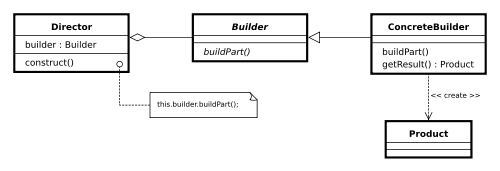
Builder Design Pattern

GOF : **Separate the construction of a complex object from its representation so that the same construction process can create different representations.**

# **Class Diagram**

  
**Java Code**

## **Vehicle.java**

**package** com.ddlab.rnd.type3;

**public** **class** Vehicle {

**private** String vehicleName;

**private** String engineType;

**private** String vehicleType;

**public** String getVehicleName() {

**return** vehicleName;

}

**public** **void** setVehicleName(String vehicleName) {

**this**.vehicleName = vehicleName;

}

**public** String getEngineType() {

**return** engineType;

}

**public** **void** setEngineType(String engineType) {

**this**.engineType = engineType;

}

**public** String getVehicleType() {

**return** vehicleType;

}

**public** **void** setVehicleType(String vehicleType) {

**this**.vehicleType = vehicleType;

}

@Override

**public** String toString() {

**return** "Vehicle [vehicleName=" + vehicleName + ", " + "engineType=" + engineType + ", " + "vehicleType="

+ vehicleType + "]";

}

}

## **IVehicleBuilder.java**

**package** com.ddlab.rnd.type3;

**public** **interface** IVehicleBuilder {

**void** buildEngine(String engineType);

**void** buildBody(String type);

Vehicle getVehicle();

}

## **CarBuilder.java**

**package** com.ddlab.rnd.type3;

**public** **class** CarBuilder **implements** IVehicleBuilder {

**private** Vehicle vehicle = **new** Vehicle();

@Override

**public** **void** buildBody(String type) {

vehicle.setVehicleType(type);

}

@Override

**public** **void** buildEngine(String engineType) {

vehicle.setEngineType(engineType);

vehicle.setVehicleName("Car");

}

@Override

**public** Vehicle getVehicle() {

**return** vehicle;

}

}

## **CarDirector.java**

**package** com.ddlab.rnd.type3;

**public** **class** CarDirector {

**private** IVehicleBuilder builder;

**public** CarDirector(IVehicleBuilder builder) {

**this**.builder = builder;

}

**public** **void** buildVehicle() {

builder.buildBody("Sedan+Luxury");

builder.buildEngine("TurboJet");

}

**public** Vehicle getVehicle() {

**return** builder.getVehicle();

}

}

## **Test.java**

**package** com.ddlab.rnd.type3;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

CarDirector director = **new** CarDirector( **new** CarBuilder() );

director.buildVehicle();

Vehicle vehicle = director.getVehicle();

System.***out***.println(vehicle);

}

}

The UML class diagram is given below.

