

Die Klasse **BinarySearchTree<ContentType extends ComparableContent<ContentType>>**

Mithilfe der generischen Klasse **BinarySearchTree** können beliebig viele Objekte des Typs **ContentType** in einem Binärbaum (binärer Suchbaum) entsprechend einer Ordnungsrelation verwaltet werden.

Ein Objekt der Klasse **BinarySearchTree** stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt vom Typ **ContentType** sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der Klasse **BinarySearchTree** sind.

Die Klasse der Objekte, die in dem Suchbaum verwaltet werden sollen, muss das generische Interface **ComparableContent** implementieren. Dabei muss durch Überschreiben der drei Vergleichsmethoden **isLess**, **isEqual**, **isGreater** (s. Dokumentation des Interfaces) eine eindeutige Ordnungsrelation festgelegt sein.

Beispiel einer solchen Klasse:

```
public class Entry implements ComparableContent<Entry> {

    int wert;
    // diverse weitere Attribute

    public boolean isLess(Entry pContent) {
        return this.getWert() < pContent.getWert();
    }

    public boolean isEqual(Entry pContent) {
        return this.getWert() == pContent.getWert();
    }

    public boolean isGreater(Entry pContent) {
        return this.getWert() > pContent.getWert();
    }

    public int getWert() {
        return this.wert;
    }
}
```

Die Objekte der Klasse **ContentType** sind damit vollständig geordnet. Für je zwei Objekte **c1** und **c2** vom Typ **ContentType** gilt also insbesondere genau eine der drei Aussagen:

- **c1.isLess(c2)** (Sprechweise: c1 ist kleiner als c2)
- **c1.isEqual(c2)** (Sprechweise: c1 ist gleichgroß wie c2)
- **c1.isGreater(c2)** (Sprechweise: c1 ist größer als c2)

Alle Objekte im linken Teilbaum sind kleiner als das Inhaltsobjekt des Binärbaumes. Alle Objekte im rechten Teilbaum sind größer als das Inhaltsobjekt des Binärbaumes. Diese Bedingung gilt auch in beiden Teilbäumen.