```java
package oop6_dat6_u;

import java.util.Comparator;

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

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


}
```

```java

package oop6_dat6_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */

public class Sphere implements Comparable<IMyObject>, IMyObject
{

    private int number; //Nummer der Kugel
    private int s1; //Radius

    public Sphere(int no, int a)
    {
        this.number = no;
        this.s1 = a;
    }

    /**
     * Berechnet das Volumen der Kugel
     * @return Volumen
     */
    public int getVolume()
    {
        return (4/3 * (int) Math.PI * s1 * s1 * s1);
    }

    /**
     * Berechnet die Oberfläche des Kubus
     * @return Oberfläche
     */
    public int getSurface()
    {
        return (4 * (int) Math.PI * s1 * s1);
    }

    /**
     * Liefert die grösste Seitenlänge als Ergebnis
     * @return grösse Seitenlänge
     */
    public int getMaxDimension()
    {
        return s1;
    }

    /**
     * String der Kugel
     * @return Beschreibung der Kugel
     */
    @Override
    public String toString()
    {
        /*
        return ("---------------------------------------\n"+
            "SPHERE " + number + "\n" +
            "Radius: " + s1 + "\n" +
            "Volume = " + this.getVolume() + "\n" +
            "Surface = " + this.getSurface()+ "\n");
         */
        return ("SPHERE " + number + " | " +
            "Radius: " + s1 + " | " +
            "Volume = " + this.getVolume() + " | " +
            "Surface = " + this.getSurface()+ "\n");
    }

    /**
     * Kubus vergleichen
     * @param other Kugel
     * @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() == ((Sphere) other).getVolume()))// 4. Vergleich relev.
Felder
        {
            return false;
        }
        return true;
    }

    /**
     * Hash-Code wird bassierend auf der (eindeutigen) Nummer definiert
     * @return Hash-Code
     */
    @Override
    public int hashCode()
```

```
101     {
102         return number;
103     }
104 
105     /**
106      * Kubus wird auf ihre Volumen verglichen
107      * @param c Cube zum Vergleichen
108      * @return Differenz des Volumen
109      */
110     public int compareTo(IMyObject c)
111     {
112         return (this.getVolume() - c.getVolume());
113     }
114 
115 
116     /**
117      * kleine "test" Methode
118      * @param args
119      */public static void main(String[] args)
120     {
121         Sphere s1 = new Sphere(1, 1);
122         Sphere s2 = new Sphere(2, 10);
123         Sphere s3 = new Sphere(3, 30);
124 
125         System.out.println(s1.toString());
126         System.out.println(s2.toString());
127         System.out.println(s3.toString());
128 
129         System.out.println("s1 == s2 ? : " + s1.equals(s2));
130         System.out.println("s2 == s3 ? : " + s2.equals(s3));
131         System.out.println("s1 >  s2 ? : " + s1.compareTo(s2));
132         System.out.println("s2 >  s3 ? : " + s2.compareTo(s3));
133     }
134 }
```

```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 IMyObject[] 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(450, 650);
        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<IMyObject> comparator = new SurfaceComparator();
            Arrays.sort(solids, comparator);
            doOutput();
        }
        if ((e.getSource() == bSort3) && (solids != null)){
            // dimension
            Comparator<IMyObject> 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.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 IMyObject[] readSolids(String fileName)
    {
        ArrayList<IMyObject> list = new ArrayList<IMyObject>();
        int s1, s2, s3, sTemp;

        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, ": ");
                String s0 = sT.nextToken();

                // Cube
                if (s0.equals("C")) {
                    s1 = Integer.parseInt(sT.nextToken());
                    s2 = Integer.parseInt(sT.nextToken());
                    s3 = Integer.parseInt(sT.nextToken());
                    //aufsteigend sortieren s1, s2, s3
                    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 ++;
                }

                // Kugel - Sphere
                if (s0.equals("S")) {
                    s1 = Integer.parseInt(sT.nextToken());
                    list.add(new Sphere(number, s1));
                    number ++;
                }

                // Zylinder - Cylinder
                if (s0.equals("Y")) {
                    s1 = Integer.parseInt(sT.nextToken());
                    s2 = Integer.parseInt(sT.nextToken());

                    //aufsteigend sortieren s1, s2
                    if (s1 > s2) {
                        sTemp = s1;
                        s1 = s2;
                        s2 = sTemp;
                    }
                    list.add(new Cylinder(number, s1, s2));
                    number ++;
                }

                line = aBufferedReader.readLine();
            }
            aFileReader.close();

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

    }

    /**
     * mini Test
```

```
102         * @param args
103         */
104        public static void main(String[] args)
105        {
106            SolidFileIO a = new SolidFileIO();
107            IMyObject[] c = a.readSolids("soliddata.txt");
108            for (IMyObject myObject : c) {
109                System.out.println(myObject.toString());
110            }
111
112        }
113    }
```

```
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<IMyObject> {
11 public int compare(IMyObject c1, IMyObject c2)
12     {
13         return (c1.getMaxDimension() - c2.getMaxDimension());
14     }
15 }
```

```java
package oop6_dat6_u;

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

    int getVolume();
    int getSurface();
    int getMaxDimension();
    String toString();
    boolean equals(Object other);
    int hashCode();
}
```

```java
package oop6_dat6_u;

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

    private int number; //Nummer des Zylinders
    private int s1; //Durchmesser
    private int s2; //Höhe

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

    /**
     * Berechnet das Volumen
     * @return Volumen
     */
    public int getVolume()
    {
        return (s1 * s1 * (int) Math.PI * s2);
    }

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

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

    /**
     * String der Kugel
     * @return Beschreibung der Kugel
     */
    @Override
    public String toString()
    {
        /*
        return ("---------------------------------------\n"+
        "CYLINDER " + number + "\n" +
        "Radius: " + s1 + "\n" +
        "Höhe: " + s2 + "\n" +
        "Volume = " + this.getVolume() + "\n" +
        "Surface = " + this.getSurface()+ "\n");
         */
        return ("CYLINDER " + number + " | " +
            "Radius: " + s1 + " | " +
            "Höhe: " + s2 + " | " +
            "Volume = " + this.getVolume() + " | " +
            "Surface = " + this.getSurface() + "\n");
    }

    /**
     * vergleichen
     * @param other
     * @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() == ((Cylinder) other).getVolume()))// 4. Vergleich relev.
Felder
        {
            return false;
        }
        return true;
    }

100
```

```
101     /**
102      * Hash-Code wird bassierend auf der (eindeutigen) Nummer definiert
103      * @return Hash-Code
104      */
105     @Override
106     public int hashCode()
107     {
108         return number;
109     }
110 
111     /**
112      * wird auf ihre Volumen verglichen
113      * @param c zum Vergleichen
114      * @return Differenz des Volumen
115      */
116     public int compareTo(IMyObject c)
117     {
118         return (this.getVolume() - c.getVolume());
119     }
120 
121     /**
122      * kleine "test" Methode
123      * @param args
124      */
125     public static void main(String[] args)
126     {
127         Cylinder y1 = new Cylinder(1, 10, 10);
128         Cylinder y2 = new Cylinder(2, 20, 10);
129         Cylinder y3 = new Cylinder(3, 30, 5);
130 
131         System.out.println(y1.toString());
132         System.out.println(y2.toString());
133         System.out.println(y3.toString());
134 
135         System.out.println("y1 == y2 ? : " + y1.equals(y2));
136         System.out.println("y2 == y3 ? : " + y2.equals(y3));
137         System.out.println("y1 >  y2 ? : " + y1.compareTo(y2));
138         System.out.println("y2 >  y3 ? : " + y2.compareTo(y3));
139     }
140 }
```

```java
package oop6_dat6_u;

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

    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");
         */
        return ("CUBE " + number + " | " +
            "Sides: " + s1 + " " + s2 + " " + s3 + " | " +
            "Volume = " + this.getVolume() + " | " +
            "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;
        }
        }
```

```
102         if ( ! (this.getVolume() == ((Cube) other).getVolume()))// 4. Vergleich relev.
Felder
103         {
104             return false;
105         }
106         return true;
107     }
108 
109     /**
110      * Hash-Code wird bassierend auf der (eindeutigen) Nummer definiert
111      * @return Hash-Code
112      */
113     @Override
114     public int hashCode()
115     {
116         return number;
117     }
118 
119     /**
120      * Kubus wird auf ihre Volumen verglichen
121      * @param c Cube zum Vergleichen
122      * @return Differenz des Volumen
123      */
124         public int compareTo(IMyObject c)
125     {
126         return (this.getVolume() - c.getVolume());
127     }
128 
129 
130     /**
131      * kleine "test" Methode
132      * @param args
133      */public static void main(String[] args)
134     {
135         Cube c1 = new Cube(1, 1, 2, 3);
136         Cube c2 = new Cube(2, 10, 20, 30);
137         Cube c3 = new Cube(3, 30, 20, 10);
138 
139         System.out.println(c1.toString());
140         System.out.println(c2.toString());
141         System.out.println(c3.toString());
142 
143         System.out.println("c1 == c2 ? : " + c1.equals(c2));
144         System.out.println("c2 == c3 ? : " + c2.equals(c3));
145         System.out.println("c1 > c2 ? : " + c1.compareTo(c2));
146         System.out.println("c2 > c3 ? : " + c2.compareTo(c3));
147     }
148 }
```