Nieuwigheden Java 8

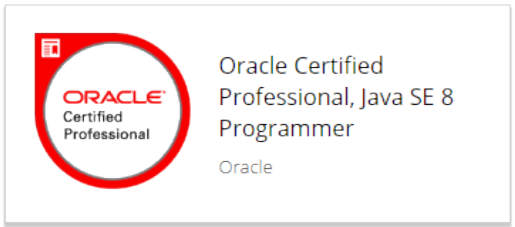
Maarten Dhondt

Realdolmen

October 25, 2018

Wie ben ik?

- Master in de ingenieurswetenschappen: computerwetenschappen (KUL)
  - Computationele informatica
- Software engineer @ Realdolmen sinds 2015
- Projecten:
  - Infrastructuur planning @ Infrabel
  - API platform @ Proximus
- Contact:
  - maarten.dhondt@realdolmen.com
  - maartendhondt
  - MDhondt



Outline

Outline

Outline

Interfaces

- Een interface gelijkt op een klasse, maar bevat enkel methoden en attributen. Interfaces hebben geen geïmplementeerde methoden, maar enkel de signatuur.
  - Implementaties hebben dezelfde signatuur maar return type kan een subklasse zijn.
  - Implementaties gooien geen andere checked exceptions dan diegene uit de interface.
  - Abstracte klassen kunnen methoden implementeren, maar niet vereist.

Interfaces

Implementatie zelfde signatuur maar return type kan subklasse zijn.

```
public abstract class Transaction {}
```

```
public class BankTransaction extends Transaction {}
```

```
public interface Transactionable {
    Transaction doTransaction();
}
```

```
public class BankTranserService implements Transactionable {
    @Override
    public BankTransaction doTransaction() {
        ...
    }
}
```

Interfaces

Implementatie gooit geen andere checked exceptions.

```
public interface ExceptionThrowingInterface {
    void doStuff() throws IOException;
}
```

```
public class ExceptionThrower implements ExceptionThrowingInterface {
    @Override
    public void doStuff() throws IOException, ReflectionException {
        // 'doStuff()' in 'ExceptionThrower' clashes with 'doStuff()' in
             'ExceptionThrowingInterface'; overriden method does not
            throw 'javax.management.ReflectionException'
    }
}
```

Interfaces

Abstracte klasse kan methode implementeren, maar niet vereist

```
public interface Moveable {
    void move();
}
```

```
public abstract class Furniture implements Moveable {}
```

```
public class Chair extends Furniture {
    @Override
    public void move() {
        System.out.println("Moved chair");
    }
}
```

Interfaces

- Java 8 introduceert `default` methoden.
  - Wat? Een implementatie in de interface.
  - Waarom? Optionele methoden
  - Waarom? Gedrag overerving van meerder klassen.
- Vorige regels blijven (uiteraard) geldig.

Interfaces

- Voorbeeld van een `default` methode.

```
public interface Animal {

    void eat();

    void move();

    void sleep();

    default void blinkEyes() {
        System.out.println("Blink");
    }
}
```

Interfaces

- `default` methoden: optionele methoden.

```
public interface Collection<E> extends Iterable<E> {

    default boolean removeIf(Predicate<? super E> filter) {
        Objects.requireNonNull(filter);
        boolean removed = false;
        final Iterator<E> each = iterator();
        while (each.hasNext()) {
            if (filter.test(each.next())) {
                each.remove();
                removed = true;
            }
        }
        return removed;
    }

}
```

Interfaces

Java 8 introduceert ook SAM (Single Abstract Method) interfaces die we functionele interfaces noemen.

- Interface moet exact 1 abstracte methode hebben.
- Met of zonder `@FunctionalInterface` annotatie.

Kunnen gebruikt worden in lambda expressies en method references

Interfaces

```
package java.lang;

@FunctionalInterface
public interface Runnable {
    public abstract void run();
}
```

```
public class Main {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newSingleThreadExecutor();
        executor.submit(() -> {
            System.out.println(Thread.currentThread().getName());
        });
    }
}
```

Outline

Lambda expressies

Lambda expressies zijn een nieuwe en belangrijke functie uit Java 8 die:

- op een duidelijke en bondige manier een interface methode beschrijven in een expressie,
- een grote verbetering mogelijk maken van de Collection libraries.

Lambda expressies

- Lambda expressies bieden een oplossing aan de verbose anonieme inner klassen door 5 lijnen code te reduceren naar 1 lijn.
- Deze horizontale oplossing, lost het verticale probleem van inner klassen op.
- Een lambda expressie bestaat uit 3 delen:
  - Argumenten lijst
  - Pijltje: ->
  - Body

```
(int x, int y) -> x + y
```

Lambda expressies

```
public interface LambdaInterface {
  String doStuff(Integer x, String y);
}
```

```
public class Main {
  public static void main(String[] args) {
    LambdaInterface anonymousImpl = new LambdaInterface() {
      @Override
      public String doStuff(Integer x, String y) {
        return "x=" + x + ",y=" + y;
      }
    };

    LambdaInterface lambdaImpl = (x, y) -> "x=" + x + ",y=" + y;

    System.out.println(anonymousImpl.doStuff(5, "Abc"));
    System.out.println(lambdaImpl.doStuff(5, "Abc"));
  }
}
```

Outline

Outline

Outline

Outline