

src\main\java\shelf\Shelf.java

```
1  package shelf;
2
3  import java.util.Comparator;
4  import java.util.Iterator;
5  import java.util.NoSuchElementException;
6
7  public class Shelf<T extends ShelfItem> implements Iterable<T>{
8
9      private T upperLeft;
10     private T upperRight;
11     private T lowerLeft;
12     private T lowerRight;
13
14     public T get(int index){
15         switch (index){
16             case 0: return upperLeft;
17             case 1: return upperRight;
18             case 2: return lowerLeft;
19             case 3: return lowerRight;
20
21             default: throw new IllegalArgumentException("invalid index");
22         }
23     }
24
25     public void set(int index, T value){
26         switch (index){
27             case 0: upperLeft = value; break;
28             case 1: upperRight = value; break;
29             case 2: lowerLeft = value; break;
30             case 3: lowerRight = value; break;
31
32             default: throw new IllegalArgumentException("invalid index");
33         }
34     }
35
36     @Override
37     public Iterator<T> iterator() {
38         return new ShelfIterator();
39     }
40
41     private class ShelfIterator implements Iterator<T> {
42
43         private int next = 0;
44
45         @Override
46         public boolean hasNext() {
47             return (next < 4);
48         }
49
50         @Override
51         public T next() {
52
53             if (!hasNext())
54                 throw new NoSuchElementException("...");
55
56             return Shelf.this.get(next++);
57         }
58     }
59 }
```

```
58     }
59
60 }
61
62 public void takeFrom(Shelf<? extends T> other){
63     if (other == null)
64         throw new IllegalArgumentException("other must not be null");
65
66     for (int i = 0; i < 4; i++) {
67         this.set(i, other.get(i));
68         other.set(i, null);
69     }
70 }
71
72 public void swap(Shelf<T> other){
73
74     for (int i = 0; i < 4; i++){
75         var temp = this.get(i);
76         this.set(i, other.get(i));
77         other.set(i, temp);
78     }
79
80 }
81
82 public T max(Comparator<T> comparator){
83
84     if (comparator == null)
85         throw new IllegalArgumentException("comparator must not be null");
86
87     T maxItem = null;
88
89     for (int i = 0; i < 4; i++){
90         if (get(i) != null && (maxItem == null || comparator.compare(maxItem, get(i)) < 0))
91             maxItem = get(i);
92     }
93
94 }
95
96 return maxItem;
97
98 }
99
100 public static <S extends ShelfItem> void transferAndTrim(Shelf<? extends S> from, Shelf<
? super S> to){
101
102     if (from == null)
103         throw new IllegalArgumentException("from must not be null");
104
105     if (to == null)
106         throw new IllegalArgumentException("to must not be null");
107
108     int toIndex = 0;
109
110     for (int i = 0; i < 4; i++){
111         to.set(i, null);
112     }
113
114     for (int i = 0; i < 4; i++){
115
```

```
116         if (from.get(i) != null){
117
118             to.set(toIndex, from.get(i));
119
120             from.set(i, null);
121
122             toIndex++;
123
124
125         }
126
127
128     }
129
130
131
132 }
133
134
135
136 public T getUpperLeft() {
137     return upperLeft;
138 }
139
140 public void setUpperLeft(T upperLeft) {
141     this.upperLeft = upperLeft;
142 }
143
144 public T getUpperRight() {
145     return upperRight;
146 }
147
148 public void setUpperRight(T upperRight) {
149     this.upperRight = upperRight;
150 }
151
152 public T getLowerLeft() {
153     return lowerLeft;
154 }
155
156 public void setLowerLeft(T lowerLeft) {
157     this.lowerLeft = lowerLeft;
158 }
159
160 public T getLowerRight() {
161     return lowerRight;
162 }
163
164 public void setLowerRight(T lowerRight) {
165     this.lowerRight = lowerRight;
166 }
167
168 }
169
```