```java

/**
 * Klasse zur Realisierung verschiedener Filteroperationen
 * auf einem Feld.
 */
public class FeldFilter {
    /**
     * Inner Klasse Node.
     * Ein Node objekt repersentiert ein Listen Element.
     * Ein Nodeobjekt besitzt eine Referenz auf das Naechste Element in der List.
     * Falls es kein naechstes Element geben sollte ist diese Referenz eine null-Referenz.
     */
    private class Node{

        Node next = null;
        int val = 0;
        /**
         * Erstellet ein neues Nodeobjekt.
         * @param val Initialer Wert des Node-Objekts.
         */
        Node(int val) {
            this.val = val;
        }
    }

    private int[] feld;

    /**
     * Erstellt einen neuen FeldFilter mit den Werten des Feldes Feld.
     * @param feld
     */
    public FeldFilter(int[] feld) {
        super();
        this.feld = feld.clone();
    }
    /**
     * Kopiert die Elemente der Liste in ein Array.
     * Das erste Element wird nicht kopiert, weil es nur zur
     * Speicherung der laenge der Liste dient.
     * @param start Erste element der Liste
     * @return Referenz auf ein int[] mit den Elementen der Liste.
     */
    private int[] listToArray(Node start){
        int[] retVal = new int[start.val];

        for(int i = 0;(start = start.next) != null;i++) {
            retVal[i] = start.val;
        }

        return retVal;
```

```
52      }
53
54      /**
55       * Gib ein Feld zur&uuml;ck, das nur die Werte aus
56       * dem gespeicherten Feld enth&auml;lt, die echt gr&ouml;&szlig;er
57       * als der &uuml;bergebene Wert sind. Die Reihenfolge der Werte
58       * darf dabei nicht ver&auml;ndert werden.
59       * @param untereSchranke Wert, der als untere Schranke gelten soll.
60       * @return Berechnetes Feld.
61       */
62      public int[] filterGroesserAls(int untereSchranke) {
63
64          //Erstes Element der Liste erstellen die die Werte anthalten soll
    die
65          //großer als untereSchranke sind.
66          //Das erste Element speichert die anzahl der groeßeren Elemente.
67          Node aktuell = new Node(0),
68               start = aktuell;
69          //Schleife sucht jedes element das groeßer als untereSchranke ist.
70          for (int val : feld) {
71              //Wenn der Wert groeßer ist wird dieser an die Liste
    angehaengt und
72              //der Wert des ersten Listenelements um 1 erhoeht.
73              if(val > untereSchranke) {
74                  aktuell = aktuell.next = new Node(val);
75                  start.val++;
76              }
77          }
78
79          //Konvertiert die Liste zu einem Array und gibt dies zurück.
80          return listToArray(start);
81      }
82
83      /**
84       * Gib ein Feld zur&uuml;ck, das nur die Werte aus
85       * dem gespeicherten Feld enth&auml;lt, die weder mit
86       * dem minimalen noch mit dem maximalen Element &uuml;bereinstimmen.
87       * Die Reihenfolge der Werte darf dabei nicht ver&auml;ndert werden.
88       * @return Berechnetes Feld.
89       */
90      public int[] eliminiereMinMax() {
91          //Falls das Feld weniger als drei Elemente enthaelt wird auf jedem
    Fall jeder Wert
92          //aus dem Feld feld entfaehrnt.
93          int[] ausgabe = {}; // Default-Rueckgabe: Leeres Feld.
94          if(feld.length<=2) {
95              return ausgabe;
96          }
97
98          int minVal = Integer.MAX_VALUE;
99          int maxVal = minVal+1; //Integer overflow => Integer.MIN_VALUE
100         int minCount = 0;
101         int maxCount = 0;
```

```
102
103         //Schleife ermittelt den minimalen und den maximalen Wert im Feld
    feld.
104         //Zudem wird die Anzahl dieser Werte gezaehlt damit spaeter die
    Groeße des
105         //neuem Feldes bestimmt werden kann.
106         for (int val : feld) {
107             if(val < minVal) {
108                 minCount = 1;
109                 minVal = val;
110             }else if(minVal == val)minCount++;
111             if(val > maxVal) {
112                 maxCount = 1;
113                 maxVal = val;
114             }else if(maxVal == val)maxCount++;
115         }
116         //Neues feld erstellen.
117         //Falls minVal == maxVal muss nur eine der Beiden zaehler
    (minCount oder maxCount) zur
118         //Berechnung der Feldgroeße genutzt werden da beide werte gleich
    sind.
119         if(minVal == maxVal)
120             ausgabe = new int[feld.length-maxCount];
121         else
122             ausgabe = new int[feld.length-minCount-maxCount];
123         //Werte in das neue Feld kopieren.
124         int i = 0;
125         for (int val : feld) {
126             if(minVal != val && maxVal != val)
127                 ausgabe [i++] = val;
128         }
129         //Werte zurueckgeben.
130         return ausgabe;
131     }
132 }
133
```