Java Advanced

Lambda expressies

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

anonymous inner class

Voorbeeld: Android

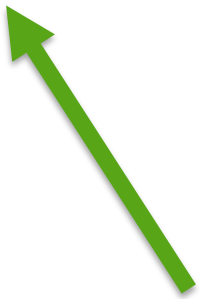
```
saveButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        // TODO: Save current state
        System.out.println("Saved...");
    }
});
```

Interface

```
public interface OnClickListener {
    void onClick(android.view.View view);
}
```

Interface

```
public interface OnClickListener {
    void onClick(android.view.View view);
}
```



FUNCTIONELE interface

Functionele interface

- Slechts 1 abstracte methode
- `@FunctionalInterface` annotatie
  - Compiler geeft foutmelding bij fout

```
@FunctionalInterface
public interface Rounder {
    public int round(double target);
}
```

- Maakt lambda expressie mogelijk

Opgave

- Maak een klasse User met 2 membervariabelen: name en role. Maak een constructor met de 2 parameters en voorzie getters.
- Maak een functionele interface DisplayOnly. De interface bevat de methode void print(User user).
- Probeer ook een tweede methode toe te voegen. Hoe kan je ervoor zorgen dat dat niet mogelijk is?

Lambda expressies

```
saveButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        // TODO: Save current state
        System.out.println("Saved...");
    }
});
```

Veel overbodige code!

Lambda expressies

```
saveButton.setOnClickListener(view -> {
    System.out.println("Saved...");
});
```

GEEN overbodige code!

Lambda expressies

```
public interface NumberPlotter {
    String plot(int number);
}


public static void printNumber(NumberPlotter p) {
    int number = 23;
    System.out.println(p.plot(number));
}
```

```
public static void main(String[] args) {
    printNumber(n -> String.format("[%d]", n));
}
```

Opgave

- Maak nu een hoofdprogramma met een lambda implementatie van de functionele interface DisplayOnly waarmee de naam en de rol van een user wordt afgedrukt. Bv. Ben [admin]

Standaard functionele interfaces

- **Argumenten** en **return type** van belang

- Abstractie: Generieke interfaces
    - ter vervanging van zelf geschreven functionele interfaces

Standaard functionele interfaces

- Predicate<T>

  het testen van een eigenschap

```
public boolean test(T t)
```

- Function<T, R>

  verwerken van een type tot een ander resultaat

```
public R apply(T t)
```

- Consumer<T>

  verwerken van een type zonder return waarde

```
public void accept(T t)
```

Predicate<T> ➔ test

```
ArrayList<String> words = new ArrayList<>();
public void filterWords(Predicate<String> filter) {
    for(String word:words) {
        if(filter.test(word)) {
            System.out.println(word + " is valid!");
        }
    }
}
```

```
filterWords(w -> w.contains("e"));
```

Opgave

- Voeg in het hoofdprogramma een Predicate toe dat true geeft indien een user de role admin heeft. Test uit.
- Maak nu een Predicate dat true geeft indien een user de role member heeft en zijn/haar naam start met een "B".

Function<T,R> ➔ omzetting

```
ArrayList<Double> numbers = new ArrayList<>();
public void roundNumbers(Function<Double, Integer> rounder) {
  for(double number:numbers) {
    int intValue = rounder.apply(number);
    System.out.println("Rounded " + number + ": " + intValue);
  }
}
```

```
roundNumbers(d -> new Double(Math.floor(d)).intValue());
```

Opgave

- Maak een Function om voor een user een wachtwoord te genereren. Het wachtwoord bestaat uit de twee eerste letters van de naam (in uppercase), gevolgd door 4 cijfers. Test uit.

Consumer<T> ➔ void

```
public void printNumbers(Consumer<Double> printer) {
    for(double number:numbers) {
        printer.accept(number);
    }
}
```

```
printNumbers(n -> {
    String formatted = String.format("%.02f", n);
    System.out.println(formatted);
});
```

Opgave

- Maak een Consumer die de naam van een user achterstevoren afdrukt. Test uit.