```java
package oop6_dat6_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class Cube implements Comparable<Cube>
{

    private int number; //Nummer des Kubus
    private int s1, s2, s3; //Die 3 Seiten des Kubus

    public Cube(int no, int a, int b, int c)
    {
        this.number = no;
        this.s1 = a;
        this.s2 = b;
        this.s3 = c;
    }

    /**
     * Berechnet das Volumen des Kubus
     * @return Volumen
     */
    public int getVolume()
    {
        return (s1 * s2 * s3);
    }

    /**
     * Berechnet die Oberfläche des Kubus
     * @return Oberfläche
     */
    public int getSurface()
    {
        return (2 * (s1 * s2 + s1 * s3 + s2 * s3));
    }

    /**
     * Liefert die grösste Seitenlänge als Ergebnis
     * @return grösse Seitenlänge
     */
    public int getMaxDimension()
    {
        int res;
        if (s1 > s2) {
            if (s1 > s3) {
                res = s1;
            } else {
                res = s3;
            }
        } else {
            if (s2 > s3) {
                res = s2;
            } else {
                res = s3;
            }
        }
        return res;
    }

    /**
     * String des Kubus
     * @return Beschreibung des Kubus
     */
    @Override
    public String toString()
    {
        return ("---------------------------------------\n"+
            "CUBE " + number + "\n" +
            "Sides: " + s1 + " " + s2 + " " + s3 + "\n" +
            "Volume = " + this.getVolume() + "\n" +
            "Surface = " + this.getSurface()+ "\n");
    }

    /**
     * Kubus vergleichen
     * @param other Kubus
     * @return Kubus gleich oder nicht
     */
    @Override
    public boolean equals(Object other)
    {
        if (this == other) // 1. Test auf Identität
        {
            return true;
        }
        if (other == null) // 2. Test auf null
        {
            return false;
        }
        if (other.getClass() != this.getClass())// 3. Test auf Vergleichbarkeit
        {
            return false;
        }
        if ( ! (this.getVolume() == ((Cube) other).getVolume()))// 4. Vergleich relev.
Felder
        {
            return false;
        }
        return true;
```

```
101     }
102
103     /**
104      * Hash-Code wird bassierend auf der (eindeutigen) Nummer definiert
105      * @return Hash-Code
106      */
107     @Override
108     public int hashCode()
109     {
110         return number;
111     }
112
113     /**
114      * Kubus wird auf ihre Volumen verglichen
115      * @param c Cube zum Vergleichen
116      * @return Differenz des Volumen
117      */
118     public int compareTo(Cube c)
119     {
120         return (this.getVolume() - c.getVolume());
121     }
122
123
124     /**
125      * kleine "test" Methode
126      * @param args
127      */public static void main(String[] args)
128     {
129         Cube c1 = new Cube(1, 1, 2, 3);
130         Cube c2 = new Cube(2, 10, 20, 30);
131         Cube c3 = new Cube(3, 30, 20, 10);
132
133         System.out.println(c1.toString());
134         System.out.println(c2.toString());
135         System.out.println(c3.toString());
136
137         System.out.println("c1 == c2 ? : " + c1.equals(c2));
138         System.out.println("c2 == c3 ? : " + c2.equals(c3));
139         System.out.println("c1 > c2 ? : " + c1.compareTo(c2));
140         System.out.println("c2 > c3 ? : " + c2.compareTo(c3));
141     }
142 }
```

```
1
2  package oop6_dat6_u;
3
4  import java.util.Comparator;
5
6  /**
7   *
8   * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
9   */
10 public class MaxDimensionComparator implements Comparator<Cube> {
11 public int compare(Cube c1, Cube c2)
12     {
13         return (c1.getMaxDimension() - c2.getMaxDimension());
14     }
15 }
```

```java
package oop6_dat6_u;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class SolidFileIO
{

    private static int number;

    public static Cube[] readSolids(String fileName)
    {
        ArrayList<Cube> list = new ArrayList<Cube>();

        try {
            FileReader aFileReader = new FileReader(fileName);
            BufferedReader aBufferedReader = new BufferedReader(aFileReader);

            String line = aBufferedReader.readLine();
            StringTokenizer sT;
            while (line != null) { // while not EOF
                sT = new StringTokenizer(line, "C: ");
                int s1 = Integer.parseInt(sT.nextToken());
                int s2 = Integer.parseInt(sT.nextToken());
                int s3 = Integer.parseInt(sT.nextToken());
                //aufsteigend sortieren s1, s2, s3
                int sTemp;
                if (s1 > s2) {
                    sTemp = s1;
                    s1 = s2;
                    s2 = sTemp;
                }
                if (s1 > s3) {
                    sTemp = s1;
                    s1 = s3;
                    s3 = sTemp;
                }
                if (s2 > s3) {
                    sTemp = s2;
                    s2 = s3;
                    s3 = sTemp;
                }
                list.add(new Cube(number, s1, s2, s3));
                number ++;
                line = aBufferedReader.readLine();
            }
            aFileReader.close();

        } catch (FileNotFoundException ex) {
            System.out.println("ERROR:");
            ex.printStackTrace();
        } catch (IOException ex) {
            System.out.println("ERROR:");
            ex.printStackTrace();
        } finally {
            return list.toArray(new Cube[1]); // siehe auch API-Dokumentation
        }

    }


    /**
     * mini Test
     * @param args
     */
    public static void main(String[] args)
    {
        SolidFileIO a = new SolidFileIO();
        Cube[] c = a.readSolids("soliddata.txt");
        for (Cube cube : c) {
            System.out.println(cube.toString());
        }

    }
}
```

```java

package oop6_dat6_u;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Arrays;
import java.util.Comparator;

/**
 * SolidWorks
 *
 * @author H. Diethelm, Hochschule Luzern
 * @version 6.4.2008
 */
public class SolidWorks extends JFrame implements ActionListener
{
    private Cube[] solids;
    private String fileName = "soliddata.txt";
    // GUI Elemente erzeugen
    private JButton bRead = new JButton("Read File '" + fileName + "'");
    private JButton bSort1 = new JButton("Sort 'volume'");
    private JButton bSort2 = new JButton("Sort 'surface'");
    private JButton bSort3 = new JButton("Sort 'dimension'");
    private JTextArea outputArea = new JTextArea();

    public SolidWorks()
    {
        super("SolidWorks");
        setSize(400, 400);
        setResizable(false);
        //center on screen
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container cp = getContentPane();
        JPanel p1 = new JPanel();

        p1.add(bRead);
        cp.add(p1, BorderLayout.NORTH);

        outputArea.setEditable(false);
        cp.add(new JScrollPane(outputArea), BorderLayout.CENTER);

        JPanel p2 = new JPanel();
        p2.add(bSort1); p2.add(bSort2); p2.add(bSort3);
        cp.add(p2, BorderLayout.SOUTH);

        setVisible(true);

        // Listener registrieren
        bRead.addActionListener(this);
        bSort1.addActionListener(this);
        bSort2.addActionListener(this);
        bSort3.addActionListener(this);
    }

    private void doOutput()
    {
        outputArea.setText("");
        for (int i = 0; i < solids.length; i++){
            outputArea.append(solids[i].toString());
        }
    }

    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == bRead){
            solids = SolidFileIO.readSolids(fileName);
            doOutput();
        }
        if ((e.getSource() == bSort1) && (solids != null)){
            // volume
            Arrays.sort(solids);
            doOutput();
        }
        if ((e.getSource() == bSort2) && (solids != null)){
            // surface
            Comparator<Cube> comparator = new SurfaceComparator();
            Arrays.sort(solids, comparator);
            doOutput();
        }
        if ((e.getSource() == bSort3) && (solids != null)){
            // dimension
            Comparator<Cube> comparator = new MaxDimensionComparator();
            Arrays.sort(solids, comparator);
            doOutput();
        }
    }

    public static void main(String[] args)
    {
        SolidWorks sw = new SolidWorks();
    }

}
```

```java
package oop6_dat6_u;

import java.util.Comparator;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class SurfaceComparator implements Comparator<Cube>
{

    public int compare(Cube c1, Cube c2)
    {
        return (c1.getSurface() - c2.getSurface());
    }


}
```