Exercise

OCL Expressions

Prof. Dr. Bernhard Rumpe
Lehrstuhl für Software Engineering
RWTH Aachen

http://www.se-rwth.de/

Exercise 3.1 – Solutions 1-2

1. The age of a customer is at least 18 years (class `customer`).

```
context Customer inv Age:
  age >= 18
```

2. All vehicles which are returned from the administrative method `getAvailableVehicles()` have the state "available"

```
context Administration a inv:
  forall v in a.getAvailableVehicles():
    v.state == VehicleState.available
```

Exercise 3.1 – Solutions 3-4

3. The minimum age of a new customer is 18 years (method `newCustomer(…)` in class `Administration`).

```
context Customer Administration.newCustomer(String
                  name, Address address, Date birthday):
  pre:  (currentDate-birthday).year >= 18
  post: result.name.equals(name) &&
        result.address.equals(address) &&
        result.birthday.equals(birthday) &&
        result.age >= 18
```

4. The number of available Vehicles is equal to or smaller than the number of all vehicles.

```
context Administration inv:
     getAvailableVehicles().size <= getVehicles().size
```

Exercise 3.1 – Solutions 5-6

5. In the administration, every customer is unambiguously referenced by the customer ID.

```
context Administration a inv:
  forall c in a.customers:
    a.customer[c.customerID] == c
```

6. A vehicle must be part of a location if it should be removed. After removing, the size of the list of vehicles in a location is reduced by one.

```
context Location.removeVehicle(Vehicle v):
  pre:  vehicles.contains(v)
  post: ! vehicles.contains(v) &&
        (vehicles@pre.size-1 == vehicles.size)
```

Exercise 3.1 – Solution 7

7. After adding a vehicle to a location, the vehicle resides in the new vehicle list in the new location and not anymore in the vehicle list of the old location.

```
context Location.addVehicle(Vehicle v):
  let Location oldLocation = v.location;
  pre:  !vehicles.contains(v)
  post: vehicles.contains(v) &&
        !oldLocation.getVehicles().contains(v)
```

Exercise 3.2 – Code Generation from OCL Expressions

- Generate Java code for the expressions 1 and 7 from task 3.1.

1. The age of a customer is at least 18 years (class `customer`).

```
context Customer inv Age:
  age >= 18
```

JAVA

```
public static boolean invariantAge() {
  Set<Customer> allCustomers =
    Customer.getAllInstances();

  for (Customer c:allCustomers) {
    if ( !(c.getAge() >= 18) ) {
      return = false;
    }
  }
  return true;
}
```

Exercise 3.2 – Run Time Optimized Solution

- Generate Java code for the expressions 1 and 7 from task 3.1.

1. The age of a customer is at least 18 years (class `customer`).

```
context Customer inv Age:
  age >= 18
```

JAVA

```
public boolean checkAge() {
  return age >= 18;
}
```

Exercise 3.2 – Expression 7

7.
```
context Location.addVehicle(Vehicle v):
  let Location oldLocation = v.location;
  pre:  !vehicles.contains(v)
  post: vehicles.contains(v) &&
        !oldLocation.getVehicles().contains(v)
```

JAVA

```
public void addVehicle(Vehicle v) {
  Location oldLocation = v.getLocation();

  // check of the pre-condition
  assert(!vehicles.contains(v));

  … // actual method-body

  // check of the post-condition
  assert(vehicles.contains(v)
         && !oldLocation.getVehicles().contains(v));
}
```