Architecturally-evident Java Applications with

jMolecules

Oliver Drotbohm / odrotbohm odrotbohm@vmware.com

github.com/odrotbohm

Search or jump to... / Pull requests Issues Marketplace Explore

Overview Repositories 99 Projects Packages

Oliver Drotbohm

odrotbohm

Frameworks & Architecture Engineering @ VMware, OpenSource enthusiast, all things Spring, Java, data, DDD, REST, software architecture, drums & music. He/him.

Edit profile

2.8k followers · **32** following · ☆ **69**

VMware
Dresden, Germany
info@odrotbohm.de
www.odrotbohm.de
@odrotbohm

Achievements

Pinned

Customize your pins

xmolecules/jmolecules
Libraries to help developers express architectural abstractions in Java code
Java ☆ 374 44

xmolecules/jmolecules-integrations
Technology integration for jMolecules
Java ☆ 10

moduliths
Building modular, monolithic applications using Spring Boot
Java ☆ 499 52

lectures
Lecture scripts and slides I use during the Software Engineering course at TU Dresden
Java ☆ 55 16

spring-restbucks
Implementation of the sample from REST in Practice based on Spring projects
Java ☆ 983 361

spring-playground
A collection of tiny helpers for building Spring applications
Java ☆ 70 8

Single sign-on to see contributions within the **pivotal** organization.

1,733 contributions in the last year

Contribution settings

Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr

Learn how we count contributions.

Less More

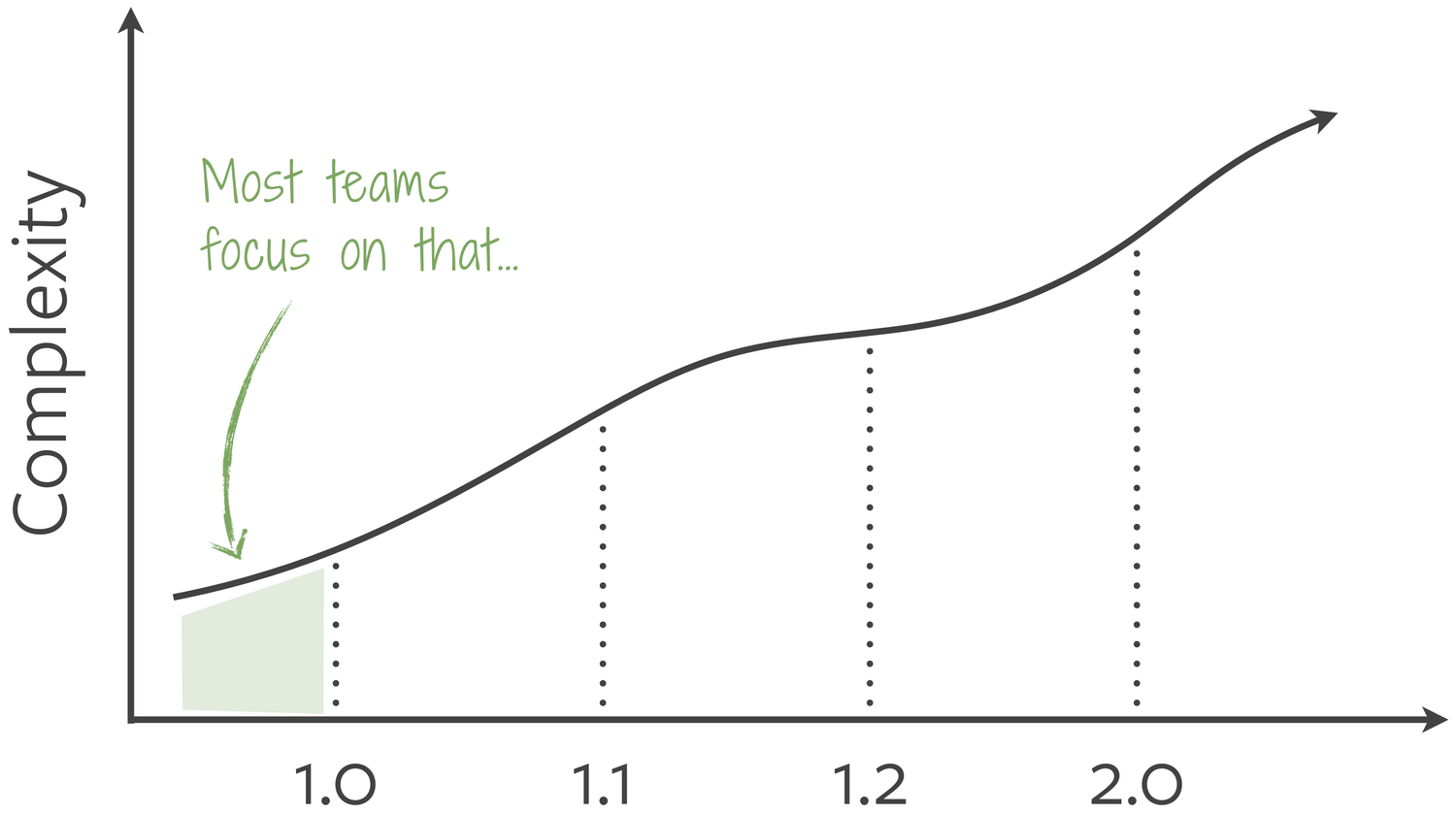
2021 2020 2019 2018 2017 2016 2015

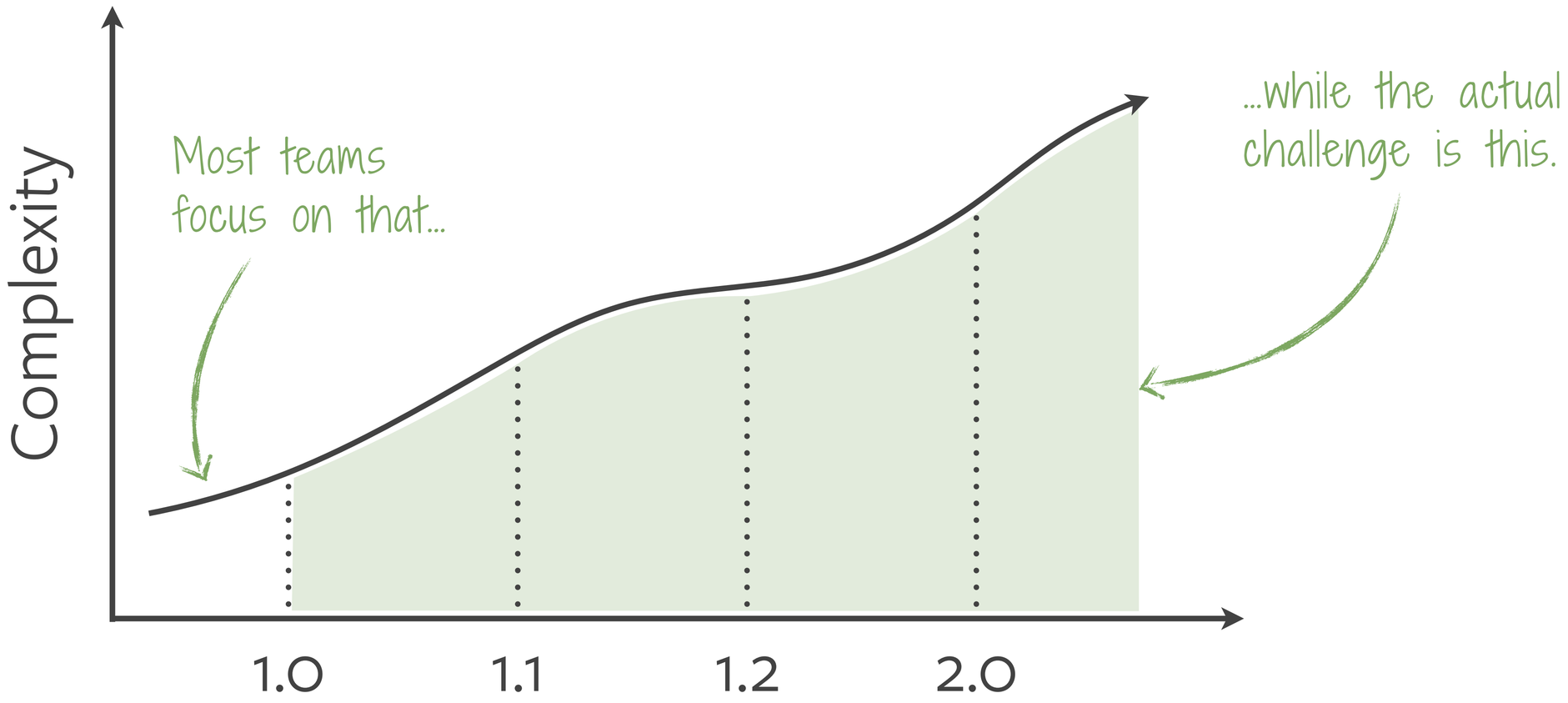
@st-tu-dresden-prak... @spring-projects @xmolecules More

Coming end of 2021...

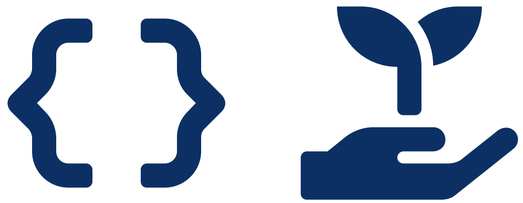
Follow @mawspring on

We want to build evolvable systems.


Complexity
Most teams
focus on that...
1.0
1.1
1.2
2.0


Complexity
Most teams
focus on that...
...while the actual
challenge is this.
1.0
1.1
1.2
2.0

Understandability



Evolvability

Architecturally-Evident Code?

Extensional

Components / Modules

Invoicing,
Shipment

Deployables / Build modules / Packages

Domain language

EmailAddress,
ZipCode

Classes, methods, fields

Intensional

Concepts & Rules

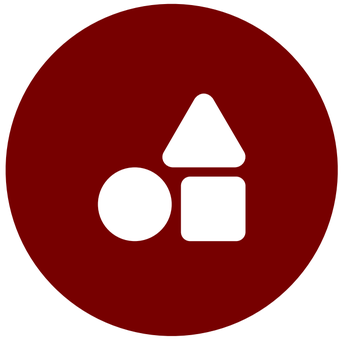
ValueObject,
Entity,
Aggregate

Layers,
Rings

Naming conventions

What else? 🤔

User Code

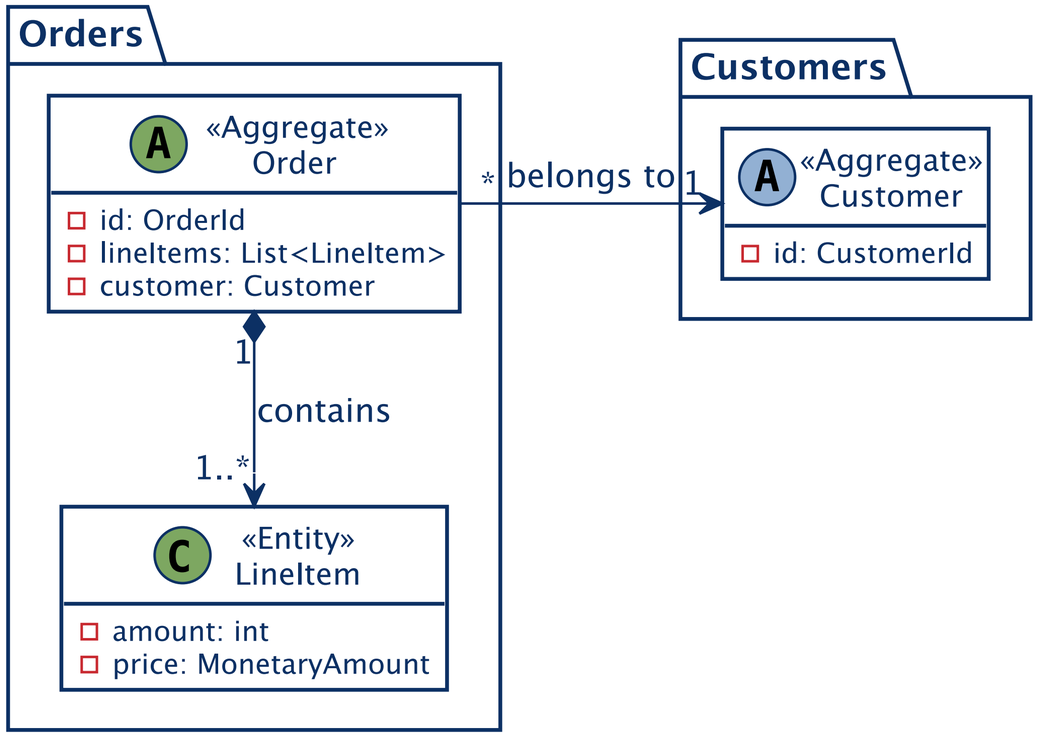


Concepts

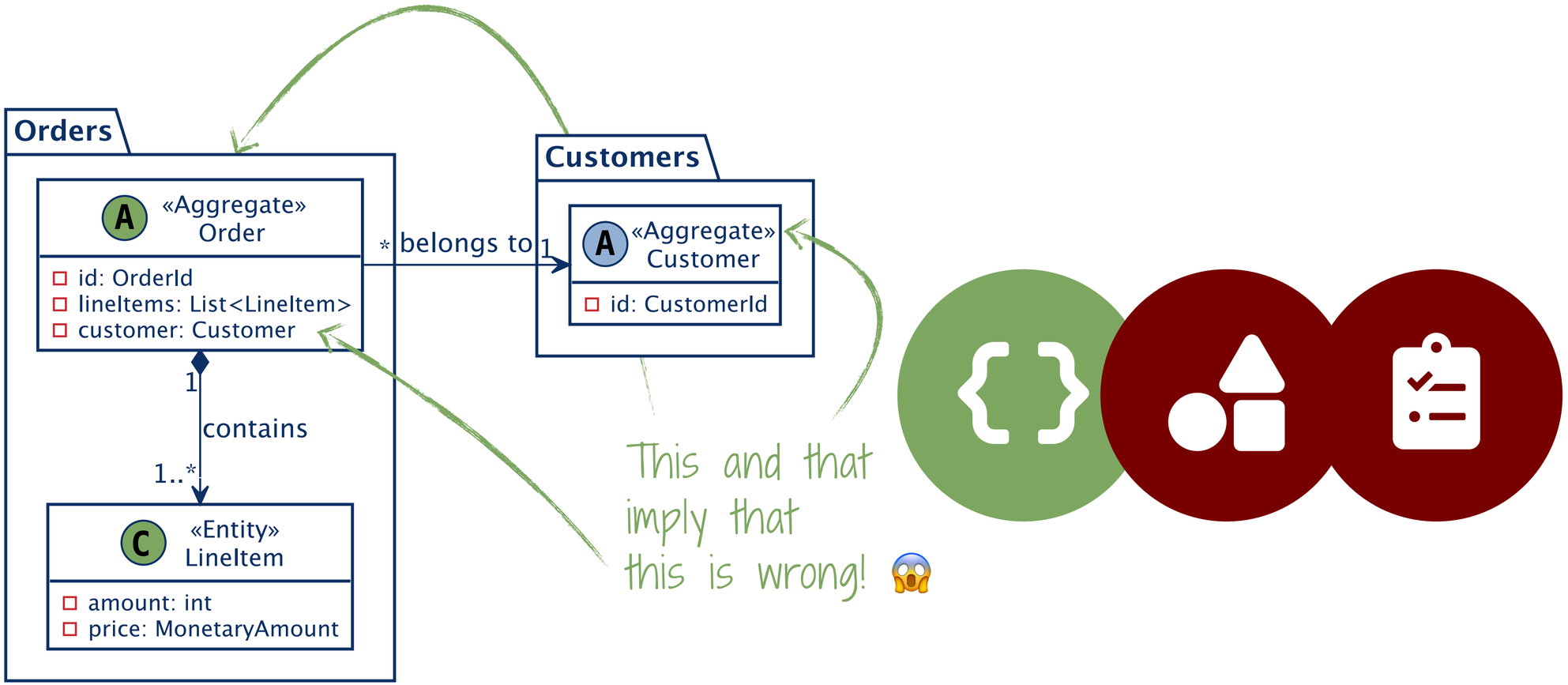
Code

Architecture

Technology

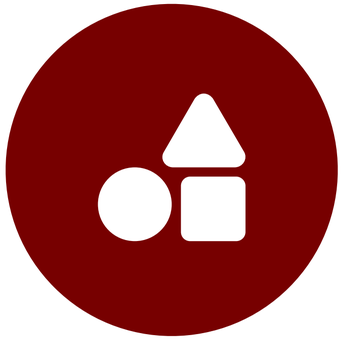


A simple Aggregate arrangement



A simple Aggregate arrangement

User Code



Concepts

Rules

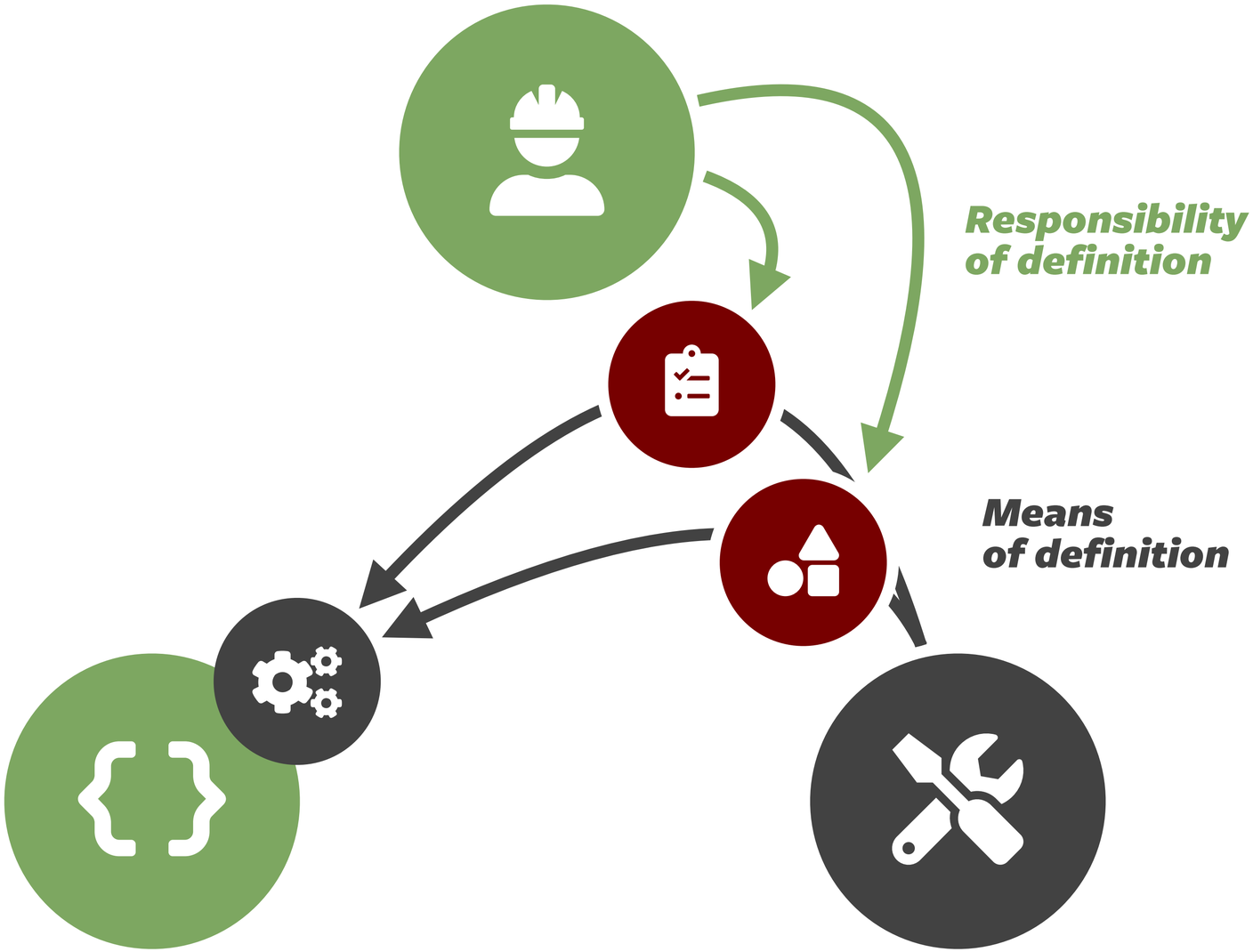
Code

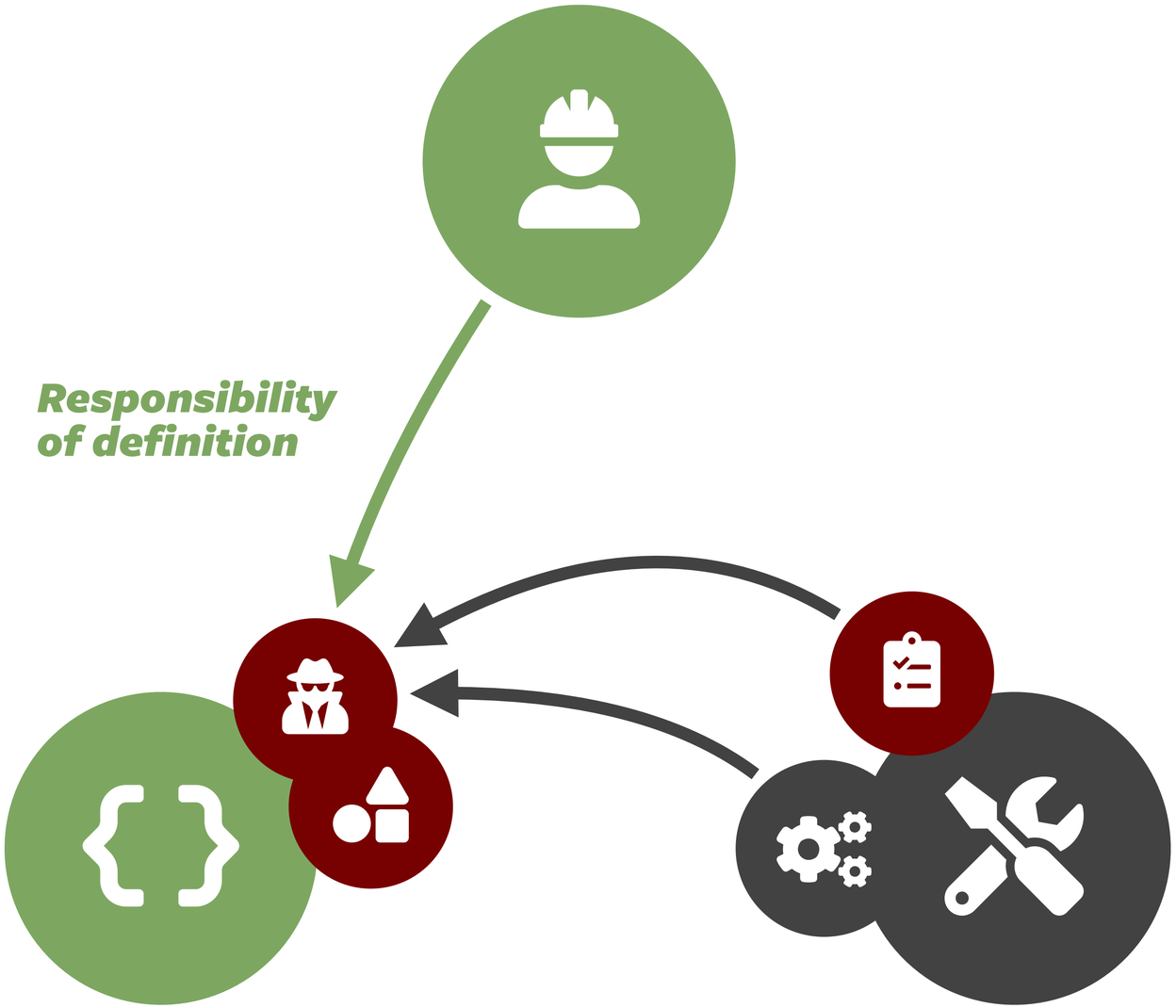
Architecture

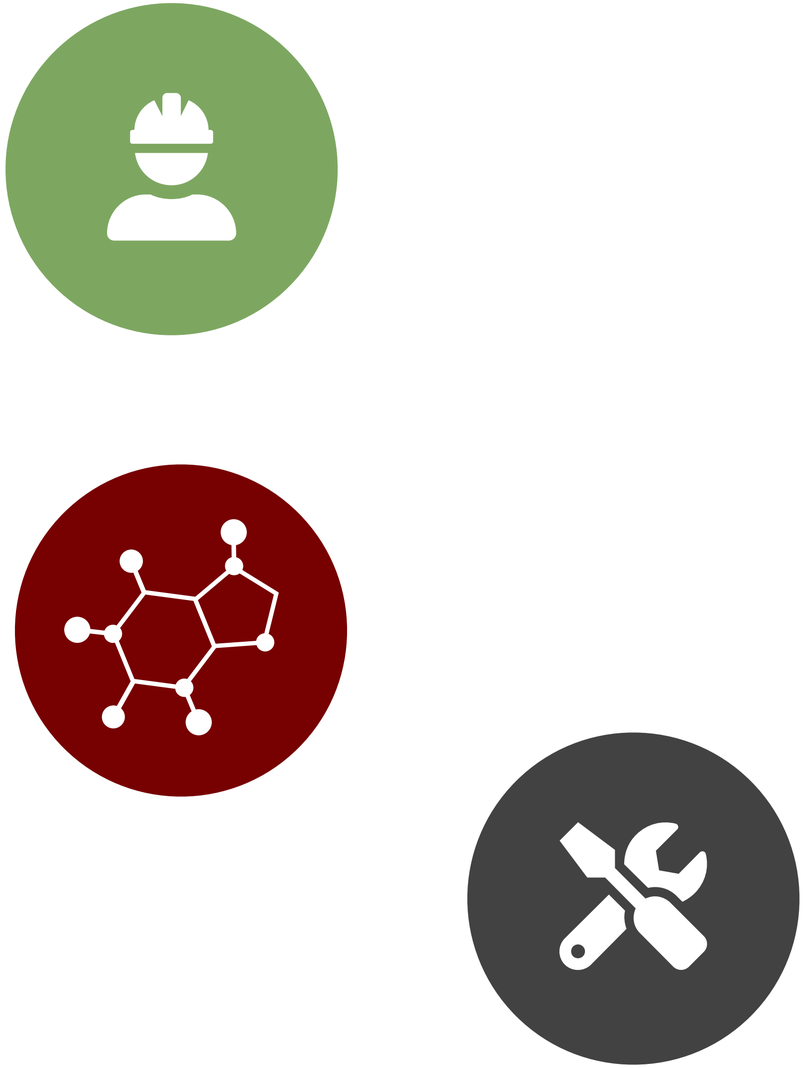
Technology

User Code | Concepts | Rules | Tools | Frameworks

Code | Architecture | Technology


Responsibility
of definition
Means
of definition


Responsibility
of definition



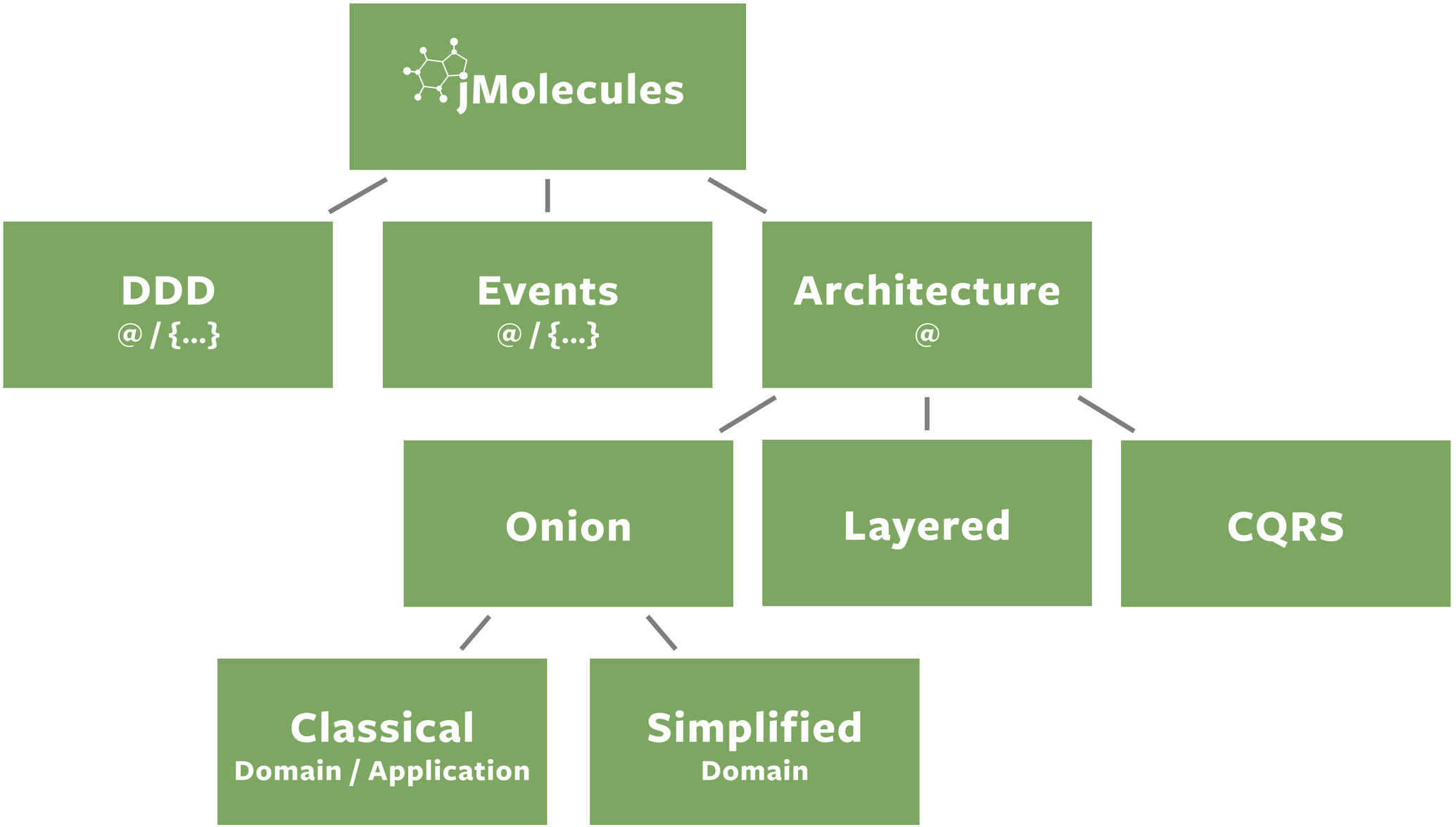
xMolecules

xMolecules

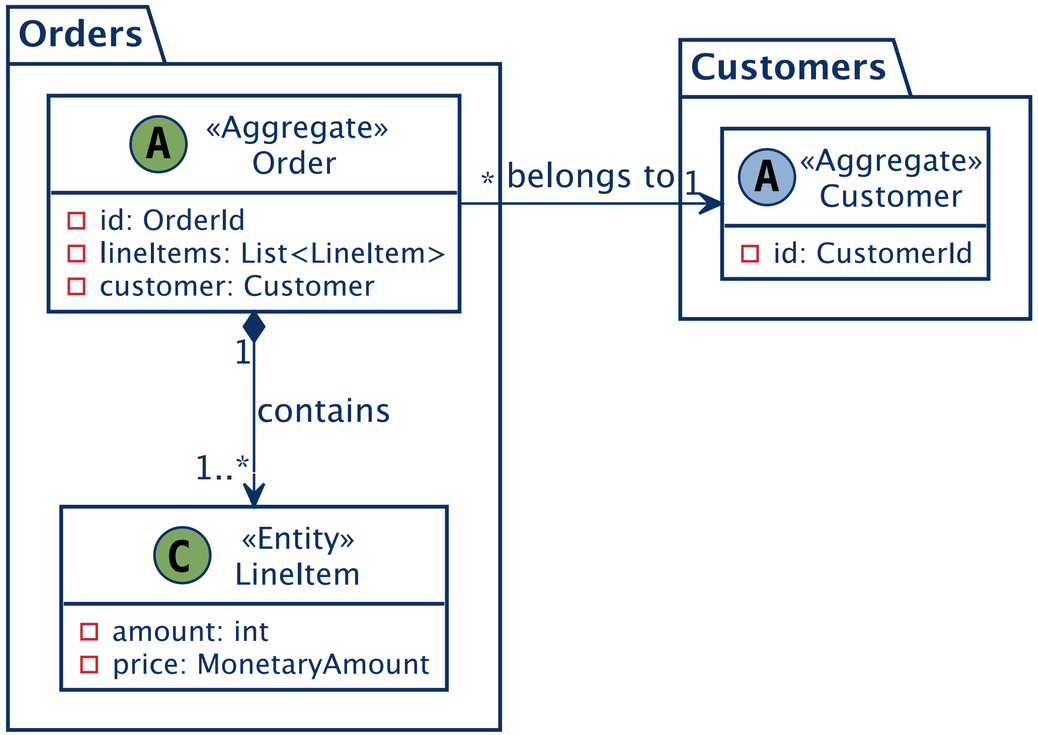
php

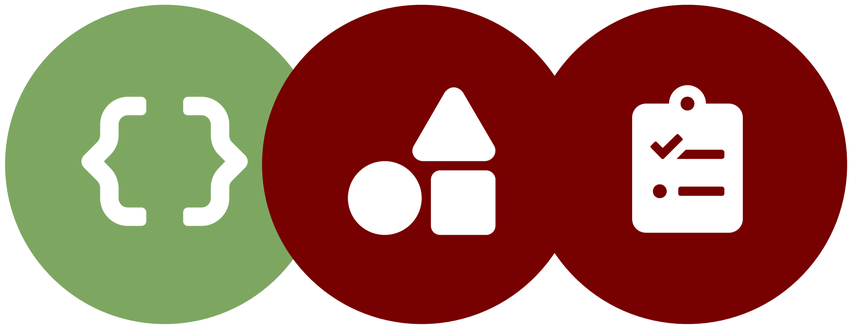
.NET

jMolecules


jMolecules
DDD
@ / {...}
Events
@ / {...}
Architecture
@
Onion
Layered
CQRS
Classical
Domain / Application
Simplified
Domain

An Example...





A simple Aggregate arrangement

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

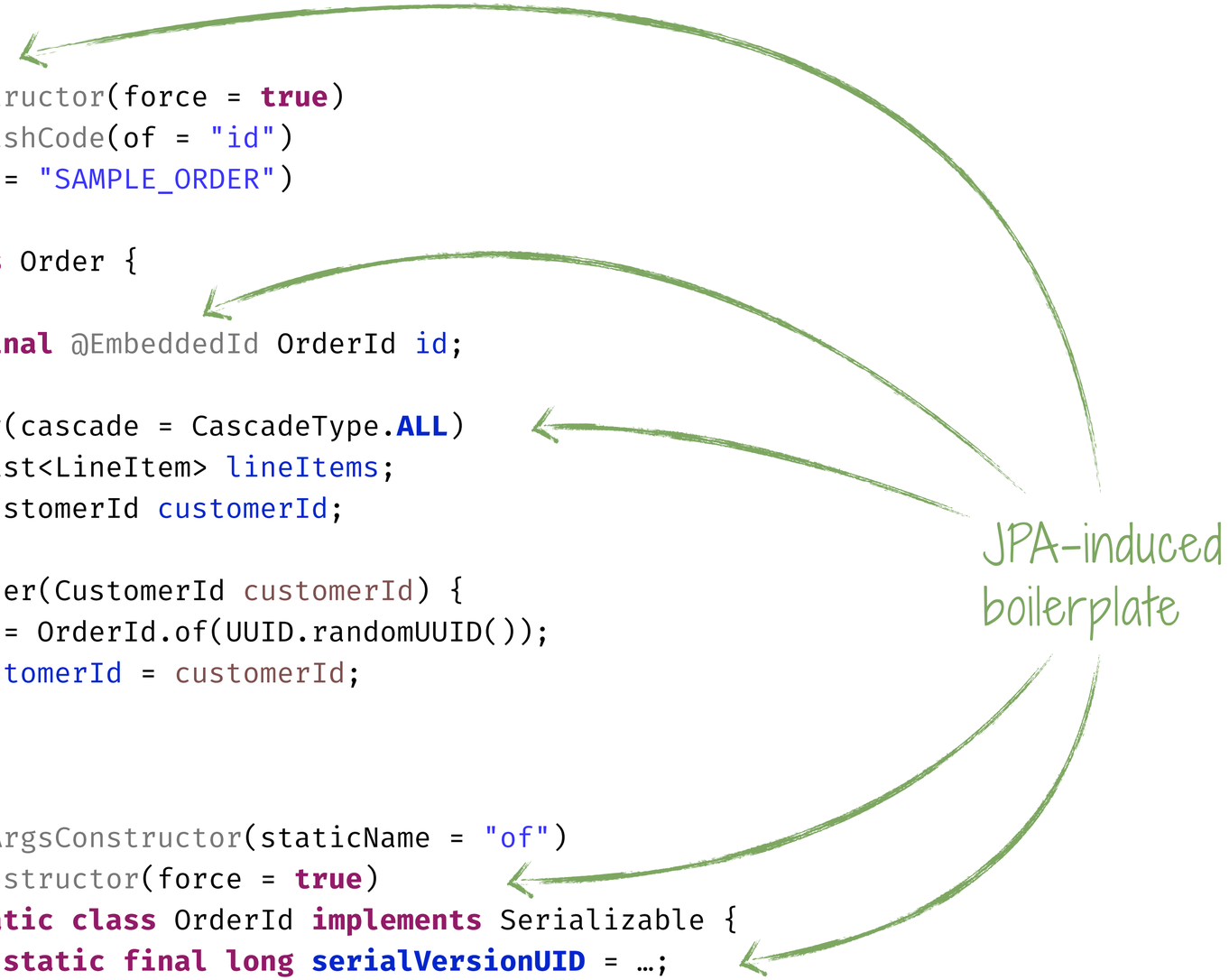
```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```



Persistent model

VS.

Dedicated persistence model

```
@Entity
class Order { … }
```

```
class Order { … }
```

```
@Entity
class JpaOrder { … }
```

Some mapping code, somewhere…


```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

Establishing an Aggregate... in jQAssistant

```
MATCH
  (repo:Java:Type)
    -[:IMPLEMENTS_GENERIC]→ (superType)
    -[:OF_RAW_TYPE]→ (:Java:Type { fqn: "o.s.d.r.Repository"}),
  (superType)
    -[:HAS_ACTUAL_TYPE_ARGUMENT { index: 0 }]→ ()
    -[:OF_RAW_TYPE]→ (aggregateType)
SET
  aggregateType:Aggregate
RETURN
  repo, aggregateType
```

Establishes the concept

Reference to tech stack 😕

```
MATCH
  (aggregate:Aggregate)
    -[:DECLARES]→ (f:Field)
    -[:OF_TYPE]→ (fieldType:Aggregate)
WHERE
  aggregate <> fieldType
RETURN
  aggregate, fieldType
```

Establishes the rule

Establishing an Aggregate... in ArchUnit

```java
@AnalyzeClasses(packagesOf = Application.class)
public class ArchitectureTest {

  @ArchTest
  void verifyAggregates(JavaClasses types) {

    var aggregates = new AggregatesExtractor();
    var aggregateTypes = aggregates.doTransform(types);

    all(aggregates)
      .should(notReferToOtherAggregates(aggregateTypes))
      .check(types);
  }
}
```

Establishes the concept

Establishes the rule

Architectural concepts...

... are only implicitly expressed in the code.

... have to be defined by the developer.

... are defined in a tool-specific way.

Explicit concepts

```
<dependency>
  <groupId>org.jmolecules</groupId>
  <artifactId>jmolecules-ddd</artifactId>
</dependency>
```

Design abstractions

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements o.j.d.t.AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements o.j.d.t.Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

Verifying a jMolecules Aggregate … in jQAssistant

```
<plugin>
  <groupId>com.buschmais.jqassistant</groupId>
  <artifactId>jqassistant-maven-plugin</artifactId>
  <version>${jqassistant.version}</version>
  <executions>
    <execution>
      <id>default-cli</id>
      <goals>
        <goal>scan</goal>
        <goal>analyze</goal>
      </goals>
      <configuration>…</configuration>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.jqassistant.contrib.plugin</groupId>
      <artifactId>jqassistant-jmolecules-plugin</artifactId>
      <version>1.2.0</version>
    </dependency>
  </dependencies>
</plugin>
```

Verifying a jMolecules Aggregate … in ArchUnit

```
@AnalyzeClasses(packagesOf = Application.class)
class ArchitectureTests {

  @ArchTest
  ArchRule ddd = JMoleculesDddRules.all();
}
```

Simply execute the predefined rules

Generated documentation… via Moduliths

```
@AnalyzeClasses(packagesOf = Application.class)
class ArchitectureTests {

  @ArchTest
  ArchRule ddd = JMoleculesDddRules.all();

  @Test
  void documentation() {
    new Documenter(Application.class).writeModuleCanvases();
  }
}
```

Generated documentation… via Moduliths

Base package	`example.jmolecules.presentation.types.customer`
Spring components	*Services* › `e.j.p.t.c.CustomerManagement` listening to `e.j.p.t.c.SampleEvent` *Repositories* › `e.j.p.t.c.Customers`
Aggregate roots	› `e.j.p.t.c.Customer`
Published events	› `e.j.p.t.c.SampleEvent` created by: › `e.j.p.t.c.Customer.<init>(…)`

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

Eliminate boilerplate

```xml
<dependency>
  <groupId>org.jmolecules</groupId>
  <artifactId>jmolecules-ddd</artifactId>
</dependency>
```

Design abstractions

```xml
<plugin>
  <groupId>net.bytebuddy</groupId>
  <artifactId>byte-buddy-maven-plugin</artifactId>
  <version>${bytebuddy.version}</version>
  <executions>
    <execution>
      <goals>
        <goal>transform</goal>
      </goals>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.jmolecules.integrations</groupId>
      <artifactId>jmolecules-bytebuddy</artifactId>
      <version>${jmolecules-integrations.version}</version>
    </dependency>
  </dependencies>
</plugin>
```

Technical integration

incl. technology-specific dependencies

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

Meanwhile in your IDE…

```
[INFO] jMolecules JPA - e.j.p.t.o.Order - Adding default constructor.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Adding @j.p.Entity.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Defaulting e.j.p.t.o.Order.id to @j.p.EmbeddedId() mapping.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Defaulting e.j.p.t.o.Order.lineItems to @j.p.OneToMany(…) mapping.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Implementing o.j.s.d.MutablePersistable<…>.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Implement j.i.Serializable.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Adding default constructor.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Adding @j.p.Embeddable.
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final OrderId id;
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value(staticConstructor = "of")
  public static class OrderId implements Identifier {
    private final UUID orderId;
  }
}
```

This is the aggregate identifier

```java
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final OrderId id;
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value(staticConstructor = "of")
  public static class OrderId implements Identifier {
    private final UUID orderId;
  }
}
```

This is a reference to another aggregate

```java
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final OrderId id;
  private List<LineItem> lineItems;
  private Association<Customer, CustomerId> customer;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customer = Association.forId(customerId);
  }

  @Value(staticConstructor = "of")
  public static class OrderId implements Identifier {
    private final UUID orderId;
  }
}
```

Meanwhile in your IDE…

```
[INFO] jMolecules JPA - e.j.p.t.o.Order - Adding default constructor.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Adding @j.p.Entity.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Defaulting e.j.p.t.o.Order.id to @j.p.EmbeddedId() mapping.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Defaulting e.j.p.t.o.Order.lineItems to @j.p.OneToMany(…) mapping.
[INFO] jMolecules JPA - e.j.p.t.o.Order - Implementing o.j.s.d.MutablePersistable<…>.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Implement j.i.Serializable.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Adding default constructor.
[INFO] jMolecules JPA - e.j.p.t.o.Order.OrderId - Adding @j.p.Embeddable.
[INFO] jMolecules Spring JPA - e.j.p.t.o.Order.customer - Adding @j.p.Convert(converter=…).
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(Customer customer) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customer.getId();
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final OrderId id;
  private List<LineItem> lineItems;
  private Association<Customer, CustomerId> customer;

  public Order(Customer customer) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customer = Association.forAggregate(customer);
  }

  @Value(staticConstructor = "of")
  public static class OrderId implements Identifier {
    private final UUID orderId;
  }
}
```

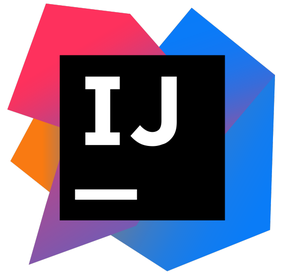
Architectural concepts...

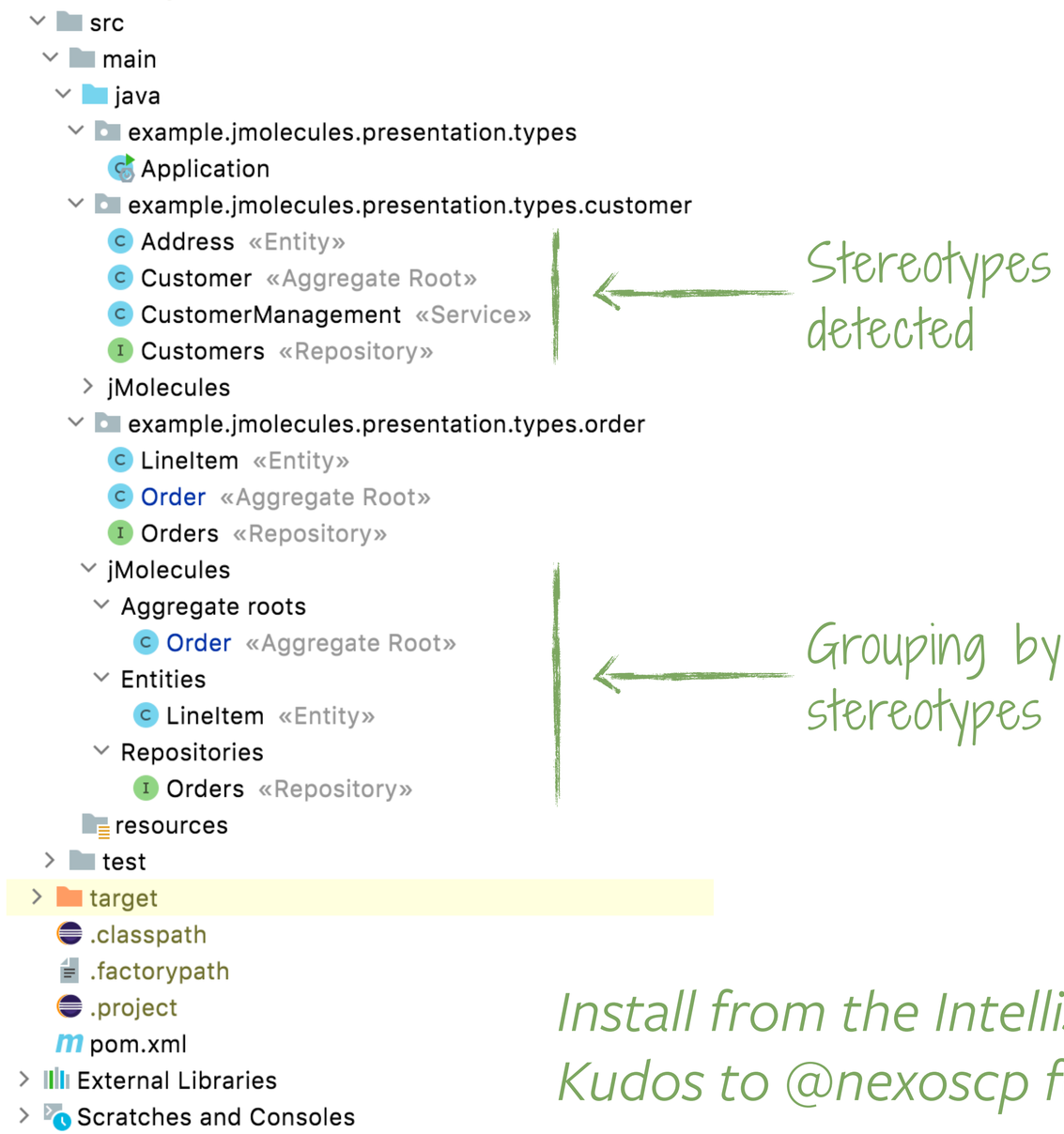
... are explicitly expressed in the code.

... are predefined based on established terms.

... are defined by jMolecules (concepts)
and tool integration (rules).

IDE support



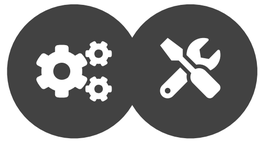
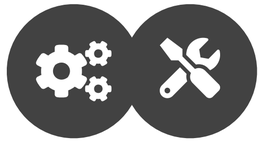


Stereotypes detected

Grouping by stereotypes

Install from the IntelliJ IDEA plugin portal.
Kudos to @nexoscp for the contributions!

Summary

	Traditional	jMolecules
Concepts	Implicit	Explicit
Who?		
How?		

Links

- **xMolecules**
  https://xmolecules.org
- **jMolecules**
  https://jmolecules.org
- **jMolecules Examples**
  https://github.com/xmolecules/jmolecules-examples
- **Gitter – Join the community!**
  https://gitter.im/xmolecules/xmolecules

Resources

- **Software Architecture for Developers**
  Simon Brown – Books
- **Just Enough Software Architecture**
  George Fairbanks – Book
- **Architecture, Design, Implementation**
  Ammon H. Eden, Rick Kazman – Paper
- **Sustainable Software Architecture**
  Carola Lilienthal – Book

Shoutouts 🤦

- Peter Gafert – ArchUnit
- Rafael Winterhalter – ByteBuddy
- Bernd Dutkowski – IDEA plugin
- You?? 🙃 – ideas, discussions

Thank you!

Oliver Drotbohm odrotbohm odrotbohm@vmware.com

Appendix

Technology integration

Spring	• Component definitions (controllers, services, repositories) • Event listeners • Converters (primitive ⇄ identifier ⇄ association) • Spring Boot auto-configuration for converters
Spring Data	• Aggregate definitions (via `Persistable`) implementation • JPA `AttributeConverter` • Converters (primitive ⇄ identifier ⇄ aggregate)
Jackson	• Single-property value objects • (De)Serializers for primitive ⇄ association
Moduliths	• Module canvas detecting stereotypes, aggregates, consumed and published events
jQAssistant	• Verification of DDD Aggregate structure • Verification of layering
ArchUnit	• Verification of DDD Aggregate structure • Verification of layering

	Legacy	jMolecules + Code verification		jMolecules + Code generation
Concepts	implicit	in metadata	in the type system	in metadata in the type system
Means of expression	Application of technology Naming conventions Technology configuration	jMolecules annotations	jMolecules types	jMolecules types and annotations
Technology projection	manual	manual		generated code
Purity of code	⌄⌄ Scattered with technology	⌄ Scattered with technology Model vocabulary		⌃⌃ pure
Potential of deviation	⌃ high	⌃ high	⌄ reduced	⌄⌄ strongly reduced
Means of verification	⌄⌄ External tooling User configuration	⌄ External tooling	⌃ External tooling	⌃⌃ External tooling (optional)
Means of integration	manual	jMolecules integrations (Spring, Jackson, documentation…)		