HotRod 2.0 - Features Overview

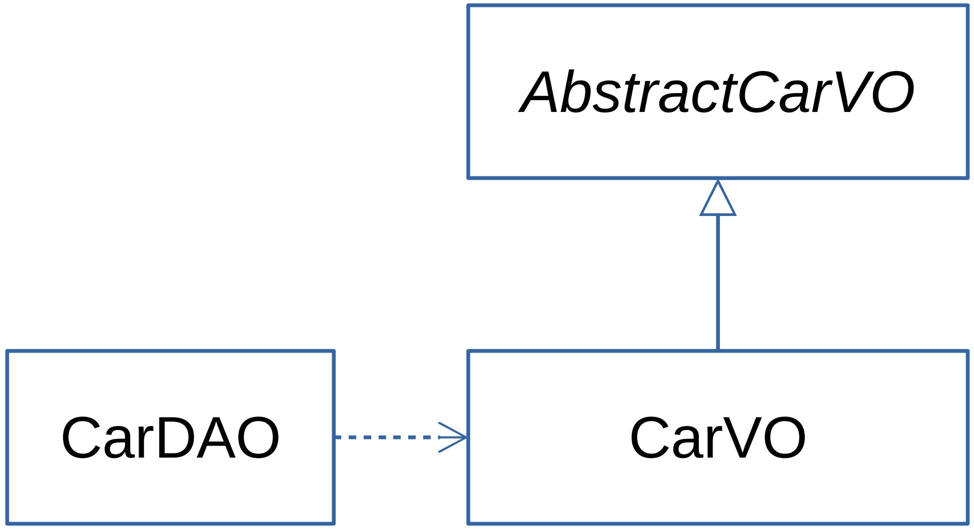
- Simple Standard ORM.
- Enhanced with “Applied SQL” features:
  - Native, Parametric SQL with all real-world tweaks.
  - Full support of MyBatis Dynamic SQL.
  - Automated SQL Column discovery.
  - Structured SQL selects (use existing entity VOs).
  - Automated Compositions (associations and collections).
- All High Performance of MyBatis.
- Automatic & User defined property types + Converters.
- Supports 10 Major Databases.

DAOs & VOs

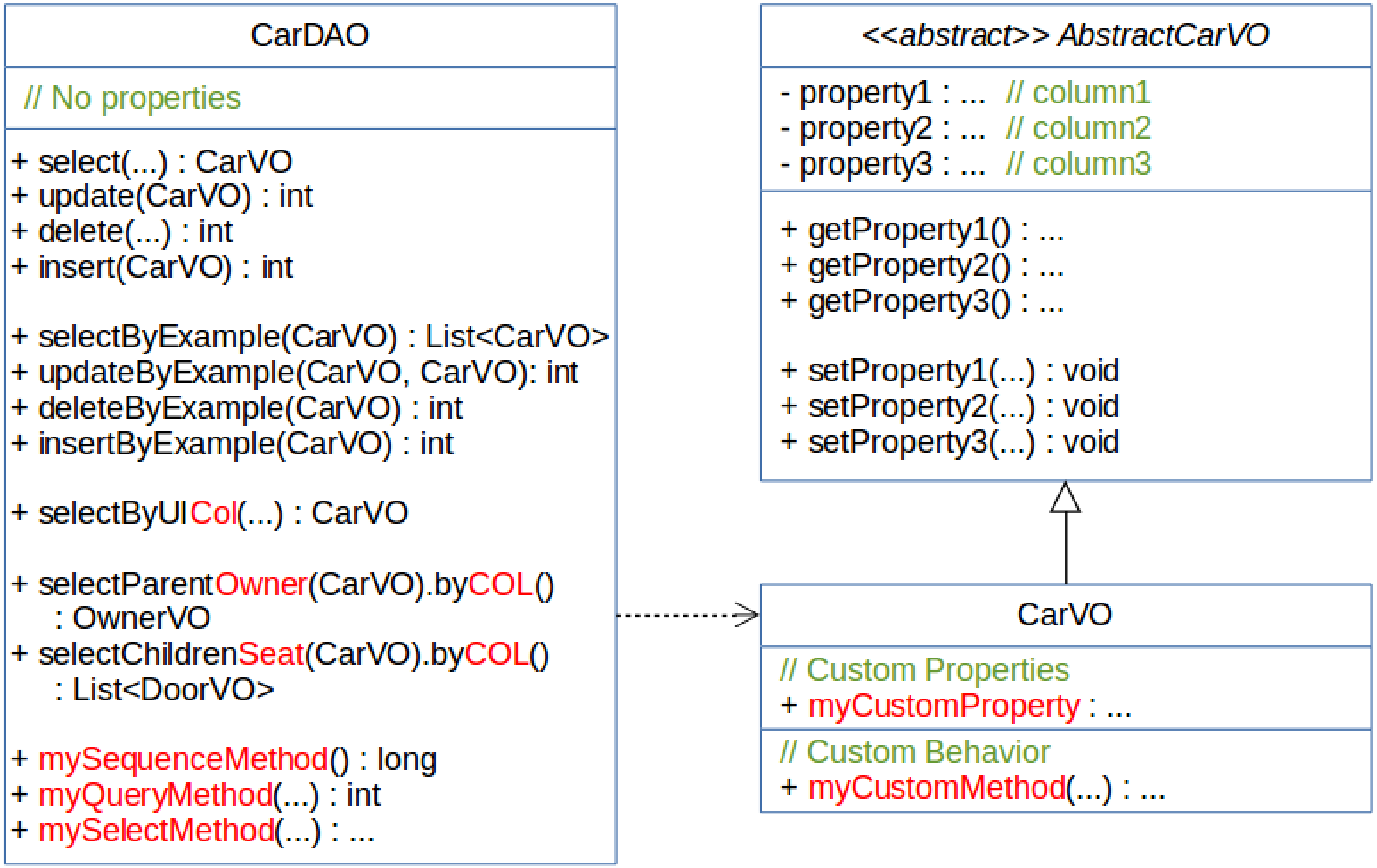
- A database table or view produces three Java classes.

```
<table name="car" />
```

- For example, the table CAR produces:



DAOs & VOs - in more detail


CarDAO
// No properties
+ select(...) : CarVO
+ update(CarVO) : int
+ delete(...) : int
+ insert(CarVO) : int
+ selectByExample(CarVO) : List<CarVO>
+ updateByExample(CarVO, CarVO): int
+ deleteByExample(CarVO) : int
+ insertByExample(CarVO) : int
+ selectByUICol(...) : CarVO
+ selectParentOwner(CarVO).byCOL() : OwnerVO
+ selectChildrenSeat(CarVO).byCOL() : List<DoorVO>
+ mySequenceMethod() : long
+ myQueryMethod(...) : int
+ mySelectMethod(...) : ...
<<abstract>> AbstractCarVO
- property1 : ... // column1
- property2 : ... // column2
- property3 : ... // column3
+ getProperty1() : ...
+ getProperty2() : ...
+ getProperty3() : ...
+ setProperty1(...) : void
+ setProperty2(...) : void
+ setProperty3(...) : void
CarVO
// Custom Properties
+ myCustomProperty : ...
// Custom Behavior
+ myCustomMethod(...) : ...

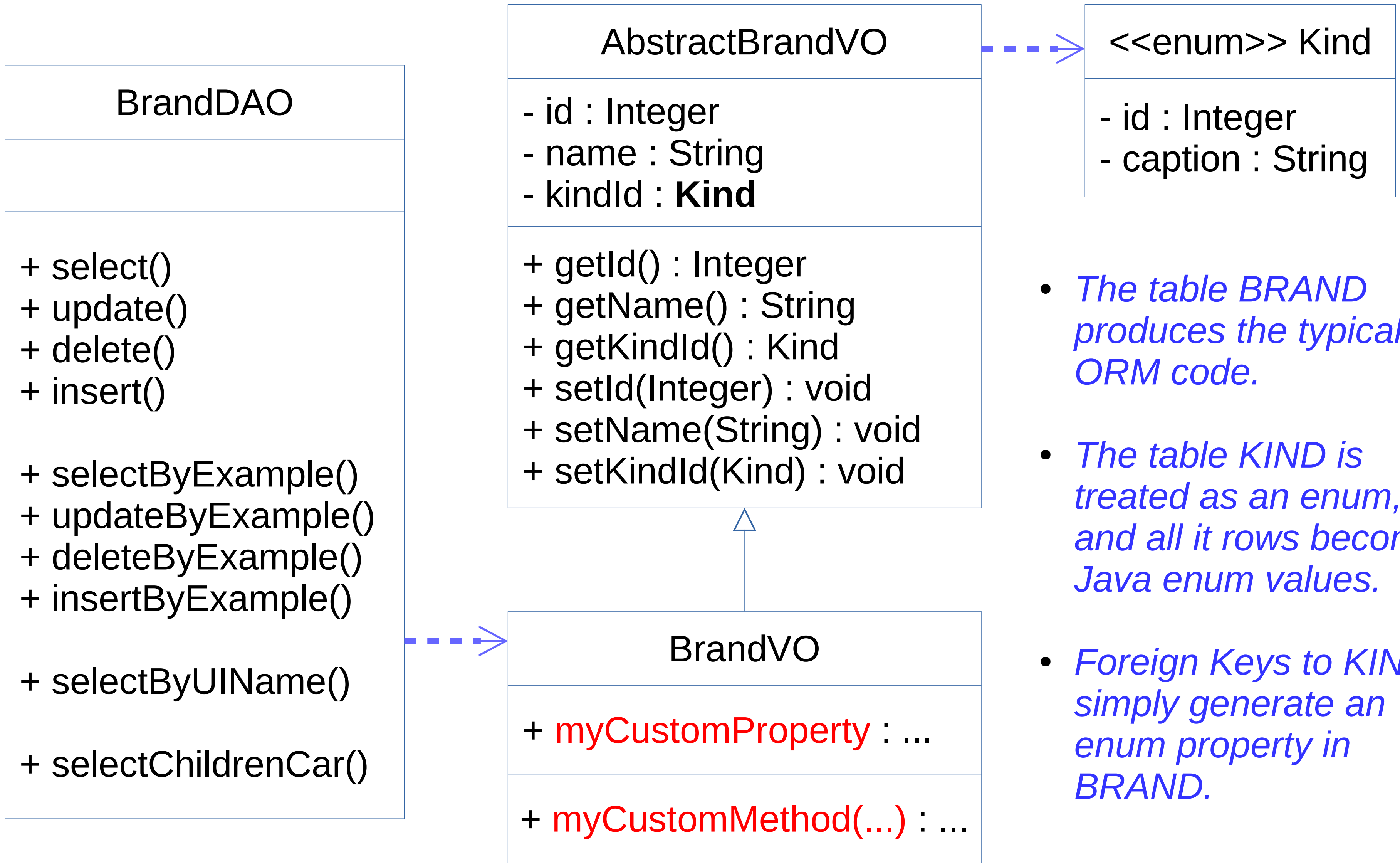
Types of Entities

In HotRod...

```
<table name="brand" />
<table name="car" />
<view name="van" />
<enum name="kind" />
```

```
create table kind (
  id int primary key,
  caption varchar(60)
);

create table brand (
  id int primary key generated always as identity,
  name varchar(40), unique (name),
  kind_id int constraint fk1 references kind
);

create table car (
  id int primary key generated always as identity,
  brand_id int constraint fk2 references brand,
  type varchar(10),
);

create view van as
  select * from car where type = 'VAN';
```

Example BrandVO



- *The table BRAND produces the typical ORM code.*
- *The table KIND is treated as an enum, and all it rows become Java enum values.*
- *Foreign Keys to KIND simply generate an enum property in BRAND.*

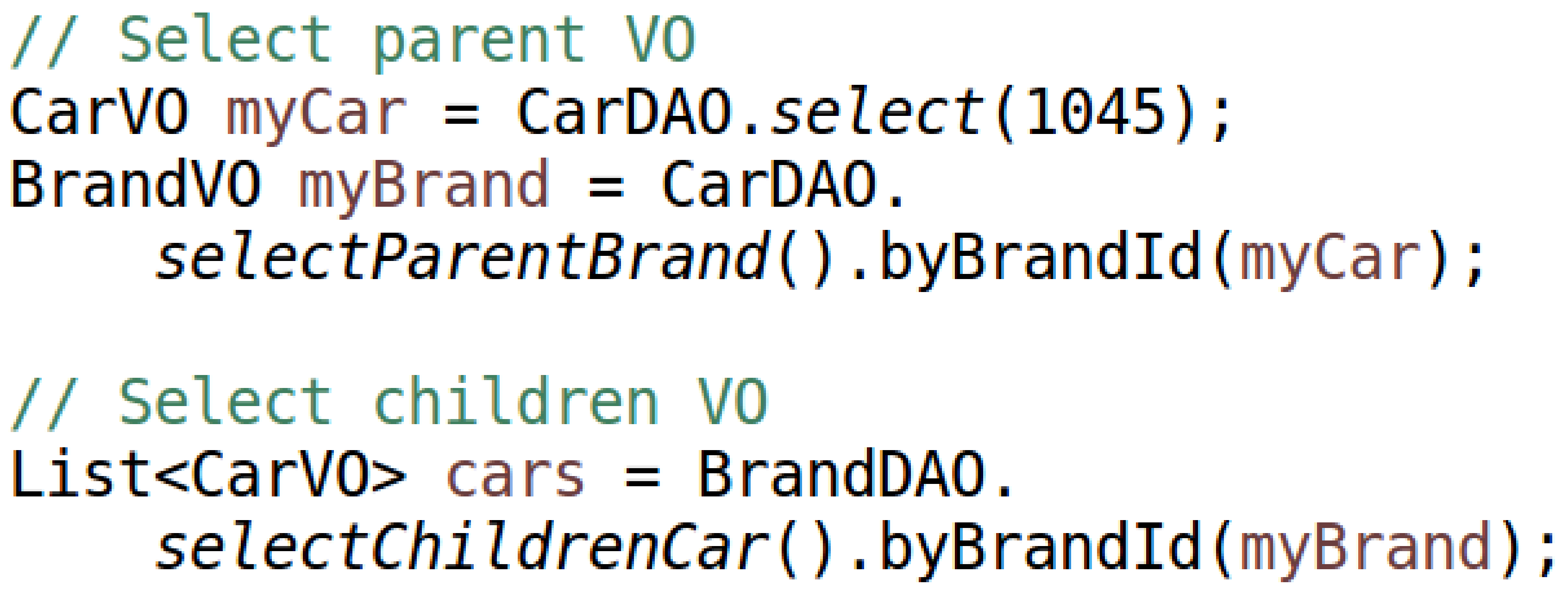
Example - Out of the box CRUD

```
// Select by PK
BrandVO fiat = BrandDAO.select(17);

// Select by Unique Index
BrandVO volvo = BrandDAO.selectByUIName("Volvo");

// Update
fiat.setName("Fiat");
BrandDAO.update(fiat);

// Delete by PK
BrandDAO.delete(volvo);

// Insert
BrandVO b = new BrandVO();
b.setName("Toyota");
BrandDAO.insert(b);
System.out.println("id=" + b.getId());
```

Example - Out of the box By Example

```java
// Select by example - Find vans with no brand ID
CarVO example = new CarVO();
example.setType("VAN");
example.setBrandId(null);
List<CarVO> vans = CarDAO.selectByExample(example);

// Update by example - Set brand ID 17 to vans
//                     with no brand ID
CarVO newValues = new CarVO();
newValues.setBrandId(17);
CarDAO.updateByExample(example, newValues);

// Delete by example - Delete all coupe
//                     with no brand ID
example = new CarVO();
example.setType("COUPE");
example.setBrandId(null);
CarDAO.deleteByExample(example);
```

Example - Out of the box Foreign Keys Navigation



```
// Select parent VO
CarVO myCar = CarDAO.select(1045);
BrandVO myBrand = CarDAO.
    selectParentBrand().byBrandId(myCar);

// Select children VO
List<CarVO> cars = BrandDAO.
    selectChildrenCar().byBrandId(myBrand);
```

Flat Selects (column auto-discovery)

In HotRod...

```
<table name="car">

  <select method="findExtendedCar" vo="ExtendedCarVO">
    <parameter name="brandId" java-type="java.lang.Integer" />
    select
      c.*,
      b.name,
      r.id as repaired_id,
      r.repaired_on, r.card_id
    from car c
    join brand b on b.id = c.brand_id
    left join repair r on r.car_id = c.id
    <complement>
      <where>
        <if test="brandId != null">
          and b.id = #{brandId}
        </if>
      </where>
    </complement>
  </select>

</table>
```

(automatically generated VO)

ExtendedCarVO

- id : Integer   // c.*
- brandId : Integer
- type : String
- name : String   // b.name
- repairedId : Integer
- repairedOn : Date
- cardId : Integer

In Java... it's a single line of code

```
List<ExtendedCarVO> extendedCars =
  CarDAO.findExtendedCar(23);
```

Structured Selects (entity VOs)

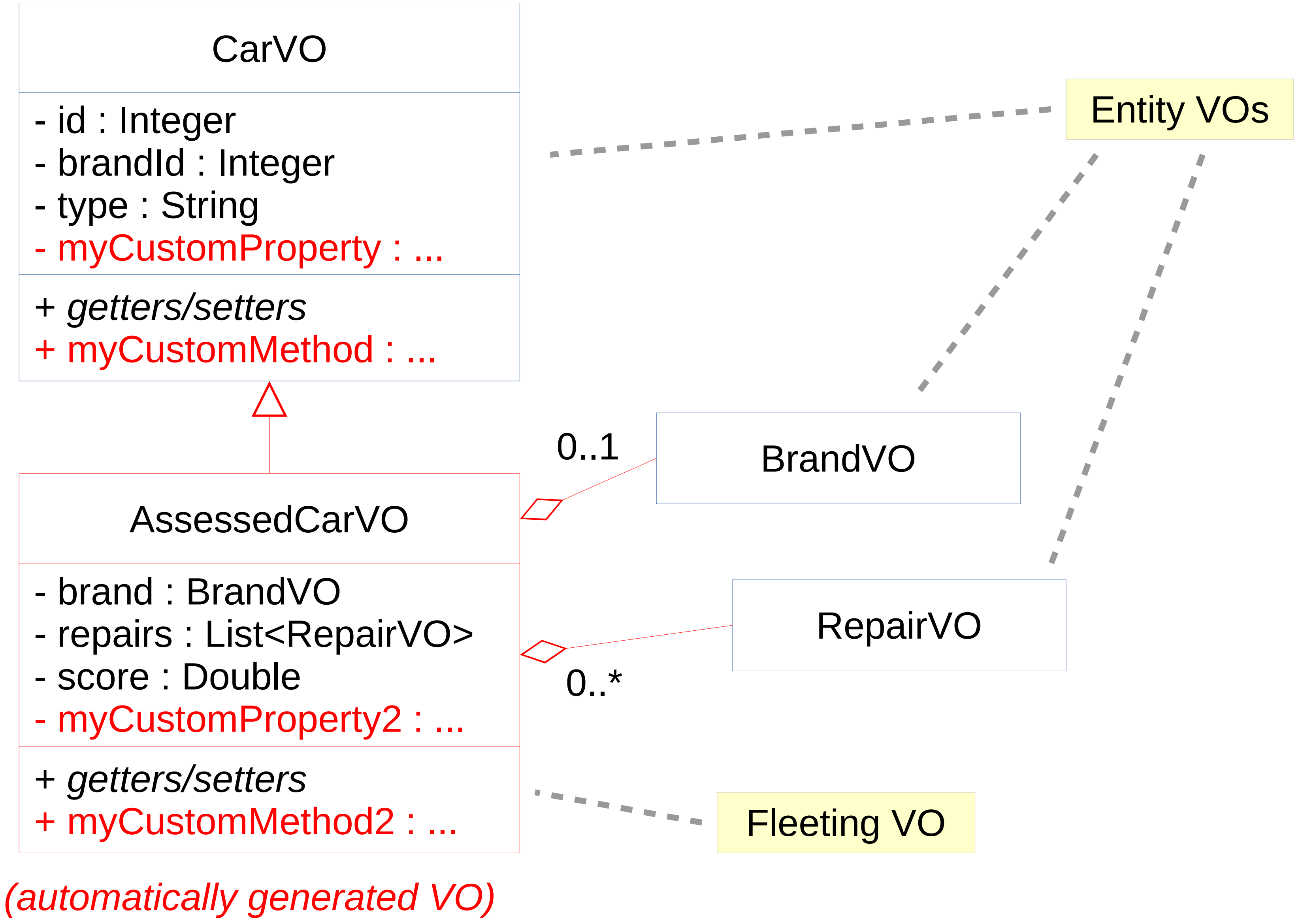
In HotRod...

```
<table name="car">

  <select method="findAssessedCar">
    <parameter name="brandId" java-type="java.lang.Integer" />
    select
    <columns>
      <vo table="car" alias="c" extended-vo="AssessedCarVO">
        <association property="brand" table="brand" alias="b" />
        <collection property="repairs" table="repair" alias="r" />
        <expression property="score"> b.id * c.id + 71 </expression>
      </vo>
    </columns>
    from car c
    join brand b on b.id = c.brand_id
    left join repair r on r.car_id = c.id
    <complement>
      <where>
        <if test="brandId != null">
          and b.id = #{brandId}
        </if>
      </where>
    </complement>
  </select>

</table>
```

In Java... it's a single line of code

```
List<AssessedCarVO> assessedCars =
  CarDAO.findAssessedCar(23);
```

Structured Selects (entity VOs) - cont


CarVO
- id : Integer
- brandId : Integer
- type : String
- myCustomProperty : ...
+ getters/setters
+ myCustomMethod : ...
Entity VOs
0..1
BrandVO
AssessedCarVO
- brand : BrandVO
- repairs : List<RepairVO>
- score : Double
- myCustomProperty2 : ...
+ getters/setters
+ myCustomMethod2 : ...
RepairVO
0..*
Fleeting VO
(automatically generated VO)