Einführung in die Programmierung 2

Gabriel Natter

Juni 2024

1 Übersicht

Inhaltsverzeichnis

2 Iteratoren

2.1 Single Iterator

Der Single Iterator ist der leichteste von allen. Diese iteriert in der Regel genau ein mal über eine Struktur.

Je nach Kommentaren im jeweiligen Iterator Interface kann es aber auch Iteratoren geben, welche kein einziges mal Iterieren. In diesem Fall returnen wir einfach immer false in der *hasNext()* Methode und werfen eine Exception in *next()*.

2.1.1 Beispielcode

```java
public class SingleStarIterator implements StarIterator {

    private final Star star;
    private boolean consumed;

    public SingleStarIterator(Star star) {
        this.star = star;
    }

    @Override
    public boolean hasNext() {
        return !consumed;
    }

    @Override
    public Star next() {
        if(!hasNext()) {
            throw new NoSuchElementException("no star element!");
        }
        consumed = true;
        return star;
    }
}
```

Listing 1: SingleStarIterator.java

```java
public class Star implements Decoration
{

    ...

    @Override
    public StarIterator iterator() {
        return new SingleStarIterator(this);
    }

}
```

Listing 2: Aufrufen des Iterators in Star.java

2.2 List Iterator

Dieser funktioniert für die meisten Iterierbare Strukturen wie z.B. ArrayList oder LinkedList.

2.2.1 Beispielcode

```java
public class MultiStarIterator implements StarIterator {

    private final Iterator<Mobile> attached;
    private StarIterator current;

    public MultiStarIterator(List<Mobile> attached) {
        this.attached = attached.iterator();
        this.current = this.attached.next().iterator();
    }

    @Override
    public boolean hasNext() {
        return attached.hasNext() || current.hasNext();
    }

    @Override
    public Star next() {
        if (!hasNext()) {
            throw new NoSuchElementException("no star element!");
        }
        if (!current.hasNext()) {
            current = attached.next().iterator();
        }
        return current.next();
    }

}
```

Listing 3: MultiStarIterator.java

```java
public class Stick implements Mobile {

    private final List<Mobile> attached;

    ...

    @Override
    public StarIterator iterator() {
        return new MultiStarIterator(attached);
    }

}
```

Listing 4: Aufrufen des Iterators in Stick.java

2.3 Tree Iterator

Dieser ist vom Aufbau her etwas anders, aber in der Regel auch immer gleich zu implementieren.

2.3.1 Beispielcode

```java
class BalancedStickIterator implements StarIterator {

    private final StarIterator leftIter, rightIter;

    public BalancedStickIterator(Mobile left, Mobile right) {
        this.leftIter = left.iterator();
        this.rightIter = right.iterator();
    }

    @Override
    public boolean hasNext() {
        return leftIter.hasNext() || rightIter.hasNext();
    }

    @Override
    public Star next() {
        if (!hasNext()) {
            throw new NoSuchElementException("no star element!");
        }
        if (leftIter.hasNext()) {
            return leftIter.next();
        }
        return rightIter.next();
    }

}
```

Listing 5: BalancedStickIterator.java

```java
public class BalancedStick implements Mobile {

    private Mobile left, right;

    ...

    @Override
    public StarIterator iterator() {
        return new BalancedStickIterator(left, right);
    }

}
```

Listing 6: Aufrufen des Iterators in BalancedStick.java