ERHVERVSAKADEMI ÅRHUS

Lambda funktioner og funktionel interfaces

ERHVERVSAKADEMI ÅRHUS

Dagens emner

Collections opgaver fra sidst

- Opgave 1 (Anvend Set)
- Opgave 2.1 (Link med List)
- Opgave 2.2 (Link med Set)
- Opgave 2.3(Link med Map)
- Opgave 3 (Comparator)

- Lambda
  - Funktionelt interface
  - Lambda funktion

Java interfaces

- Abstrakte metoder uden krop
- Default metoder (erklæret med nøgleordet default)
- Statiske metoder (erklæret med nøgleordet static)

ERHVERVSAKADEMI ÅRHUS

Funktionelt interface

Et interface med præcis en abstrakt metode

Foreksempel:

```
public interface Filter {
    boolean accept(String s);
}
```

```
public interface EventHandler<AvtionEvent> {
    public void handle(ActionEvent event);

}
```

Lambda funktion

En lambda funktion er en metode med navn, parametre og returtype

Kun variable af funktionel interface kan referere en lambda funktion

```
Filter f5 = (String str) -> {
    return str.length() > 5;
};

boolean accepted = f5.accept("Lambda funktion");
System.out.println(accepted);
```

Lambda anvendelse

En lambda funktion kan anvendes som parameter til en metode

```
public class Button {
    public void setOnAction
        (EventHandler<ActionEvent> handler)
    {//...
    }
}

public interface EventHandler<AvtionEvent> {
    public void handle(ActionEvent event);

}
```

```
Button btnPrint = new Button("Print greeting");
btnPrint.setOnAction(event -> this.printAction());
```

Java interfaces og lambda udtryk

- Fra Java 8 kan lambda udtryk anvendes med **default metoder** i Java interfaces
- Default metoder (erklæret med nøgleordet default)

Default metoder
i Java Collection Framework (JCF)

I interfacet Iterable<E> findes default metoden:

```
default void forEach(Consumer<E> action)
```

Consumer<E> er et funktionelt interface med metoden

```
void action(E e)
```

```
List<Person> persons = ... // liste med person objekter
persons.forEach(p -> System.out.println(p.getName());
```

```
List<Person> persons = ... // liste med person objekter
persons.forEach(p -> {
        if (p.getAge())> 18
                System.out.println(p.getName());
});
```

Default metoder i JCF

I interfacet List<E> findes default metoden:

```
default void sort(Comparator<E> c)
```

Comparator<E> er et funktionelt interface med metoden

```
int compare(E e1,E e2)
```

```
List<Person> persons = ... // liste med person objekter
persons.sort((p1, p2) -> p1.getAge() - p2.getAge());
```

Default metoder i JCF

I interfacet List<E> findes default metoden:

```
default void removeIf(Predicate<E> filter)
```

Predicate<E> er et funktionelt interface med metoden

```
boolean test(E e)
```

```
List<Person> persons = ... // liste med person objekter
persons.removeIf(p ->
        p.getAge() < 18 || p.getAge() > 60);
```