inv: **self. owner. age $>=$ 18**

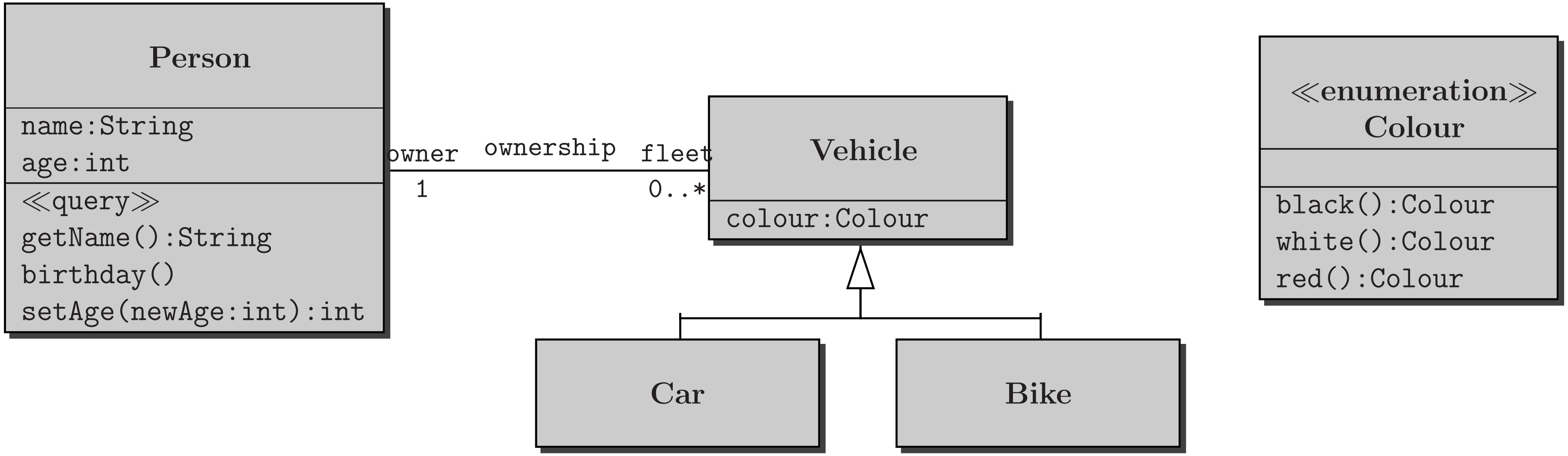
Some OCL examples I



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

context **Vehicle**
inv: **self. owner. age** $>=$ **18**

Some OCL examples I



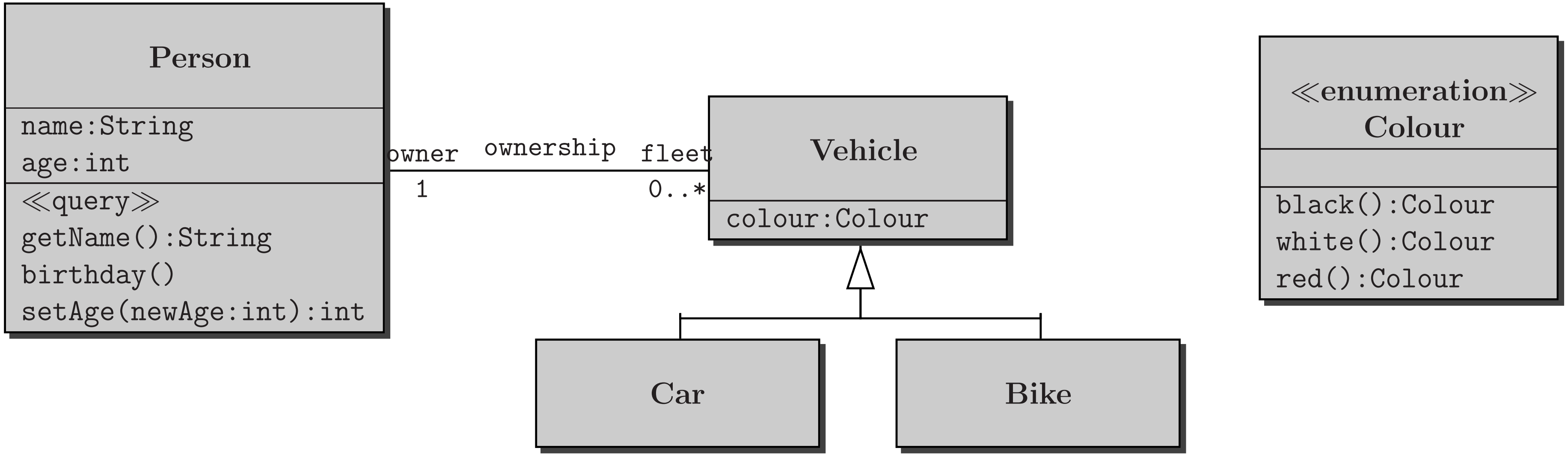
"A vehicle owner must be at least 18 years old":

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

What does this mean, instead?

context Person
inv: self.age $>=$ 18

Some OCL examples I



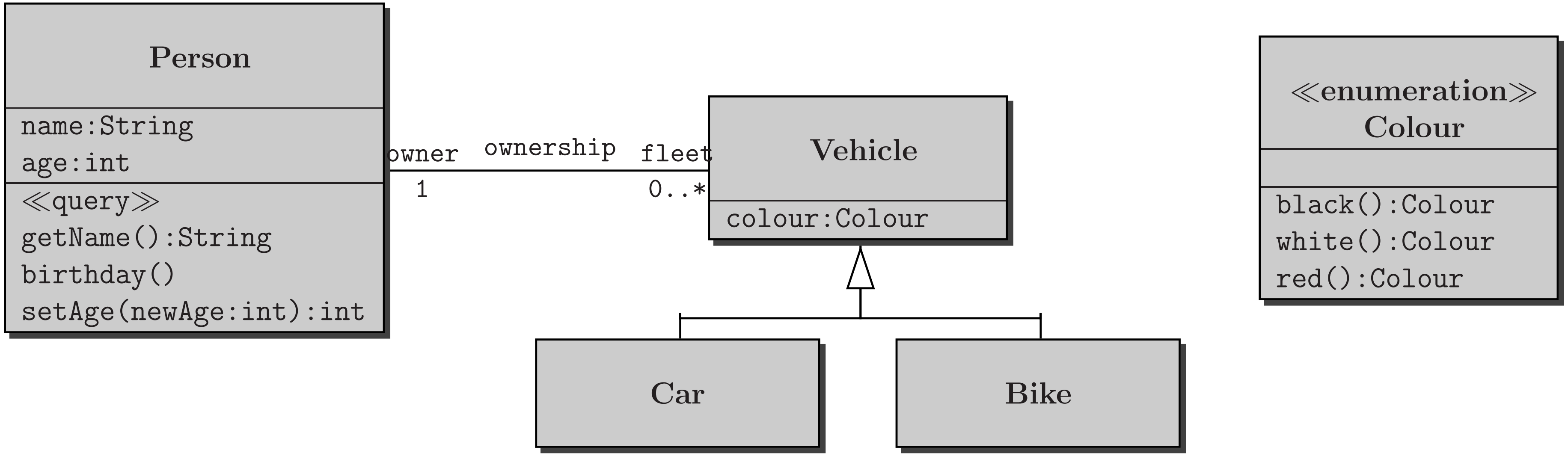
“A vehicle owner must be at least 18 years old”:

context **Vehicle**
inv: **self. owner. age $>=$ 18**

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

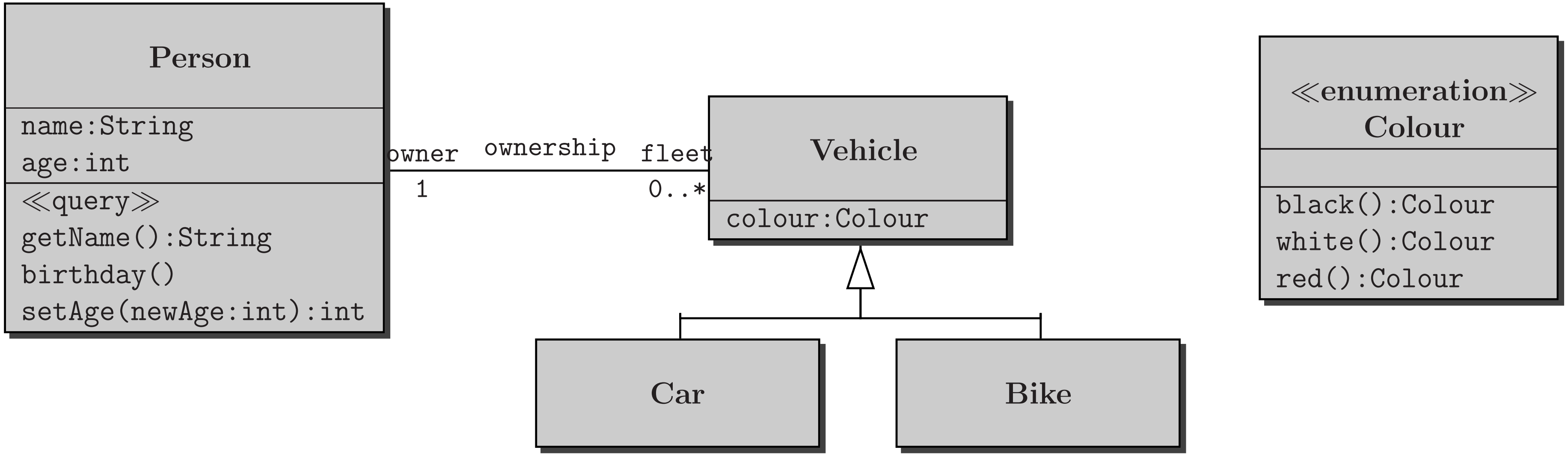
context **Car**
inv: **self.owner.age $>=$ 18**

Some OCL examples II



“Nobody has more than 3 vehicles”:

Some OCL examples II



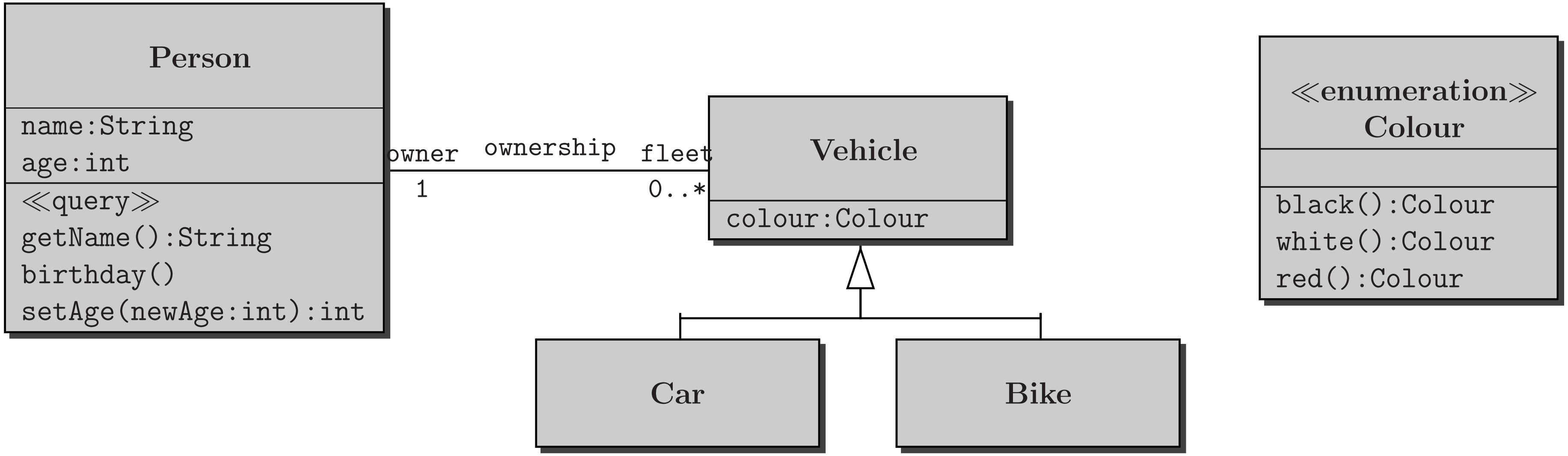
“Nobody has more than 3 vehicles”:

context Person

inv: self.fleet–>size <= 3

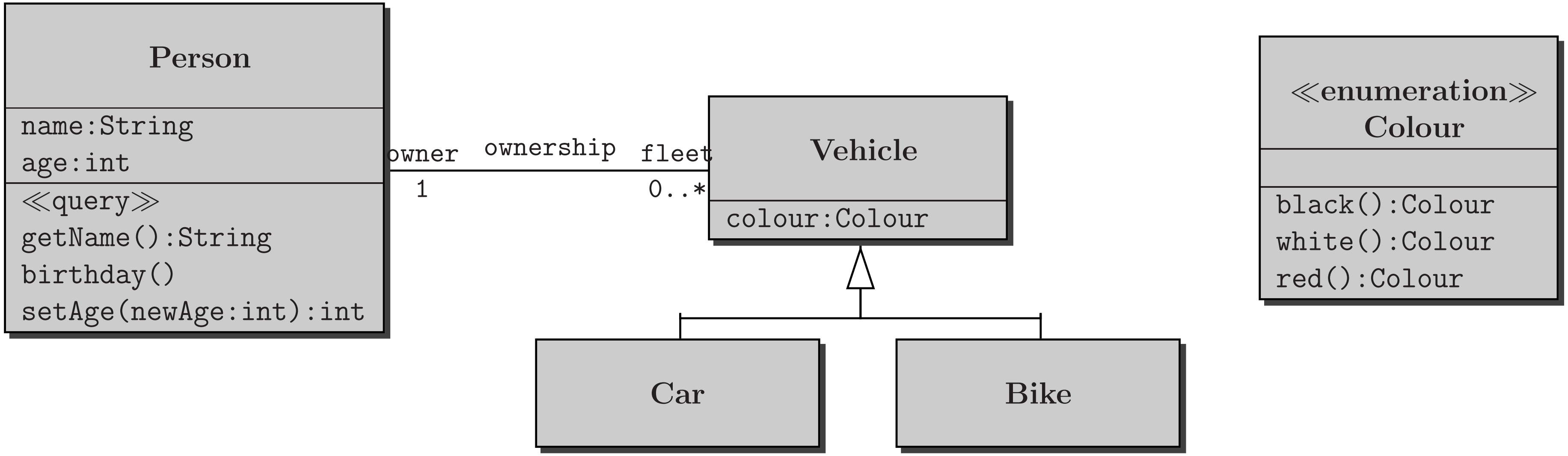
or change multiplicity

Some OCL examples II



“All cars of a person are black”:

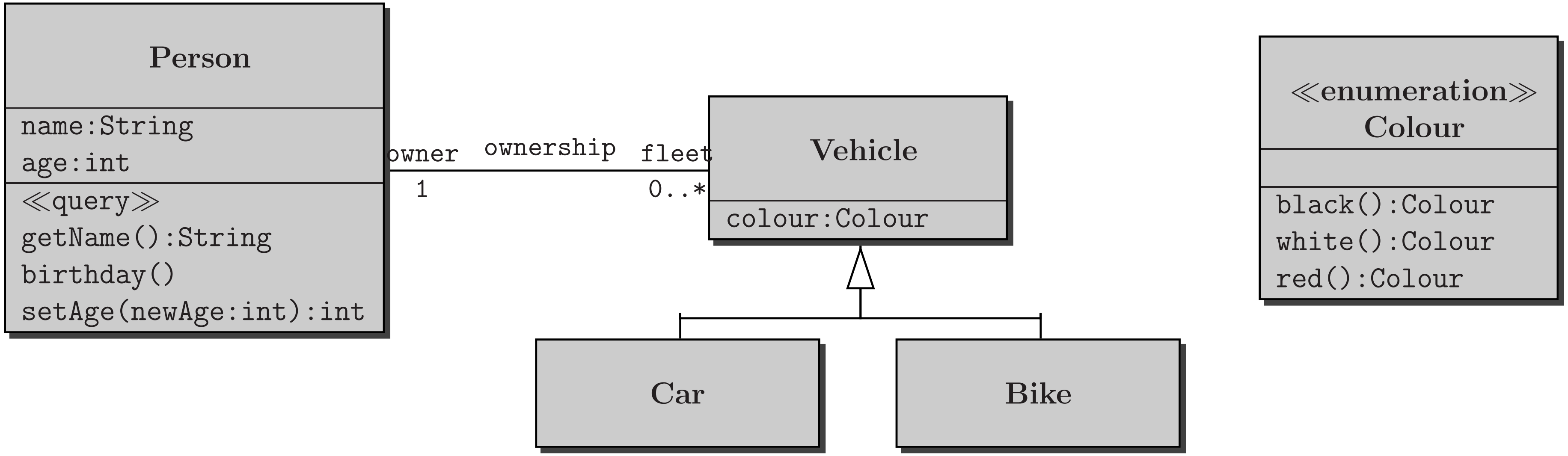
Some OCL examples II



“All cars of a person are black”:

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

Some OCL examples II

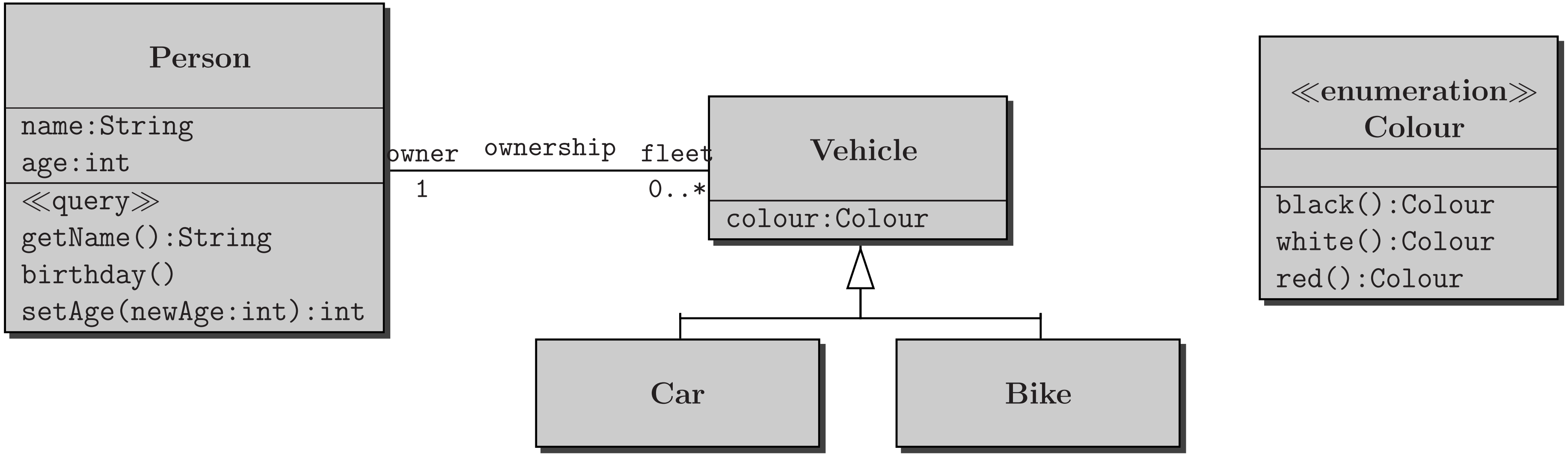


“All cars of a person are black”:

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

“Nobody has more than 3 black vehicles”:

Some OCL examples II



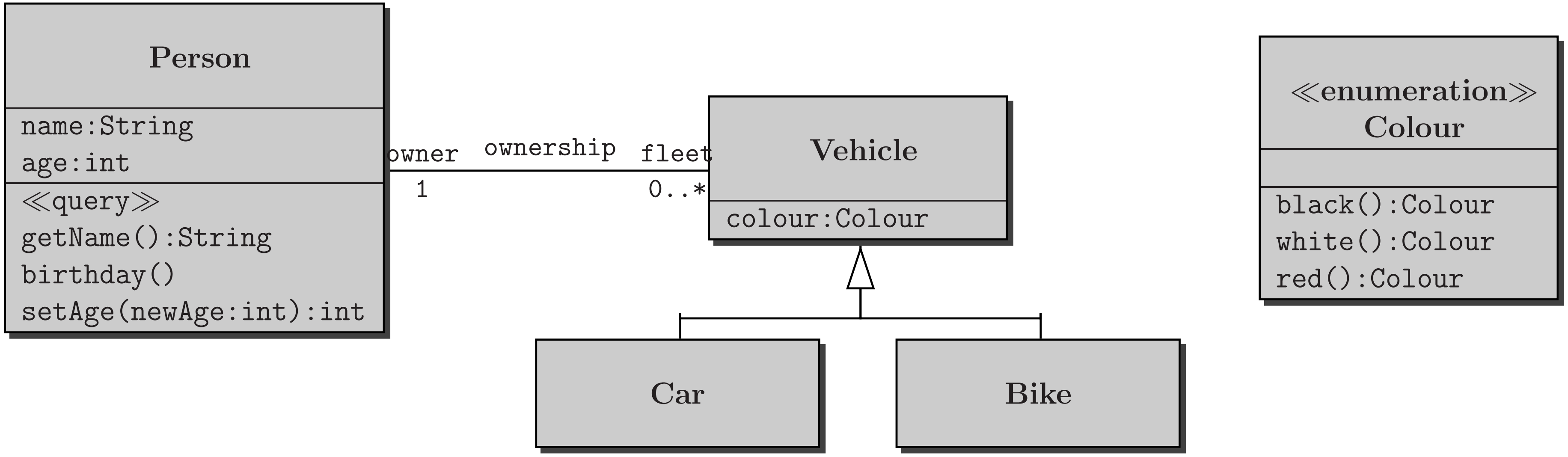
“All cars of a person are black”:

```
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
```

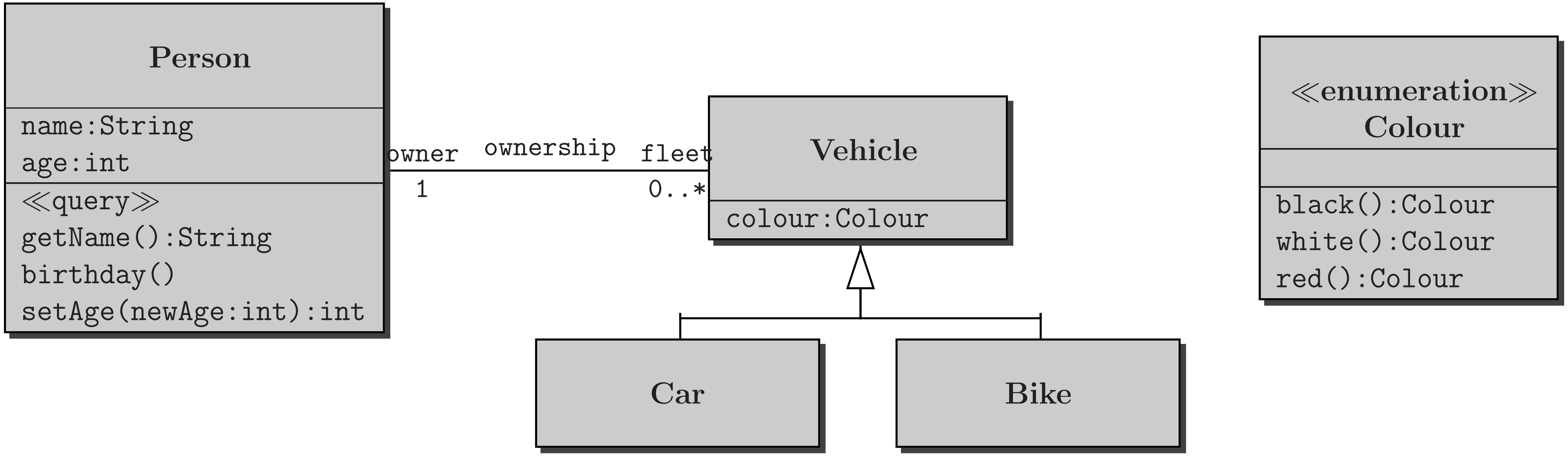
Some OCL examples III — iterate



What does this mean?

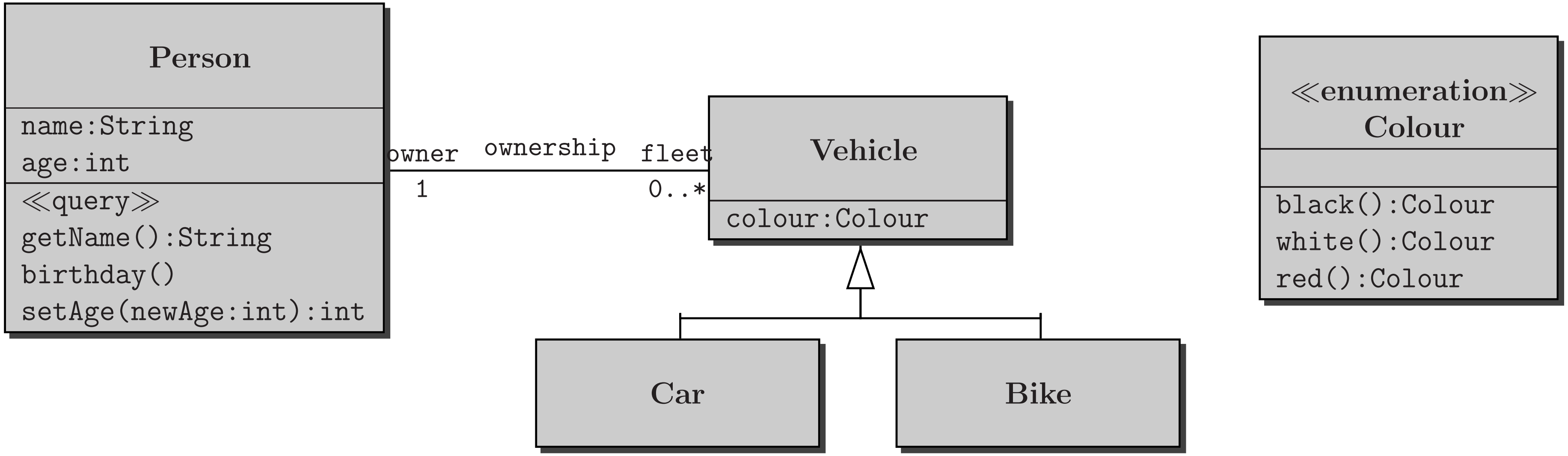
```
context   Person
inv:      self.fleet–>iterate(v; acc:Integer=0
                              | if (v.colour=#black)
                              then acc + 1 else acc endif) <=3
```

Some OCL examples IV — oclIsKindOf



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

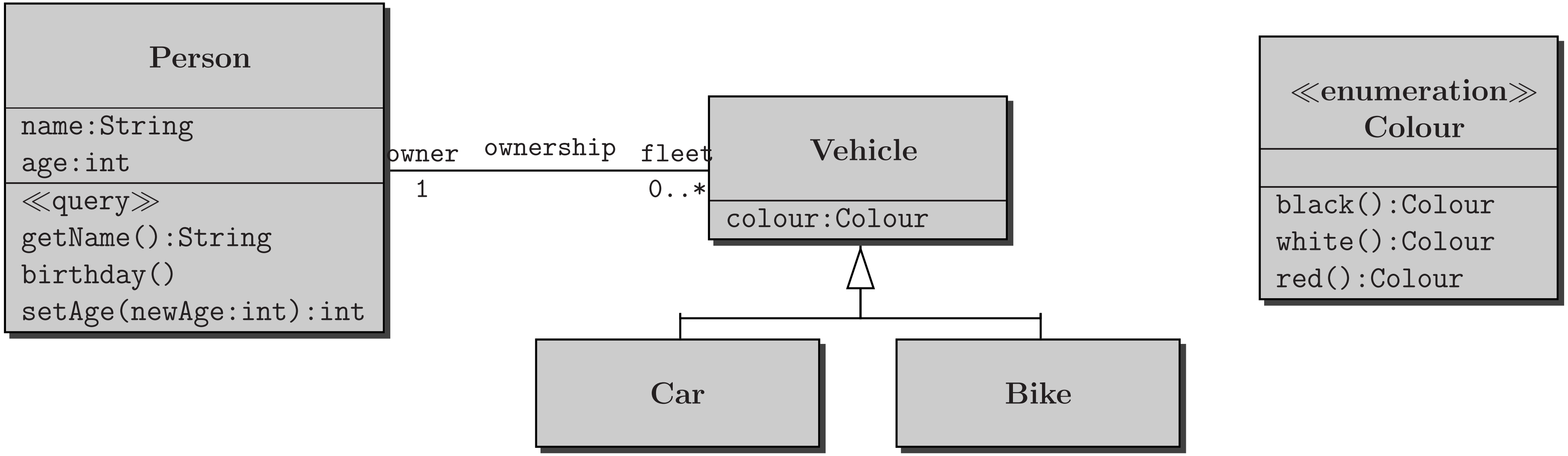
Some OCL examples IV — oclIsKindOf



context **Person**
inv: **age<18 implies self.fleet–>forAll(v | not v.oclIsKindOf(Car))**

“A person younger than 18 owns no cars.”

Some OCL examples IV — oclIsKindOf

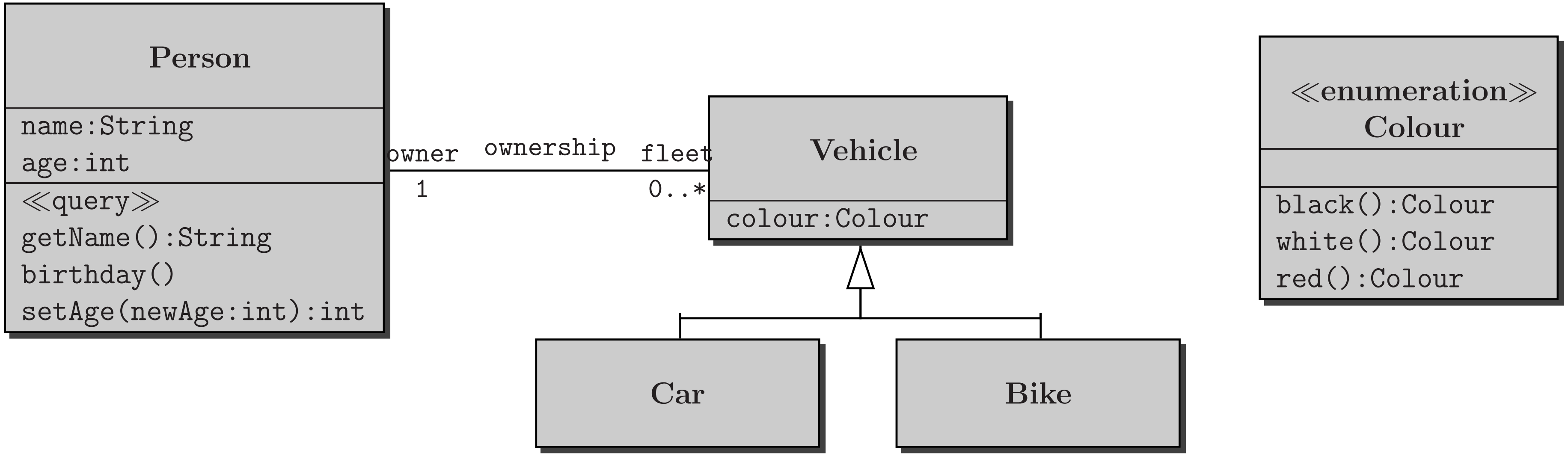


context Person
inv: age<18 implies self.fleet–>forAll(v | not v.oclIsKindOf(Car))

“A person younger than 18 owns no cars.”

“self” can be omitted.

Some OCL examples IV — oclIsKindOf



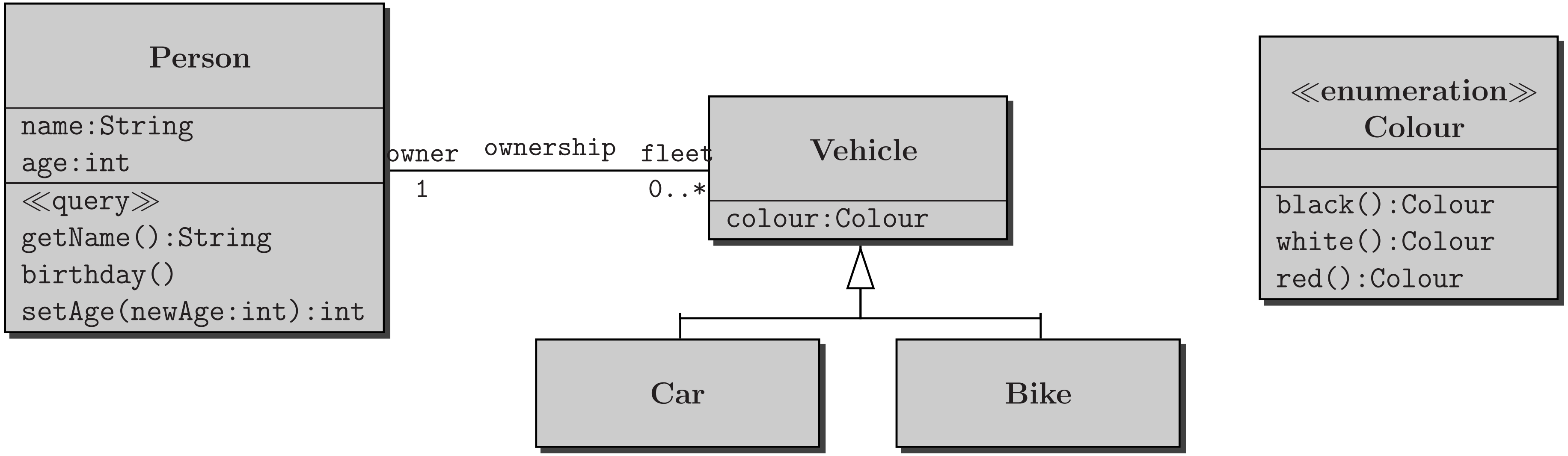
context Person
inv: age<18 implies self.fleet–>forAll(v | not v.oclIsKindOf(Car))

“A person younger than 18 owns no cars.”

“self” can be omitted.

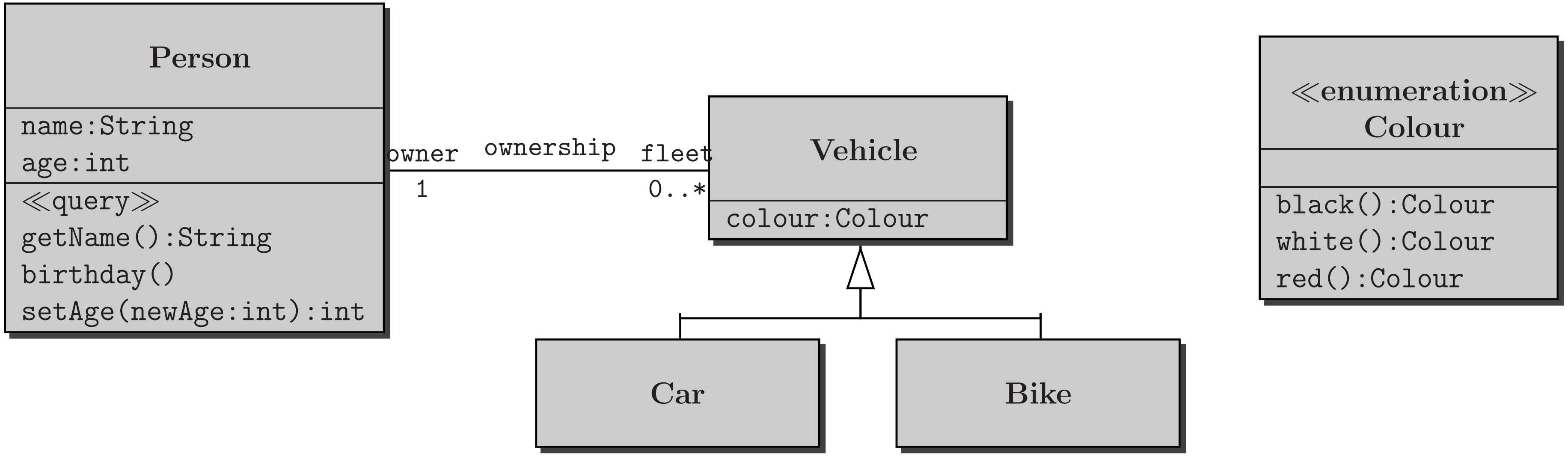
Logical Junctors: and, or, not, implies, if. . . then. . . else. . . endif, =

Some OCL examples V — allInstances



```
context   Car
inv:      Car.allInstances()->exists(c | c.colour=#red)
```

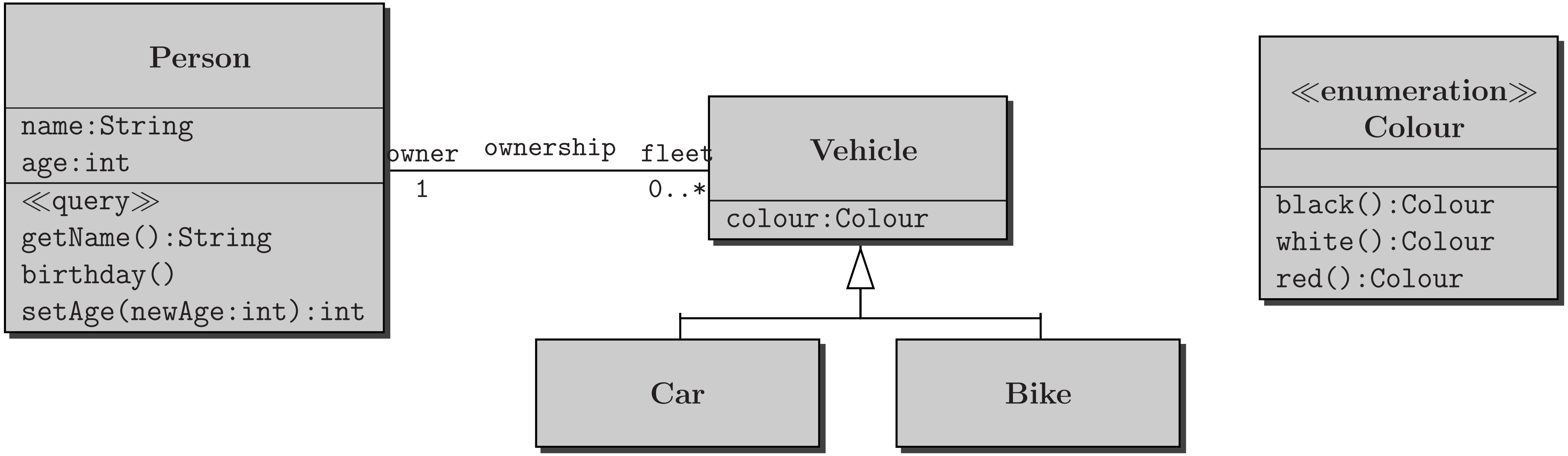
Some OCL examples V — allInstances



context **Car**
inv: **Car.allInstances()->exists(c | c.colour=#red)**

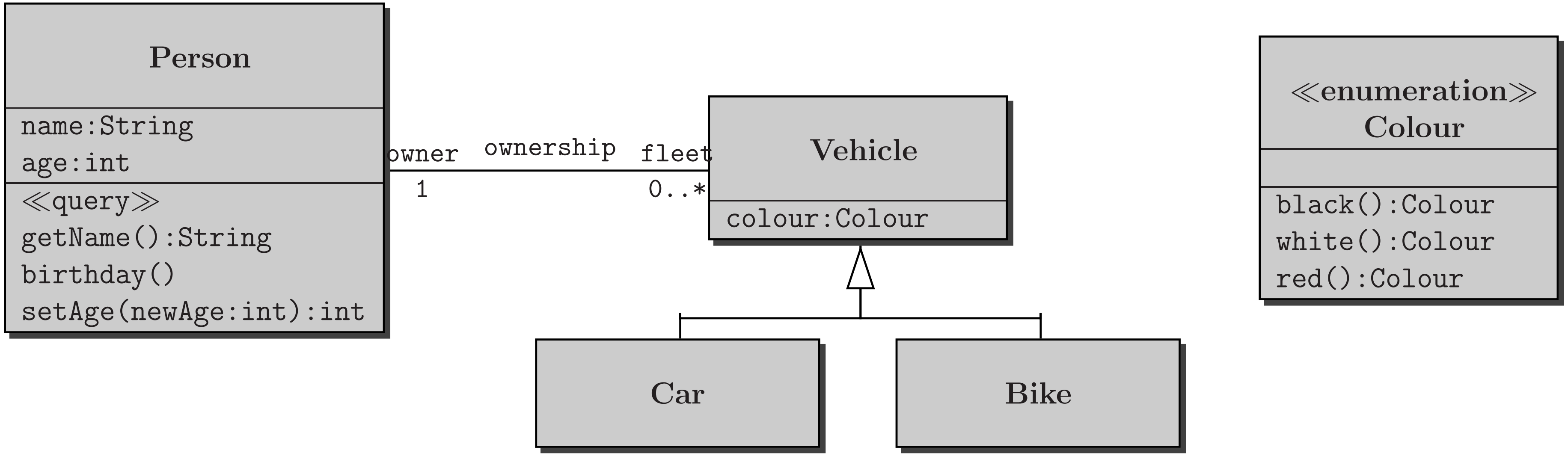
“There is a red car.”

OCL pre-/post conditions — Examples



So far only considered class invariants.

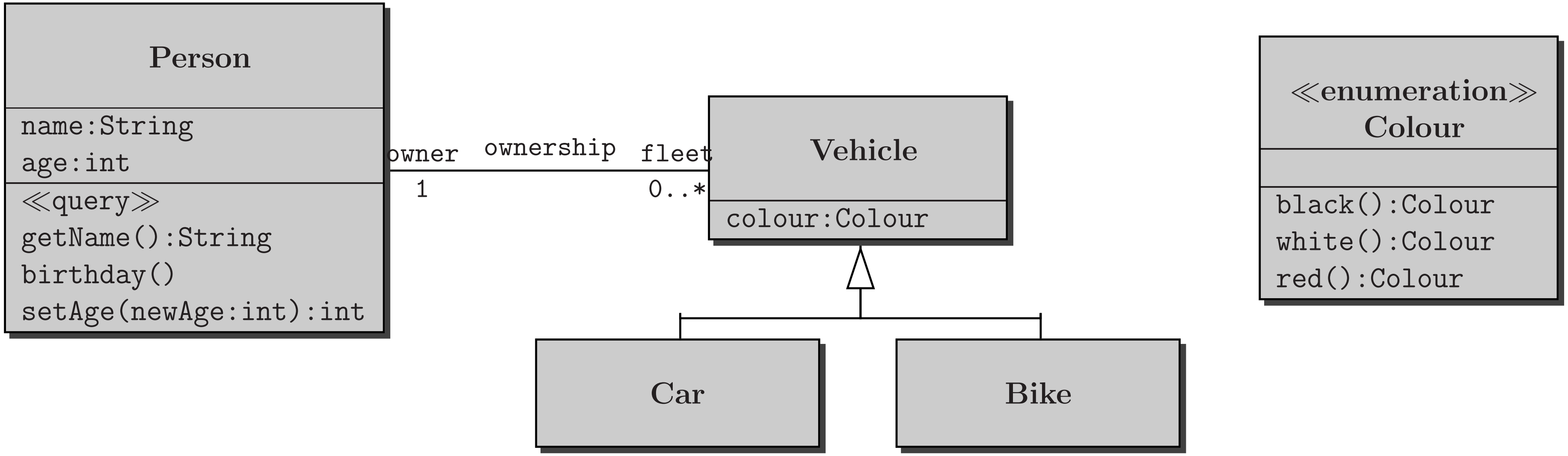
OCL pre-/post conditions — Examples



So far only considered class invariants.

OCL can also specify operations:

OCL pre-/post conditions — Examples



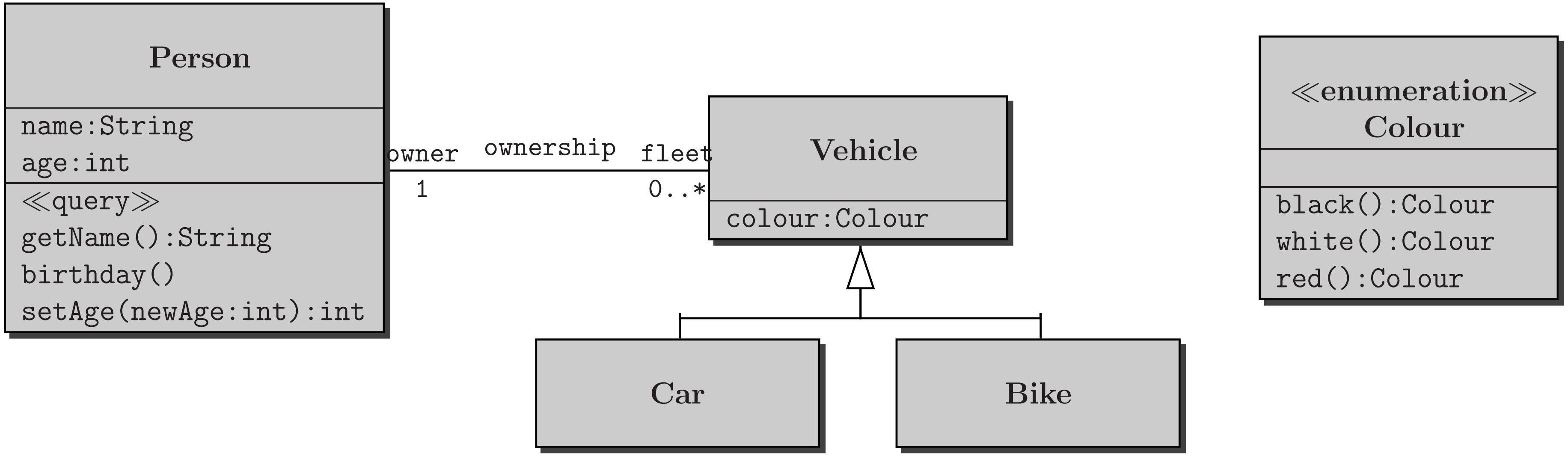
So far only considered class invariants.

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



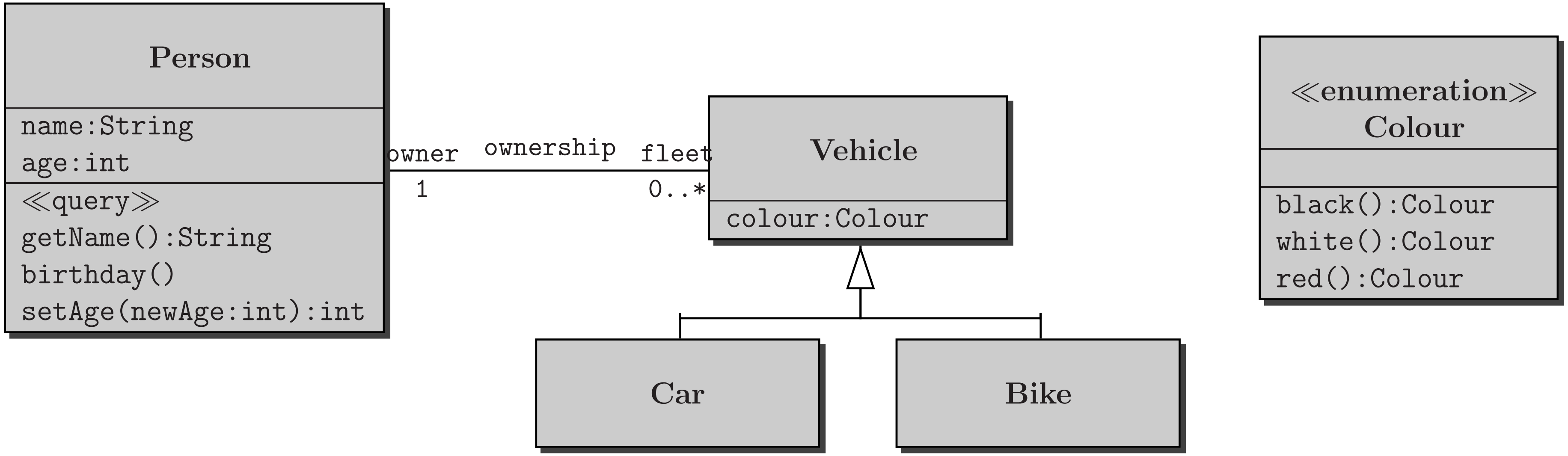
So far only considered class invariants.

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



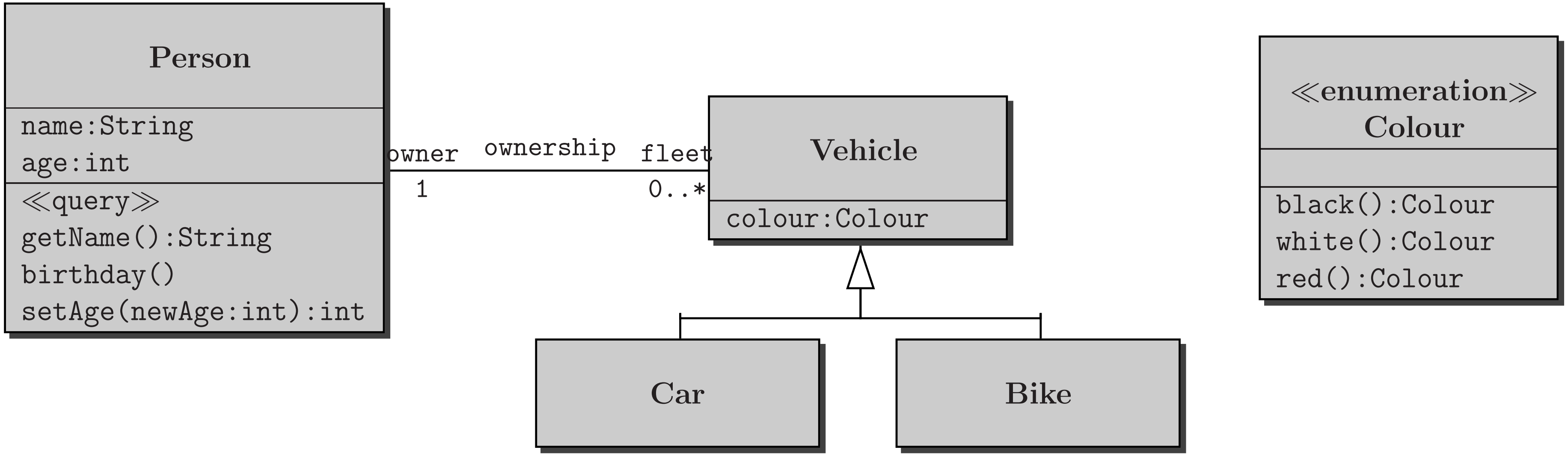
So far only considered class invariants.

OCL can also specify operations:

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

```
context   Person::getName()
post:     result = name
```

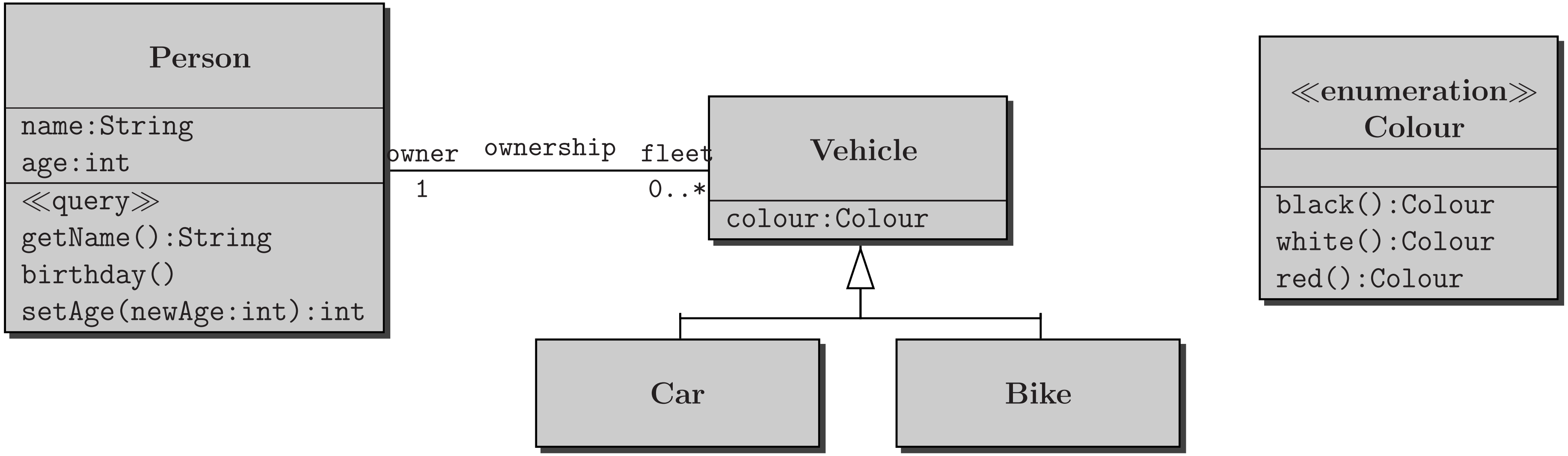
Queries



Special to OCL are operations with a ≪query≫ stereotype:

Only these operations can be used within an OCL expression.

Queries



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.”

context Person
inv: self.getName() = name