THE DEVELOPER'S CONFERENCE

Organizando as regras de negócios com Specification em aplicações Java

Murillo da Silveira Grübler
Mestre em computação aplicada e engenheiro de software

Agenda

THE DEVELOPER'S CONFERENCE

- Introdução;
- Specification;
- Hard Coded Specification;
- Parameterized Specification;
- Composite Specification;
- Specification com Java 8;
- Dúvidas.

Primeiramente, vamos imaginar...

A funcionalidade precisa:

THE DEVELOPER'S CONFERENCE

- Verificar se a remessa existe;
- Validar se o pacote tem pelo menos um item;
- Validar se o peso do pacote não ultrapassa o máximo estipulado;
- Atualizar o estado para confirmar.

THE DEVELOPER'S CONFERENCE

```
@Override
public void confirmation(Long shipmentId) {
    ShipmentEntity shipmentEntity = shipmentRepository
            .findById(shipmentId)
            .orElseThrow(() -> new NotFoundException(SHIPMENT_NOT_FOUND));

    if (shipmentEntity.getQuantity().equals(NumberUtils.LONG_ZERO)){
        throw new BusinessException(PACKAGE_MUST_HAVE_AT_LEAST_ONE_ITEM);
    }

    if (shipmentEntity.getWeight() < MINIMUM_WEIGHT){
        throw new BusinessException(INVALID_MINIMUM_WEIGHT);
    }

    shipmentEntity.setConfirmed(true);
    shipmentRepository.save(shipmentEntity);
}
```

A funcionalidade precisa:

- Verificar se a remessa existe;
- Validar se o pacote tem mais pelo menos um item;
- Validar se o peso do pacote não ultrapassa o máximo estipulado;
- **Validar se a requisição está no prazo;**
- Atualizar o estado para confirmar.

```java
@Override
public void confirmation(Long shipmentId) {
    ShipmentEntity shipmentEntity = shipmentRepository
            .findById(shipmentId)
            .orElseThrow(() -> new NotFoundException(SHIPMENT_NOT_FOUND));

    if (shipmentEntity.getQuantity().equals(NumberUtils.LONG_ZERO)){
        throw new BusinessException(PACKAGE_MUST_HAVE_AT_LEAST_ONE_ITEM);
    }

    if (shipmentEntity.getWeight() < MINIMUM_WEIGHT){
        throw new BusinessException(INVALID_MINIMUM_WEIGHT);
    }

    if (ChronoUnit.DAYS.between(shipmentEntity.getRequest(), LocalDateTime.now()) > TWO_WEEKS){
        throw new BusinessException(LATE_REQUEST);
    }

    shipmentEntity.setConfirmed(true);
    shipmentRepository.save(shipmentEntity);
}
```

A funcionalidade precisa:

- Verificar se a remessa existe;
- Validar se o pacote tem mais pelo menos um item;
- Validar se o peso do pacote não ultrapassa o máximo estipulado;
- Validar se a requisição está no prazo;
- **Validar se a requisição possui todos os documentos necessários;**
- Atualizar o estado para confirmar.

THE DEVELOPER'S CONFERENCE

```java
@Override
public void confirmation(Long shipmentId) {
    ShipmentEntity shipmentEntity = shipmentRepository
            .findById(shipmentId)
            .orElseThrow(() -> new NotFoundException(SHIPMENT_NOT_FOUND));

    if (shipmentEntity.getQuantity().equals(NumberUtils.LONG_ZERO)){
        throw new BusinessException(PACKAGE_MUST_HAVE_AT_LEAST_ONE_ITEM);
    }

    if (shipmentEntity.getWeight() < MINIMUM_WEIGHT){
        throw new BusinessException(INVALID_MINIMUM_WEIGHT);
    }

    if (ChronoUnit.DAYS.between(shipmentEntity.getRequest(), LocalDateTime.now()) > TWO_WEEKS){
        throw new BusinessException(LATE_REQUEST);
    }

    if (shipmentEntity.getDocuments().isEmpty()){
        throw new BusinessException(SHIPMENT_DOES_NOT_HAVE_DOCUMENTS);
    }

    shipmentEntity.setConfirmed(true);
    shipmentRepository.save(shipmentEntity);
}
```

Então...

THE DEVELOPER'S CONFERENCE

- As regras mudam constantemente durante o desenvolvimento.
- É necessário adotar estratégias para diminuir o impacto de futuras alterações no sistema.

Specification

THE DEVELOPER'S CONFERENCE

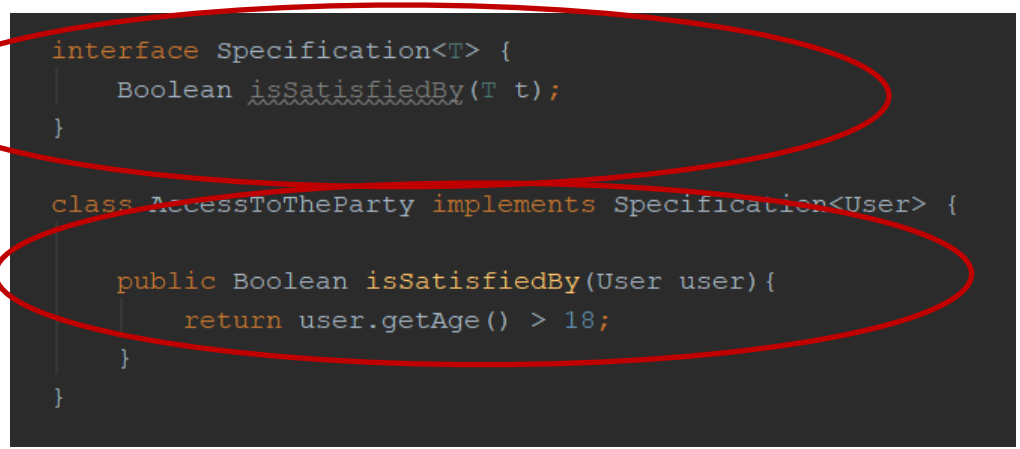
- Martin Fowler e Eric Evans;
- Isolar a regra de negócio;
- Propostas:
  - Validar um objeto;
  - Selecionar um objeto de uma coleção;
  - Especificar a criação de um novo objeto.

Specification

THE DEVELOPER'S CONFERENCE

- Há três diferentes estratégias de implementação:
  - **Hard Coded Specification;**

Hard Coded Specification



```
interface Specification<T> {
    Boolean isSatisfiedBy(T t);
}

class AccessToTheParty implements Specification<User> {

    public Boolean isSatisfiedBy(User user){
        return user.getAge() > 18;
    }
}
```

```
Boolean result = new AccessToTheParty().isSatisfiedBy(new User( age: 20));
```

Hard Coded Specification

THE DEVELOPER'S CONFERENCE

Vantagens:

- Fácil desenvolvimento;
- Expressivo.

Desvantagens:

- Inflexível.

Specification

THE DEVELOPER'S CONFERENCE

- Há três diferentes estratégias de implementação:
  - Hard Coded Specification;
  - **Parameterized Specification;**

Parameterized Specification

THE DEVELOPER'S CONFERENCE

```
interface Specification<T> {
    Boolean isSatisfiedBy(T t);
}

class AccessToTheParty implements Specification<User> {

    private Integer ageOfMajority;

    public AccessToTheParty(Integer ageOfMajority){
        this.ageOfMajority = ageOfMajority;
    }

    public Boolean isSatisfiedBy(User user){
        return user.getAge() > ageOfMajority;
    }
}
```

```
User user = new User( age: 20);
Integer over21 = 21;
Boolean result = new AccessToTheParty(over21).isSatisfiedBy(user);
```

Parameterized Specification

THE DEVELOPER'S CONFERENCE

Vantagens:

- Fácil desenvolvimento;
- Expressivo;
- Há um pequeno ganho de flexibilidade.

Desvantagens:

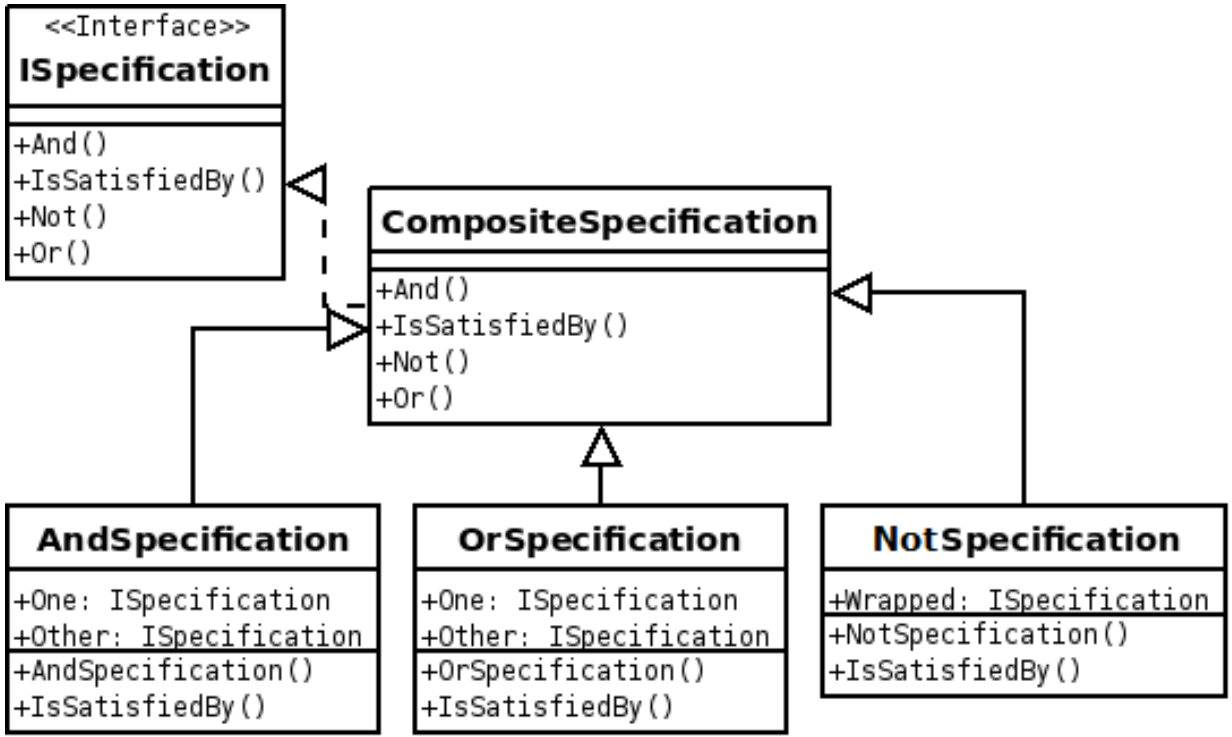
- Inflexível.

Specification

THE DEVELOPER'S CONFERENCE

- Há três diferentes estratégias de implementação:
  - Hard Coded Specification;
  - Parameterized Specification;
  - **Composite Specification.**

Composite Specification



Composite Specification

THE DEVELOPER'S CONFERENCE

```
public interface Specification<T> {
    Boolean isSatisfiedBy(T t);
    Specification<T> and(Specification<T> other);
    Specification<T> or(Specification<T> other);
}
```

Composite Specification

THE DEVELOPER'S CONFERENCE

```
public abstract class CompositeSpecification<T> implements Specification<T> {

    @Override
    public Specification<T> and(Specification<T> other) {
        return new AndSpecification<>( leftRule: this, other);
    }

    @Override
    public Specification<T> or(Specification<T> other) {
        return new OrSpecification<>( leftRule: this, other);
    }

}
```

Composite Specification

THE DEVELOPER'S CONFERENCE

```
public class AndSpecification<T> extends CompositeSpecification<T> {
    private Specification<T> leftRule;
    private Specification<T> rightRule;

    public AndSpecification(Specification<T> leftRule, Specification<T> rightRule) {
        this.leftRule = leftRule;
        this.rightRule = rightRule;
    }

    @Override
    public Boolean isSatisfiedBy(T t) {
        return leftRule.isSatisfiedBy(t) && rightRule.isSatisfiedBy(t);
    }
}
```

Composite Specification

THE DEVELOPER'S CONFERENCE

```
public class OrSpecification<T> extends CompositeSpecification<T> {
    private Specification<T> leftRule;
    private Specification<T> rightRule;

    public OrSpecification(Specification<T> leftRule, Specification<T> rightRule) {
        this.leftRule = leftRule;
        this.rightRule = rightRule;
    }

    @Override
    public Boolean isSatisfiedBy(T t) {
        return leftRule.isSatisfiedBy(t) || rightRule.isSatisfiedBy(t);
    }

}
```

THE DEVELOPER'S CONFERENCE

```java
public class IsOnTime extends CompositeSpecification<Shipment>{

    private final Integer TWO_WEEKS = 14;

    @Override
    public Boolean isSatisfiedBy(Shipment shipment) {
        return ChronoUnit.DAYS.between(shipment.getDateTime(), LocalDateTime.now()) <= TWO_WEEKS;
    }
}
```

Composite Specification

THE DEVELOPER'S CONFERENCE

```
Boolean result = new MinimumWeightAllowed()
        .and(new IsInProgress())
        .and(new IsOnTime())
        .isSatisfiedBy(shipment);
```

Composite Specification

Composite Specification

THE DEVELOPER'S CONFERENCE

Vantagens:

- Suporta operações lógicas como AND, OR ou NOT;
- Flexível.

Desvantagens:

- Possui uma estrutura inicial mais complexa.
- Não indicado para validações de coleções de objetos.

Voltando ao nosso sistema...

THE DEVELOPER'S CONFERENCE

```java
@Override
public void confirmation(Long shipmentId) {
    ShipmentEntity shipmentEntity = shipmentRepository
            .findById(shipmentId)
            .orElseThrow(() -> new NotFoundException(SHIPMENT_NOT_FOUND));

    if (shipmentEntity.getQuantity().equals(NumberUtils.LONG_ZERO)){
        throw new BusinessException(PACKAGE_MUST_HAVE_AT_LEAST_ONE_ITEM);
    }

    if (shipmentEntity.getWeight() < MINIMUM_WEIGHT){
        throw new BusinessException(INVALID_MINIMUM_WEIGHT);
    }

    if (ChronoUnit.DAYS.between(shipmentEntity.getRequest(), LocalDateTime.now()) > TWO_WEEKS){
        throw new BusinessException(LATE_REQUEST);
    }

    if (shipmentEntity.getDocuments().isEmpty()){
        throw new BusinessException(SHIPMENT_DOES_NOT_HAVE_DOCUMENTS);
    }

    shipmentEntity.setConfirmed(true);
    shipmentRepository.save(shipmentEntity);
}
```

THE DEVELOPER'S CONFERENCE

```java
@Override
public void confirmation(Long shipmentId) {
    ShipmentEntity shipmentEntity = shipmentRepository
            .findById(shipmentId)
            .orElseThrow(() -> new NotFoundException(SHIPMENT_NOT_FOUND));

    Boolean validation = new MinimumWeight()
            .and(new PackageMustHaveAtLeastOneItem())
            .and(new RequestIsTimed())
            .and(new ShipmentDoesHaveDocuments().or(new ShipmentDoesHaveAuthorization()))
            .isSatisfiedBy(shipmentEntity);

    if (!validation){
        throw new BusinessException(SHIPMENT_BUSINESS_EXCEPTION);
    }

    shipmentEntity.setConfirmed(true);
    shipmentRepository.save(shipmentEntity);
}
```

Specification com Java 8

- Interface funcional *Predicate*;
- Flexível, sendo utilizado para avaliar condições em um grupo de dados;
- Utilização de expressões lambdas.

THE DEVELOPER'S CONFERENCE

```
Predicate<Shipment> minimumWeight = s -> s.getWeight() >= 10L;
Predicate<Shipment> packageMustHaveAtLeastItems = s -> s.getQuantity() >= 1000000L;
Predicate<Shipment> requestWithinTheTimeLimit = s ->
        ChronoUnit.DAYS.between(s.getRequest(), LocalDateTime.now()) <= 14;
Boolean resultPredicate = minimumWeight
        .and(packageMustHaveAtLeastItems)
        .and(requestWithinTheTimeLimit)
        .test(shipment);
System.out.println(resultPredicate ? SUCCESS_MESSAGE : ERROR_MESSAGE);
```

THE DEVELOPER'S CONFERENCE

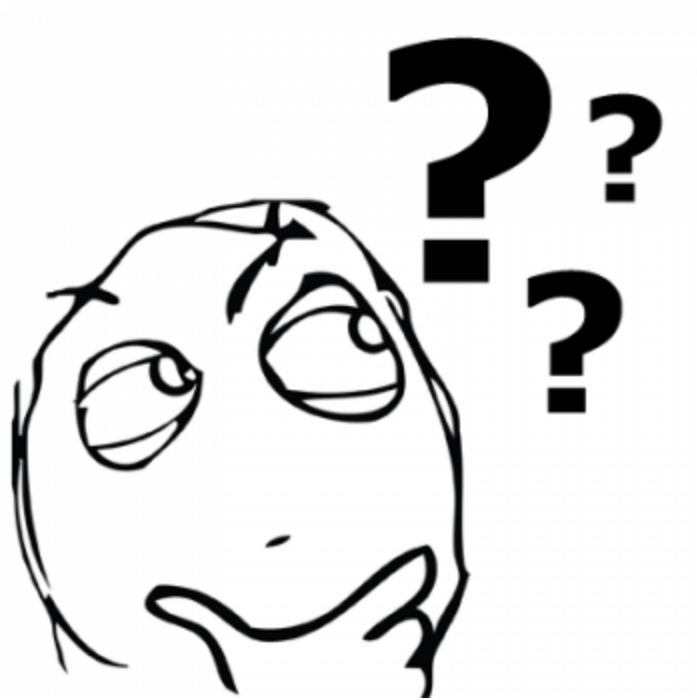
```
Predicate<Shipment> minimumWeight = s -> s.getWeight() >= 10L;
Boolean resultPredicate = minimumWeight
        .and(s -> s.getQuantity() >= 1000000L)
        .and(s -> ChronoUnit.DAYS.between(s.getRequest(), LocalDateTime.now()) <= 14)
        .test(shipment);
System.out.println(resultPredicate ? SUCCESS_MESSAGE : ERROR_MESSAGE);
```

THE DEVELOPER'S CONFERENCE

```
public class ShipmentRules {

    private static final Long MINIMUM_WEIGHT = 10L;
    private static final Integer TWO_WEEKS = 14;

    public static Predicate<Shipment> packageMustHaveAtLeastItems(long total){
        return shipment -> shipment.getQuantity() >= total;
    }

    public static Predicate<Shipment> minimumWeight(){
        return shipment -> shipment.getWeight() >= MINIMUM_WEIGHT;
    }


    public static Predicate<Shipment> requestWithinTheTimeLimit(){
        return shipment -> ChronoUnit.DAYS.between(shipment.getRequest(), LocalDateTime.now()) <= TWO_WEEKS;
    }

}
```

THE
DEVELOPER'S
CONFERENCE

```
boolean resultPredicate = minimumWeight()
        .and(packageMustHaveAtLeastItems( total: 1000000L))
        .and(requestWithinTheTimeLimit())
        .test(shipment);
System.out.println(resultPredicate ? SUCCESS_MESSAGE : ERROR_MESSAGE);
```



Repositório com os exemplos

THE DEVELOPER'S CONFERENCE

Contatos

- E-mail: msgrubler@gmail.com
- GitHub: github.com/murillo
- Linkedin: linkedin.com/in/murillo-grubler
- Twitter: @msgrubler

THE DEVELOPER'S CONFERENCE

Obrigado!