CITERUS

Specification pattern som refactoring-verktyg

Patrik Fredriksson - Citerus AB
patrik.fredriksson@citerus.se

Bilhandlare med specifika krav

- säljer bilar med följande egenskaper

- Röd
- Körts under bra förhållanden
- Inte äldre än fem

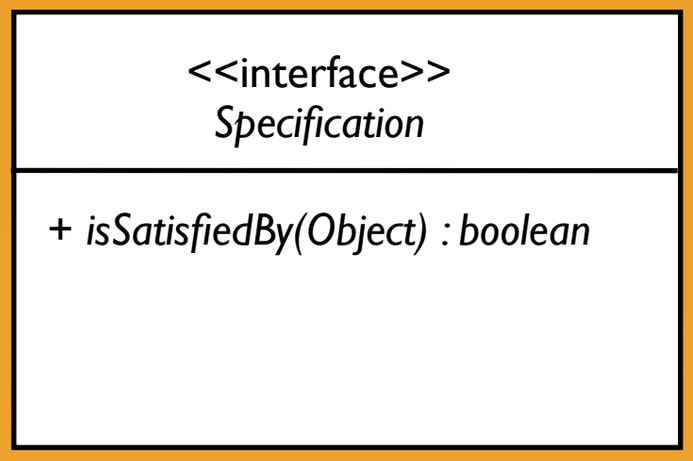
eller

- Röd
- Cabriolet

Första försöket

```
public class CarServiceImpl implements CarService {

  ...

  public Collection<Car> findCandidateCars() {

    final CalendarDate today = Clock.now().calendarDate(TimeZone.getTimeZone("GMT"));

    final Duration maxAge = Duration.years(5);

    final Collection<Car> cars = repository.findAllCarsInStock();
    final Collection<Car> keepers = new HashSet();

    final Set<Region> authorizedRegions = getAuthorizedRegions();

    for (Car car : cars) {
      if (car.color() == Color.RED &&
            (car.isConvertible() ||
               authorizedRegions.contains(car.owner().homeAddress().region()) &&
                  maxAge.startingFrom(car.manufacturingDate()).includes(today)))
          )
        keepers.add(car);

    }
    return keepers;
  }

   ...
}
```

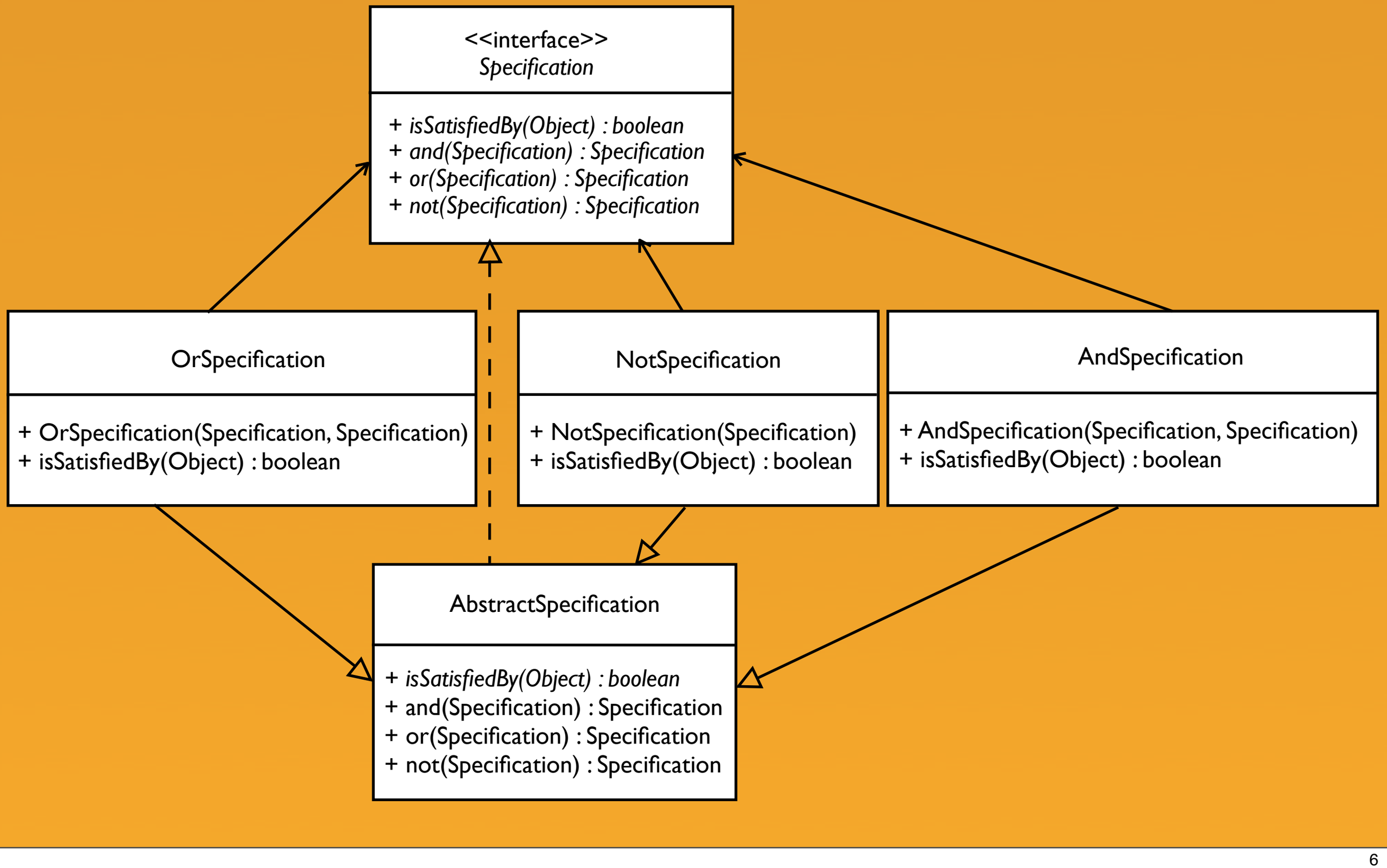
Specification



Första försöket

```
public class CarServiceImpl implements CarService {

  ...

  public Collection<Car> findCandidateCars() {

    final CalendarDate today = Clock.now().calendarDate(TimeZone.getTimeZone("GMT"));

    final Duration maxAge = Duration.years(5);

    final Collection<Car> cars = repository.findAllCarsInStock();
    final Collection<Car> keepers = new HashSet();

    final Set<Region> authorizedRegions = getAuthorizedRegions();

    for (Car car : cars) {
      if (car.color() == Color.RED &&
            (car.isConvertible() ||
               authorizedRegions.contains(car.owner().homeAddress().region()) &&
                  maxAge.startingFrom(car.manufacturingDate()).includes(today)))
          )
        keepers.add(car);

    }
    return keepers;
  }

   ...
}
```

Specification



CarColorSpecification

```
public class CarColorSpecification extends AbstractSpecification {
  private Color color;

  public CarColorSpecification(Color color) {
    this.color = color;
  }

  public boolean isSatisfiedBy(Object o) {
    if (o instanceof Car) {
      Car car = (Car) o;
      return car.color() == color;
    } else {
      throw new ClassCastException("I only deal with cars, you gave me: " +
         o.getClass().getCanonicalName());
    }
  }
}
```

Service med specification

```
public class CarServiceImpl implements CarService {

  ...

  public Collection<Car> findCandidateCars() {

    final Collection<Car> keepers = new HashSet<Car>();

    final CalendarDate today = Clock.now().calendarDate(TimeZone.getTimeZone("GMT"));

    final Specification approvedAge = new CarAgeSpecification(today, 5);
    final Specification colorRed = new CarColorSpecification(RED);
    final Specification convertible = new ConvertibleCarSpecification();
    final Specification approvedRegion =
      new CarOwnerRegionSpecification(getAuthorizedRegions());

    final Specification candidateCarSpecification =
            colorRed.and(approvedRegion.and(approvedAge).or(convertible));

    final Collection<Car> cars = repository.findAllCarsInStock();

    for (Car car : cars) {
      if (candidateCarSpecification.isSatisfiedBy(car))
        keepers.add(car);
    }

    return keepers;
  }

  ...

}
```

CarColorSpecification med Java 5 Generics

```
public class CarColorSpecification extends AbstractSpecification<Car> {

  private Color color;

  public CarColorSpecification(Color color) {
    this.color = color;
  }

  public boolean isSatisfiedBy(final Car car) {
    return car.color() == color;
  }
}
```

Generaliserad service

```
public class CarServiceImpl implements CarService {

...

  /**
   * Check the national used car repository and find
   * cars satistfying the given specification.
   *
   * @param specification Car specification.
   * @return Candidate cars.
   */
  public Collection<Car> findCandidateCars(Specification<Car> specification) {

    final Collection<Car> keepers = new HashSet();
    final Collection<Car> cars = repository.findAllCarsInStock();

    for (final Car car : cars) {
      if (specification.isSatisfiedBy(car))
        keepers.add(car);
    }
    return keepers;
  }

...

}
```

Specification pattern

- Användningsområden inkluderar:
  - Filtrering
  - Konstruktion
  - Validering
- Är ett bra modelleringsverktyg
- Kan med fördel användas även i refactoring-syfte

Mer information

- Presentation i artikelform,
  - http://pnehm.citerus.se/
- Exempelkod och specification.jar,
  - http://code.google.com/p/specification/
- Eric Evans bok,
  - http://www.domaindrivendesign.org/books/index.html#DDD
- Mer om DDD,
  - http://www.domaindrivendesign.org/
- Specification pattern,
  - http://www.martinfowler.com/apsupp/spec.pdf

CITERUS

Specification pattern som refactoring-verktyg

Patrik Fredriksson - Citerus AB
patrik.fredriksson@citerus.se