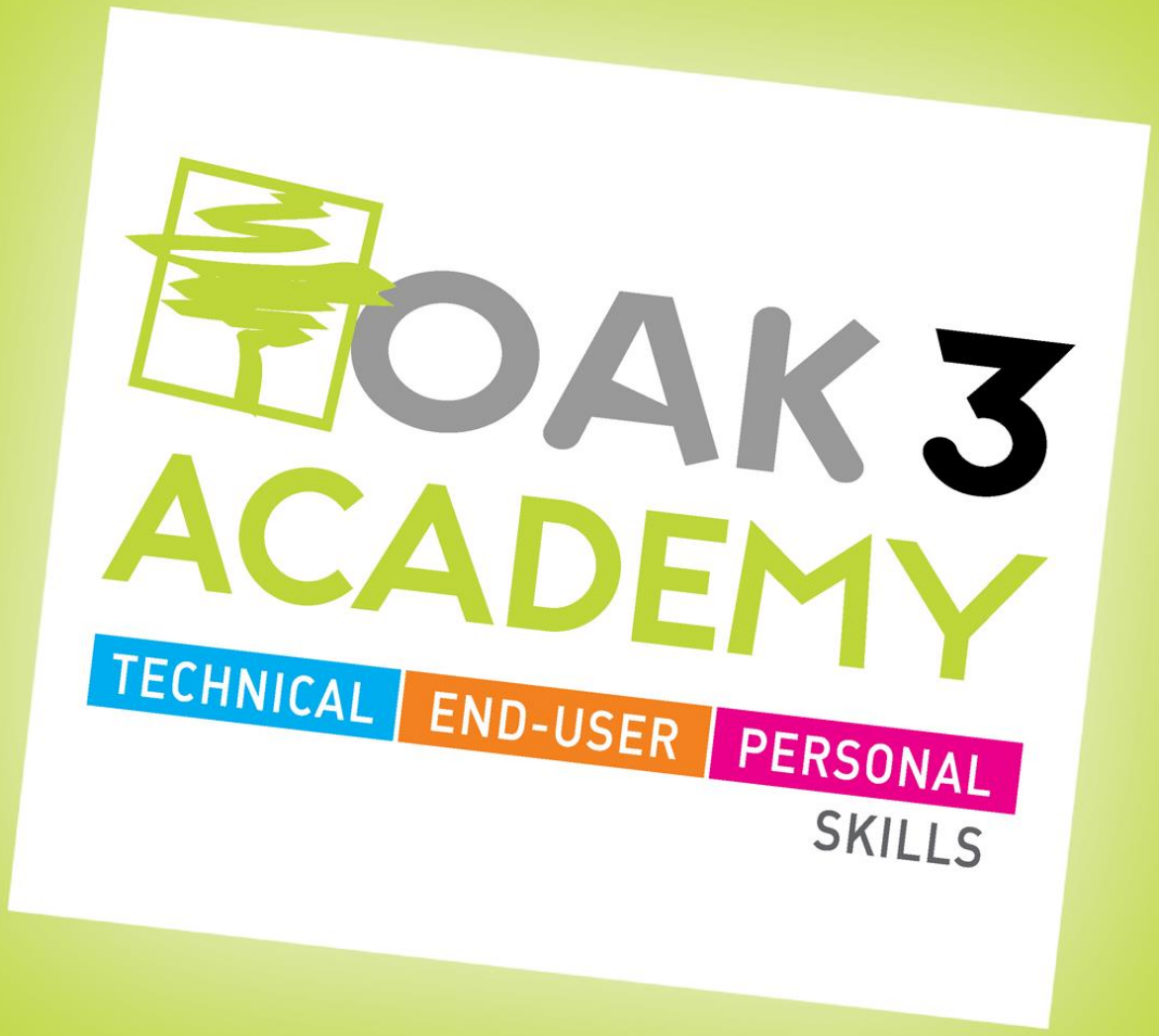


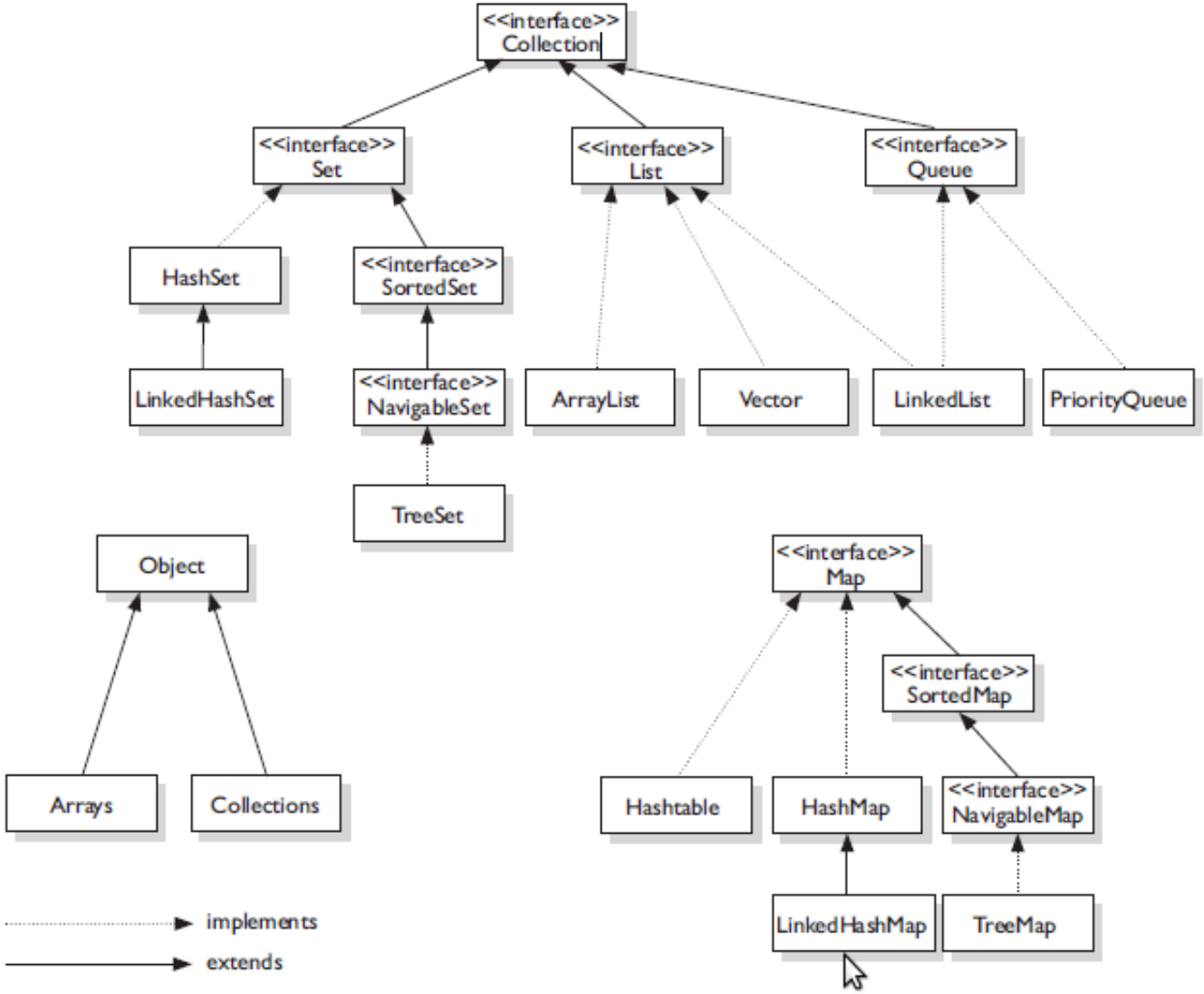
Collections & Generics

Table of contents

Collections Framework

Java Programming

Collections



Collection interface

Java Programming

Collection interface

Volgende interfaces erven over van Collection:

- **List**: Geordende verzameling van elementen. Duplicaten mogelijk.
- **Set**: Verzameling van unieke objecten
- **Queue:** Geordende verzameling waarbij elementen 1 voor 1 opgevraagd kunnen worden.

List interface

- ArrayList: Zeer snelle willekeurige toegang op basis van index.

= array die zich dynamisch aanpast.

- LinkedList: elementen zijn aan elkaar gekoppeld. Lijst kan in 2 richtingen doorlopen worden.

→ Ideaal voor willekeurig toevoegen en verwijderen van elementen

Demo

List met en zonder generics.

Opdracht 1

Maak opdracht 1 in cursus.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Set interface

- Hashset: ongeordende en ongesorteerde verzameling.

→ Zeer efficiënt voor opzoeken, toevoegen en verwijderen van elementen.

- LinkedHashSet: geordende en ongesorteerde verzameling.

→ Als volgorde ook van belang is.

- TreeSet: geordende en gesorteerde verzameling.

Opdracht 2, 3 en 4

Maak de opdrachten in de cursus.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Queue interface

- PriorityQueue: Gesorteerd volgens natuurlijke volgorde of comparator.
- LinkedList: First in First out
- ArrayDeque: Snelle random access aan kop en staart.

Opdracht 5 en 6

Maak de opdrachten in de cursus.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Verzamelingen sorteren

Java Programming

Sorteren

- Natuurlijke volgorde: klasse moet de interface Comparable<T> implementeren.

- Eigen volgorde: Toevoegen van de Comparator<T> interface.

→ TreeSet, PriorityQueue … gebruiken automatisch natuurlijke volgorde!

Voorbeeld: Person klasse

- Person: firstName, lastName, age

```
@Override
public int compareTo(Person person) {
    return this.getLastname().compareTo(person.getLastname());
}
```

→ Natuurlijke volgorde= sorteren op achternaam.

Voorbeeld: Natuurlijke volgorde

```
Person jos = new Person("Jos", "Janssens", 28);
Person monnik = new Person("Monnik", "Mertens", 27);
Person jelle = new Person("Jelle", "De Bie", 1);

List<Person> persons = new ArrayList<>();
persons.add(monnik);
persons.add(jos);
persons.add(jelle);

Collections.sort(persons);
```

→ Compile error als Person class Comparable interface NIET implementeerd.

Voorbeeld: Eigen volgorde

```
Person jos = new Person("Jos", "Janssens", 28);
Person monnik = new Person("Monnik", "Mertens", 27);
Person jelle = new Person("Jelle", "De Bie", 1);

List<Person> persons = new ArrayList<>();
persons.add(monnik);
persons.add(jos);
persons.add(jelle);

Collections.sort(persons, new Comparator<Person>() {

    @Override
    public int compare(Person p1, Person p2) {
        return p1.getAge() - p2.getAge();
    }
});
```

Opdracht 7, 8 en 9

Maak de opdrachten in de cursus.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS

Map interface

Java Programming

Map interface

= Verzameling van key-value paren.

→ Keys zijn uniek

Map interface

Volgende klassen implementeren MAP interface

- **HashMap**: Ongeordende en ongesorteerde map.
- **LinkedHashMap**: Geordende en ongesorteerde map.
- **TreeMap:** Geordende en gesorteerde map.

Opdracht 11 en 12

Maak de opdrachten in de cursus.

OAK 3 ACADEMY
TECHNICAL END-USER PERSONAL SKILLS