TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Fakultät Informatik

Professur Softwaretechnologie

OOSE_03

JAVA-COLLECTION-FRAMEWORK

AM BEISPIEL VON TODO-LISTEN

Dr.-Ing. Birgit Demuth

Sommersemester 2017

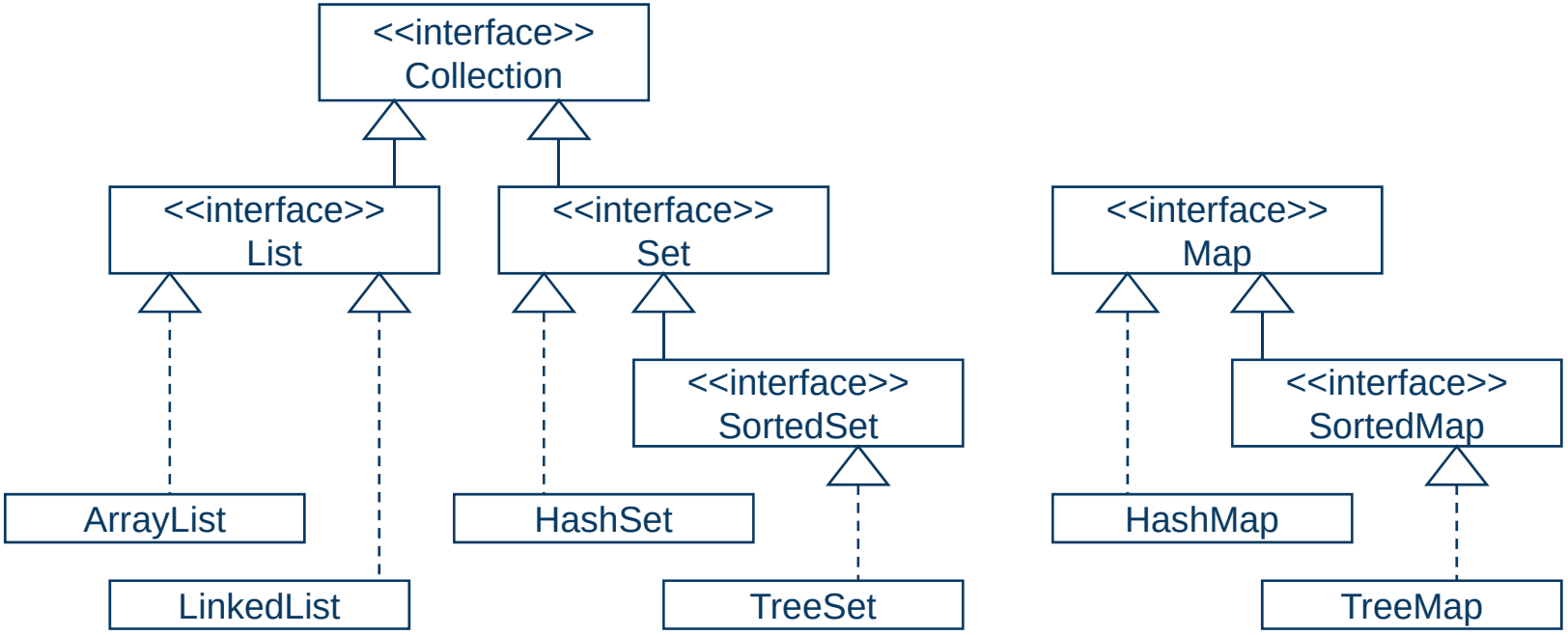
Phasen in der Softwareentwicklung und Anwendung der UML

Phase	Modellierung in UML (Klassendiagramme)
Analyse (OOA)	aUML (UML-Analyseklassendiagramm)
Entwurf (OOD)	dUML (UML-Entwurfsklassendiagramm)
Implementierung und Test (OOP)	jUML (UML-Implementationsklassendiagramm für Java)

Literatur:

Birgit Demuth (Hrsg.): Softwaretechnologie für Einsteiger.
Pearson Studium, 2. geänderte Auflage, 2014
Abschnitt Klassendiagramme, S. 73 – 112

Überblick zum Collection Framework von Java (Ausschnitt)



Eigenschaften von Java-Datenstrukturen [AMCS]

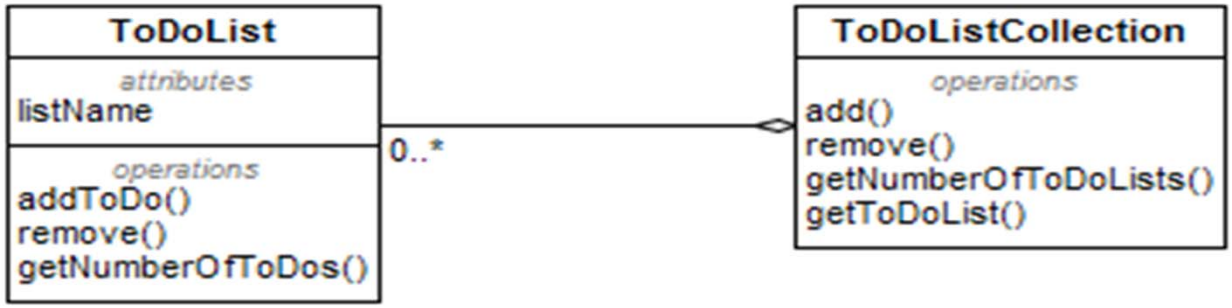
Welche Eigenschaften haben Objekte einer Klassenimplementierung des Interfaces

- java.util.Set
- java.util.List
- java.util.Map

Was sind in Java dynamische Datenstrukturen?

Welche Klasse des Java-Collection-Frameworks enthält Algorithmen, die auf einer Familie von anderen Klassen arbeiten?

ToDo-Listen (aUML) [vgl. Klausur WS 2014/15]

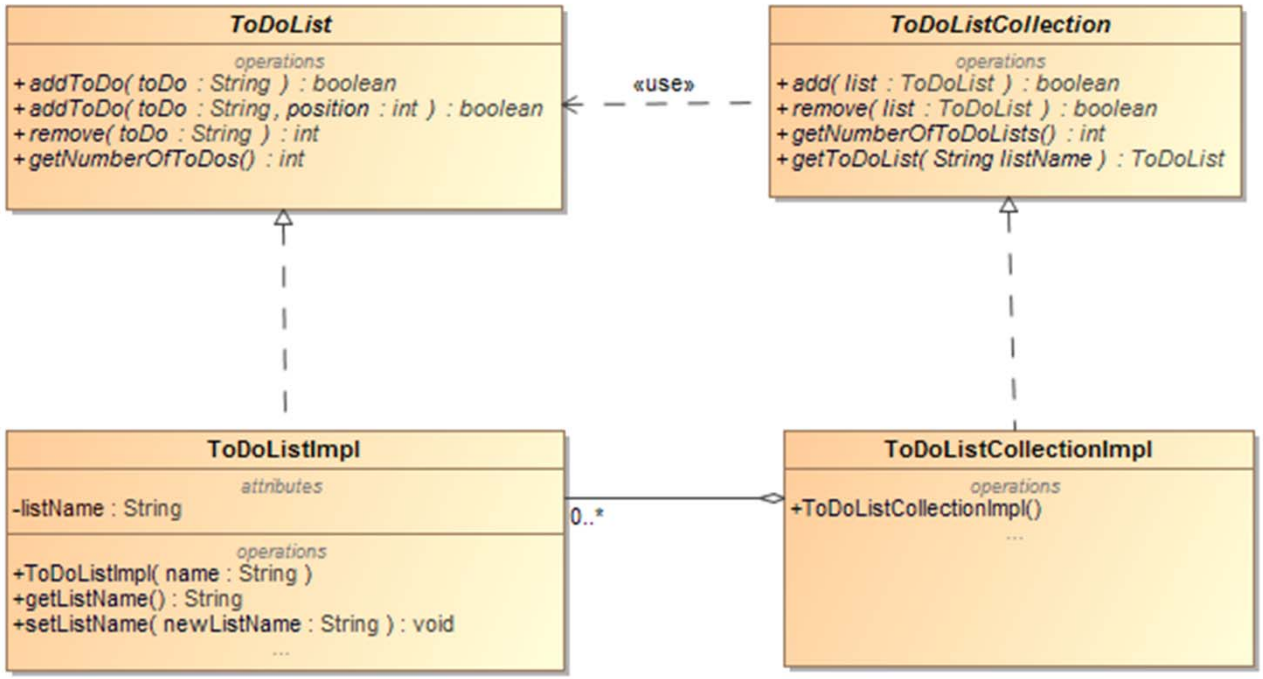


ToDoList: Eine ToDo-Liste beinhaltet in geordneter Reihenfolge „toDos“.

ToDoListCollection: Die ToDo-Listen einer Person werden in einer Kollektion verwaltet.

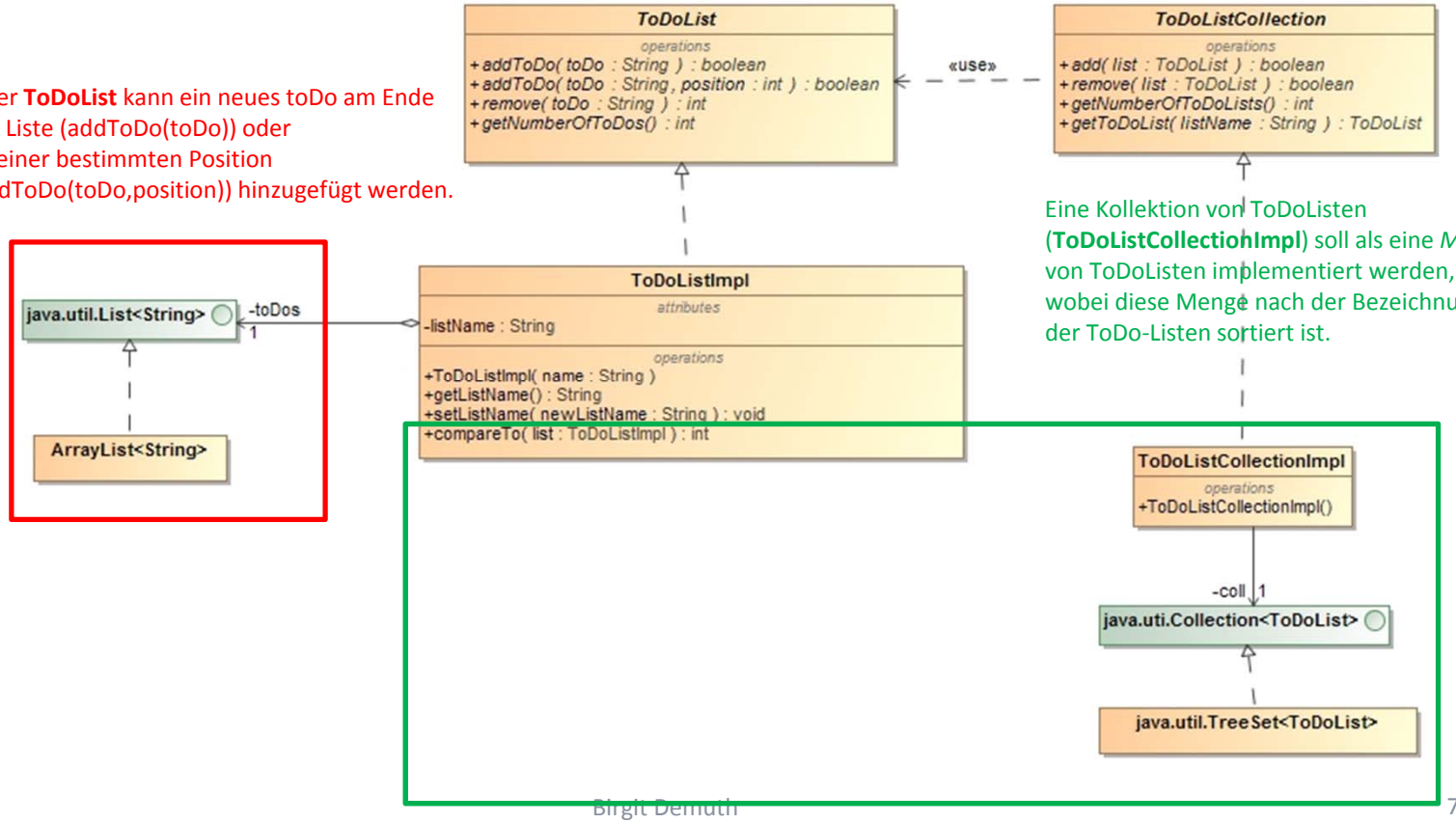
Über die Methode getToDoList() soll über den Listennamen (listname) auf eine konkrete ToDoListe zugegriffen werden.

ToDo-Listen (dUML)



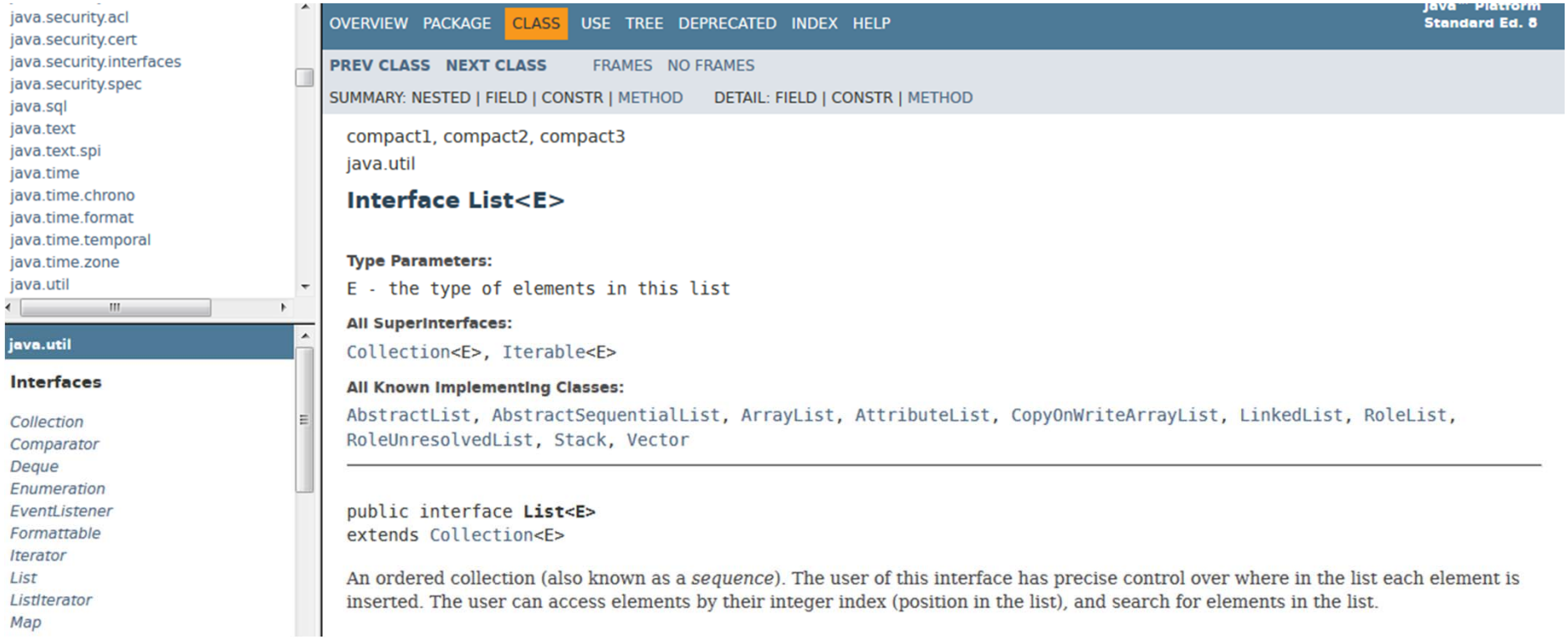
ToDo-Listen (jUML)

Einer **ToDoList** kann ein neues toDo am Ende der Liste (addToDo(toDo)) oder an einer bestimmten Position (addToDo(toDo,position)) hinzugefügt werden.



Eine Kollektion von ToDoListen (**ToDoListCollectionImpl**) soll als eine *Menge* von ToDoListen implementiert werden, wobei diese Menge nach der Bezeichnung der ToDo-Listen sortiert ist.

Nutze javadoc



java.util.List<E> (Auszug)

[https://docs.oracle.com/javase/8/docs/api/java/util/List.html]

```
public interface List<E> {

  public boolean add (E o)
  public boolean add (int index, E o)
  public boolean remove (Object o)
  public E remove (int index)
  public void clear()
  public boolean isEmpty()
  public boolean contains (Object o)
  public int size()
  ...
  public boolean equals (Object o)
  public E get(int index)
  ...
  public Iterator<E> iterator()
}
```

java.util.Set<E> (Auszug)

[https://docs.oracle.com/javase/8/docs/api/java/util/Set.html]

```
public interface Set<E> {

  public boolean add (E o);
  public boolean addAll(Collection<? extends E> c)
  public boolean remove (Object o);
  public void clear();
  public boolean isEmpty();
  public boolean contains (Object o);
  public int size();
  public boolean equals (Object o);
  public int hashCode();
  ...
  public Iterator<E> iterator();
}
```

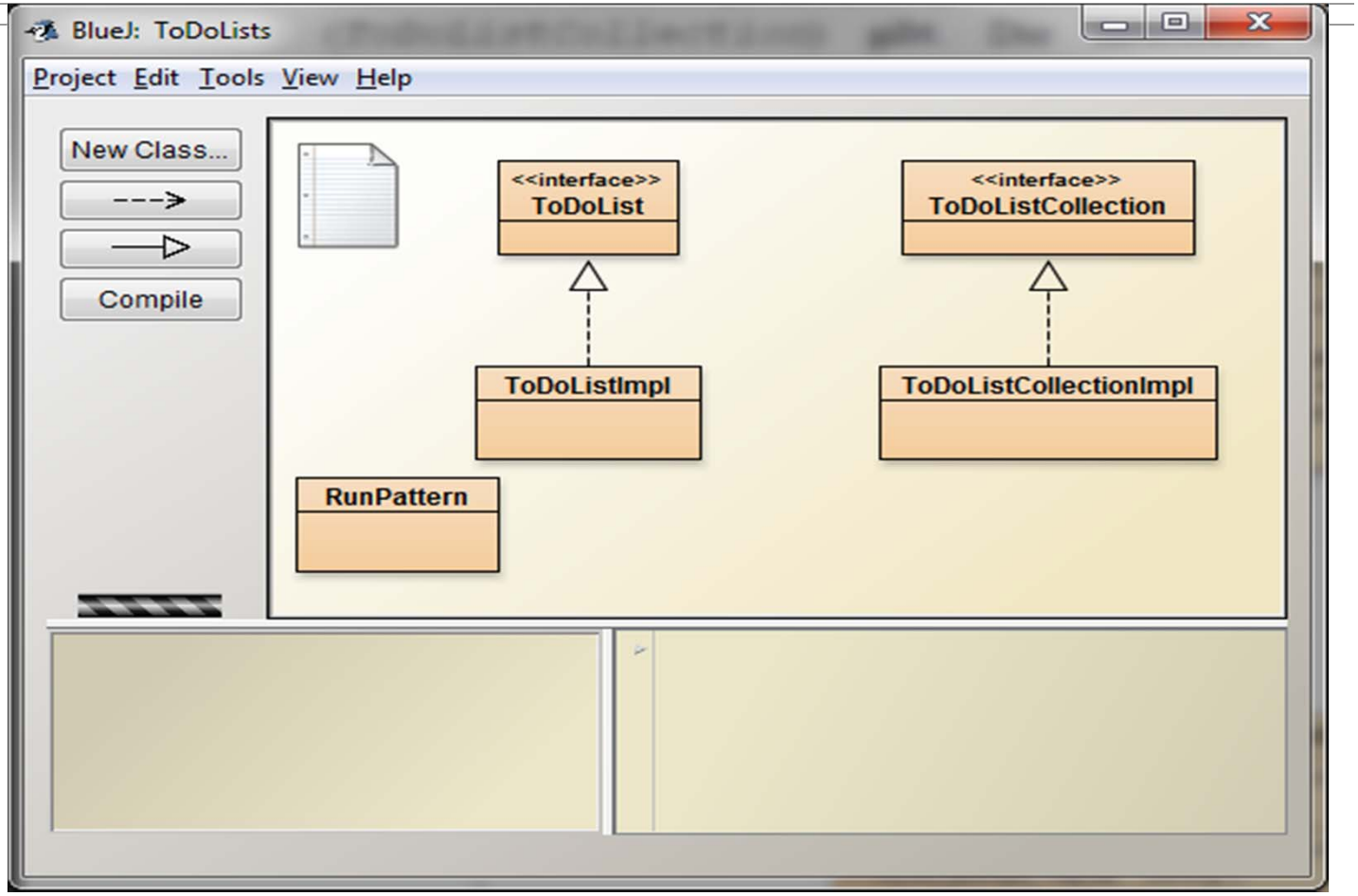
java.lang.Comparable<T>

https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

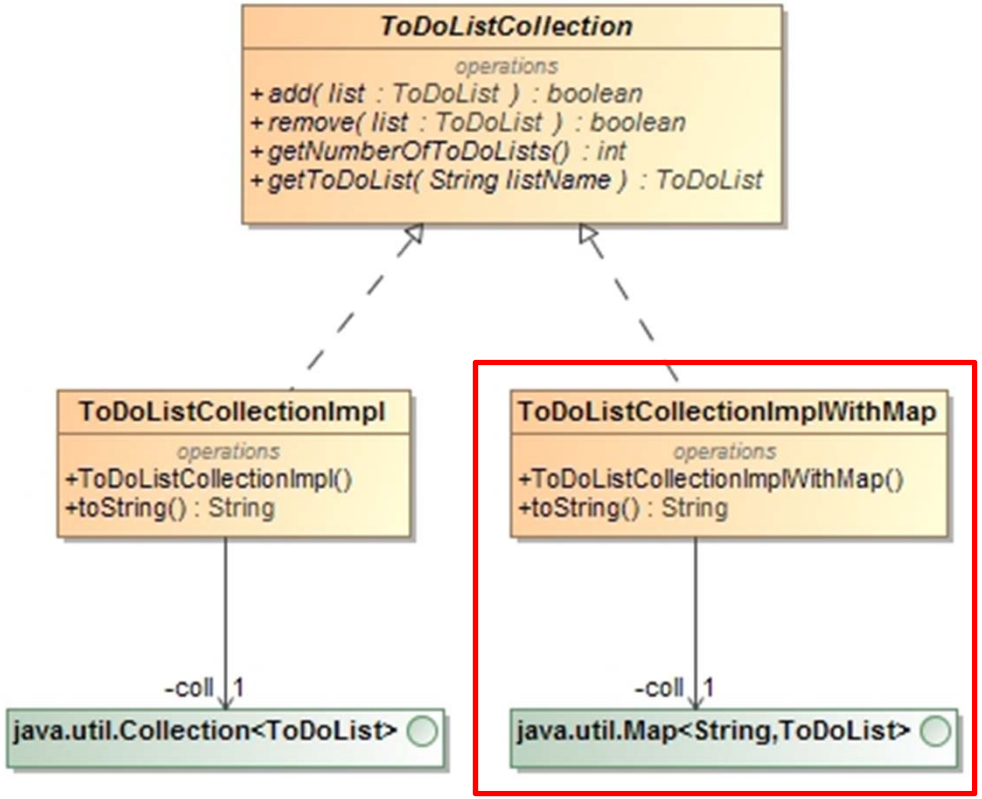
```
public interface Comparable<T> {

   public int compareTo (T o);
}
```

- Interface für den Test der Ordnung auf Elementen
- T ist der Typ des Objektes, mit dem verglichen wird
- Resultat kleiner/gleich/größer 0:
  "this" kleiner/gleich/größer als Objekt o
- Standarddatentypen (z.B. String) implementieren Comparable

ToDo-Listen in BlueJ



Erweiterung von ToDo-Listen (jUML) (gegenüber Klausuraufgabe)



Weitere ToDoCollection-Implementierung:
über den Namen der ToDoList soll direkt auf die ToDoList zugegriffen werden.

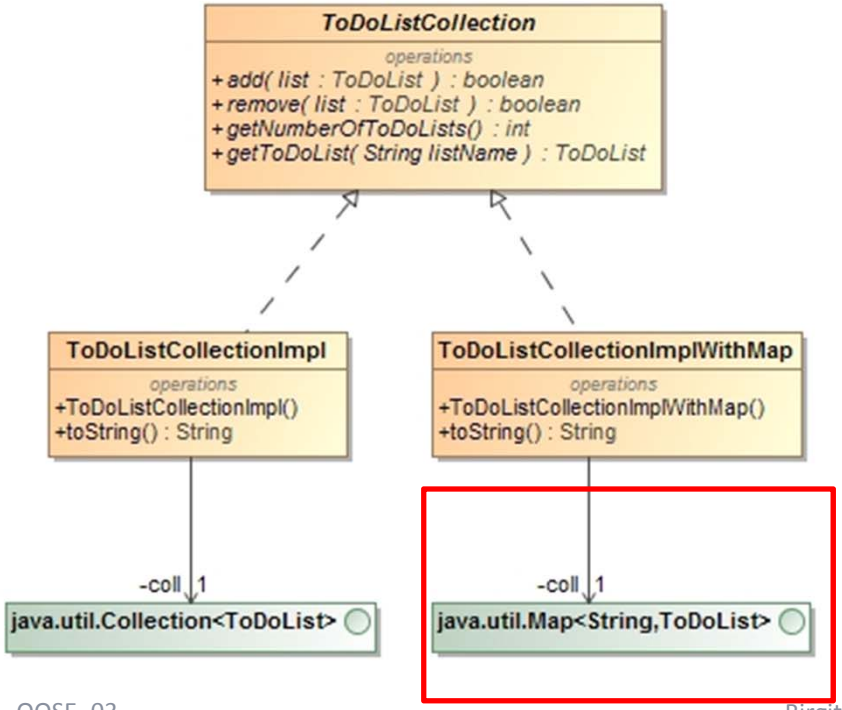
java.util.Map (Auszug)

[https://docs.oracle.com/javase/8/docs/api/java/util/Map.html]

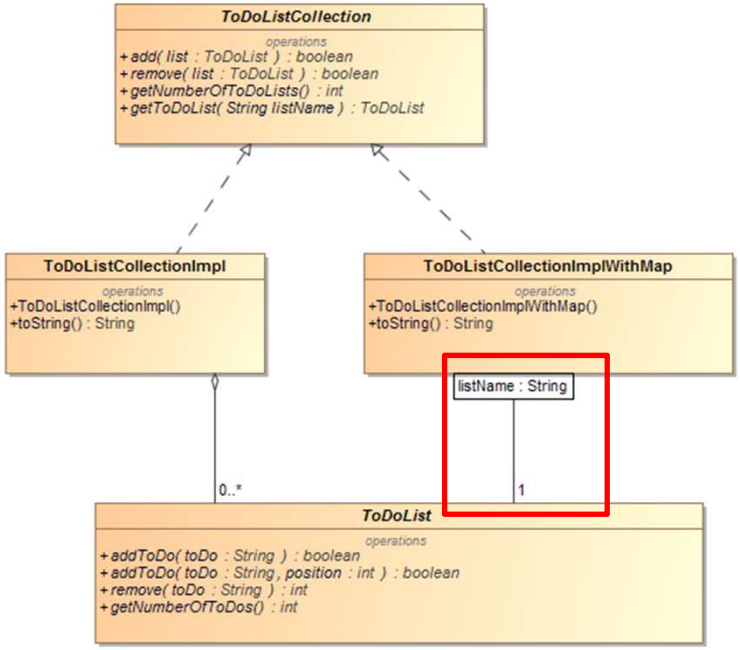
```
public interface Map<K,V> {
  ...
  public boolean containsKey (Object key);
  public boolean containsValue (Object value);
  public V get (Object key);
  public V put (K key, V value);
  public V remove (Object key);
  public int size();
  public Set<K> keySet();
  public Collection<V> values();
  ...
}
```

ToDoListCollection Implementationen

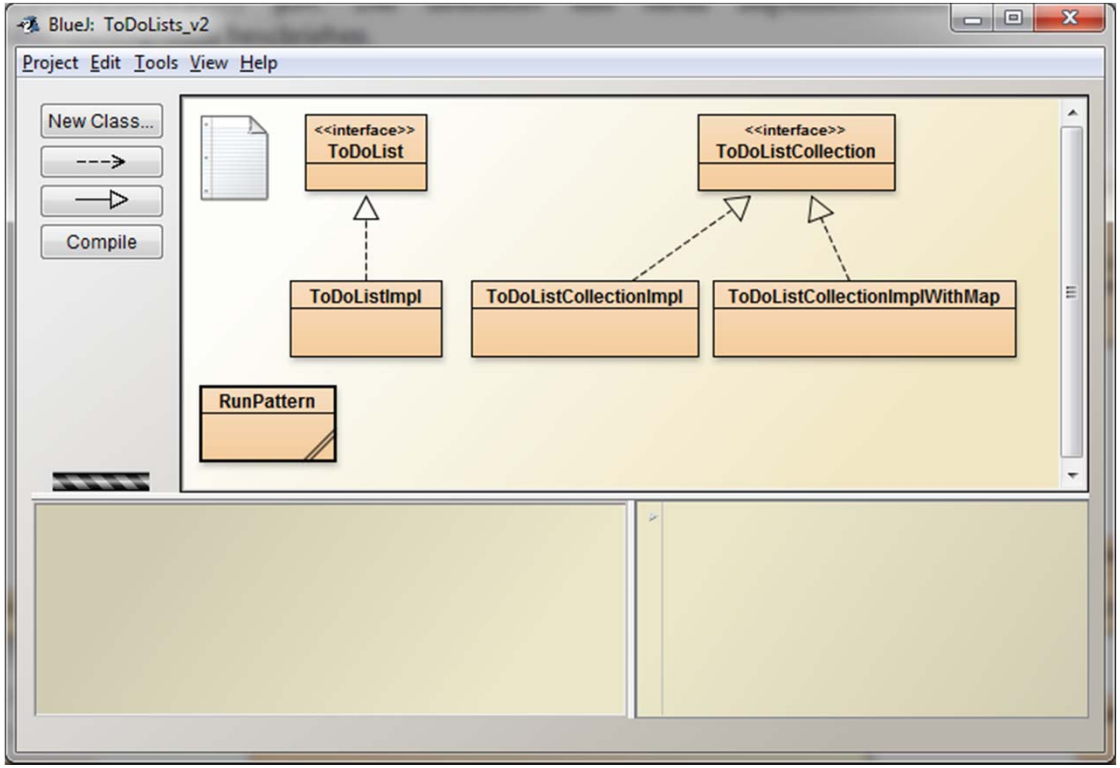
jUML



dUML



ToDo-Listen erweitert (v2) in BlueJ



Ende