Aufgabe 5

Gegeben sei eine Standarddatenstruktur Stapel (Stack) mit den Operationen

- `void push(Element e)`
- `Element pop()`,
- `boolean isEmpty()`.

sowie dem Standardkonstruktor `Stapel()`, der einen leeren Stapel zur Verfügung stellt.

(a) Geben Sie eine Methode `Stapel merge(Stapel s, Stapel t)` an, die einen aufsteigend geordneten Stapel zurückgibt, unter der Bedingung, dass die beiden übergebenen Stapel aufsteigend sortiert sind, d. h. `S.pop()` liefert das größte Element in `s` zurück und `T.pop()` liefert das größte Element in `t` zurück. Als Hilfsdatenstruktur dürfen Sie nur Stapel verwenden, keine Felder oder Listen.

Hinweis: Nehmen Sie an, dass Objekte der Klasse `Element`, die auf dem Stapel liegen mit `compareTo()` vergleichen werden können. Zum Testen haben wir Ihnen eine Klasse `StapelTest` zur Verfügung gestellt, sie können Ihre Methode hier einfügen und testen, ob die Stapel korrekt sortiert werden. Überlegen Sie auch, was geschieht, wenn einer der Stapel (oder beide) leer ist!

```java
46    public static Stapel merge(Stapel s, Stapel t) {
47      // Die beiden Stapel unsortiert aneinander hängen.
48      Stapel mergedStack = new Stapel();
49      while (!s.isEmpty()) {
50        mergedStack.push(s.pop());
51      }
52      while (!t.isEmpty()) {
53        mergedStack.push(t.pop());
54      }
55      // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
56      Stapel tmpStack = new Stapel();
57      while (!mergedStack.isEmpty()) {
58        Element tmpElement = mergedStack.pop();
59        while (!tmpStack.isEmpty() && tmpStack.top.getValue() >
          ↪  tmpElement.getValue()) {
60          mergedStack.push(tmpStack.pop());
61        }
62        tmpStack.push(tmpElement);
63      }
64      return tmpStack;
65    }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java`

Komplette Klasse Stapel

```java
3   /**
4    * https://www.studon.fau.de/file2860857_download.html
5    */
6   public class Stapel {
```

```
7     public Element top;
8
9     public Stapel() {
10      top = null;
11    }
12
13    /**
14     * @param element Das Element, dass hinzugefügt werden soll zur
      ↪  Stapel.
15     */
16    public void push(Element element) {
17      element.setNext(top);
18      top = element;
19    }
20
21    /**
22     * @return Das Element oder null, wenn der Stapel leer ist.
23     */
24    public Element pop() {
25      if (top == null) {
26        return null;
27      }
28      Element element = top;
29      top = top.getNext();
30      return element;
31    }
32
33    /**
34     * @return Wahr wenn der Stapel leer ist.
35     */
36    public boolean isEmpty() {
37      return top == null;
38    }
39
40    /**
41     * @param s Stapel s
42     * @param t Stapel t
43     *
44     * @return Ein neuer Stapel.
45     */
46    public static Stapel merge(Stapel s, Stapel t) {
47      // Die beiden Stapel unsortiert aneinander hängen.
48      Stapel mergedStack = new Stapel();
49      while (!s.isEmpty()) {
50        mergedStack.push(s.pop());
51      }
52      while (!t.isEmpty()) {
53        mergedStack.push(t.pop());
54      }
55      // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
56      Stapel tmpStack = new Stapel();
57      while (!mergedStack.isEmpty()) {
58        Element tmpElement = mergedStack.pop();
59        while (!tmpStack.isEmpty() && tmpStack.top.getValue() >
          ↪  tmpElement.getValue()) {
60          mergedStack.push(tmpStack.pop());
61        }
62        tmpStack.push(tmpElement);
63      }
64      return tmpStack;
65    }
```

```
66
67    public static void main(String[] args) {
68      Stapel sa = new Stapel();
69      sa.push(new Element(1));
70      sa.push(new Element(2));
71      sa.push(new Element(4));
72      sa.push(new Element(5));
73      sa.push(new Element(7));
74      sa.push(new Element(8));
75      Stapel sb = new Stapel();
76      sb.push(new Element(2));
77      sb.push(new Element(3));
78      sb.push(new Element(6));
79      sb.push(new Element(9));
80      sb.push(new Element(10));
81
82      Stapel sc = Stapel.merge(sa, sb);
83
84      while (!sc.isEmpty()) {
85        System.out.print(sc.pop().getValue() + ", ");
86      }
87    }
88  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java`

Komplette Klasse Element

```
3   /**
4    * https://www.studon.fau.de/file2860856_download.html
5    */
6   public class Element {
7     public int value;
8
9     public Element next;
10
11    public Element() {
12      this.next = null;
13    }
14
15    public Element(int value, Element element) {
16      this.value = value;
17      this.next = element;
18    }
19
20    public Element(int value) {
21      this.value = value;
22      this.next = null;
23    }
24
25    public int getValue() {
26      return value;
27    }
28
29    public Element getNext() {
30      return next;
31    }
32
33
```

```
34    public void setNext(Element element) {
35      next = element;
36    }
37
38    public int compareTo(Element element) {
39      if (getValue() > element.getValue()) {
40        return 1;
41      } else if (element.getValue() == getValue()) {
42        return 0;
43      } else {
44        return -1;
45      }
46    }
47
48  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/Element.java`

Test-Klasse

```
3   import static org.junit.Assert.*;
4   import org.junit.Test;
5
6   /**
7    * https://www.studon.fau.de/file2860850_download.html
8    */
9   public class StapelTest {
10
11    @Test
12    public void testeMethodenPushPop() {
13      Stapel stapel = new Stapel();
14      stapel.push(new Element(1));
15      stapel.push(new Element(2));
16      stapel.push(new Element(3));
17
18      assertEquals(3, stapel.pop().value);
19      assertEquals(2, stapel.pop().value);
20      assertEquals(1, stapel.pop().value);
21    }
22
23    @Test
24    public void testeMethodeMerge() {
25      Stapel sa = new Stapel();
26      sa.push(new Element(1));
27      sa.push(new Element(3));
28      sa.push(new Element(5));
29
30      Stapel sb = new Stapel();
31      sb.push(new Element(2));
32      sb.push(new Element(4));
33
34      Stapel sc = Stapel.merge(sa, sb);
35
36      assertEquals(5, sc.pop().getValue());
37      assertEquals(4, sc.pop().getValue());
38      assertEquals(3, sc.pop().getValue());
39      assertEquals(2, sc.pop().getValue());
40      assertEquals(1, sc.pop().getValue());
41    }
```

```
42
43      @Test
44      public void testeMethodeMergeMehrWerte() {
45        Stapel sa = new Stapel();
46        sa.push(new Element(1));
47        sa.push(new Element(2));
48        sa.push(new Element(4));
49        sa.push(new Element(5));
50        sa.push(new Element(7));
51        sa.push(new Element(8));
52        Stapel sb = new Stapel();
53        sb.push(new Element(2));
54        sb.push(new Element(3));
55        sb.push(new Element(6));
56        sb.push(new Element(9));
57        sb.push(new Element(10));
58
59        Stapel sc = Stapel.merge(sa, sb);
60
61        assertEquals(10, sc.pop().getValue());
62        assertEquals(9, sc.pop().getValue());
63        assertEquals(8, sc.pop().getValue());
64        assertEquals(7, sc.pop().getValue());
65        assertEquals(6, sc.pop().getValue());
66        assertEquals(5, sc.pop().getValue());
67        assertEquals(4, sc.pop().getValue());
68        assertEquals(3, sc.pop().getValue());
69        assertEquals(2, sc.pop().getValue());
70        assertEquals(2, sc.pop().getValue());
71        assertEquals(1, sc.pop().getValue());
72      }
73
74      @Test
75      public void testeMethodeMergeBLeer() {
76        Stapel sa = new Stapel();
77        sa.push(new Element(1));
78        sa.push(new Element(3));
79        sa.push(new Element(5));
80
81        Stapel sb = new Stapel();
82
83        Stapel sc = Stapel.merge(sa, sb);
84
85        assertEquals(5, sc.pop().getValue());
86        assertEquals(3, sc.pop().getValue());
87        assertEquals(1, sc.pop().getValue());
88      }
89
90      @Test
91      public void testeMethodeMergeALeer() {
92        Stapel sa = new Stapel();
93
94        Stapel sb = new Stapel();
95        sb.push(new Element(2));
96        sb.push(new Element(4));
97
98        Stapel sc = Stapel.merge(sa, sb);
99
100       assertEquals(4, sc.pop().getValue());
101       assertEquals(2, sc.pop().getValue());
102     }
```

```
103
104 }
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/StapelTest.java`

(b) Analysieren Sie die Laufzeit Ihrer Methode.

Best case: $\mathcal{O}(1)$ Worst case: $\mathcal{O}(n^2)$