© Chris Hübsch, 1997

Hello World Applet:

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
  public void paint(Graphics g) {
    g.drawString("Hello world!", 50, 25);
  }
}

<HTML><HEAD><TITLE> A Simple Program </TITLE></HEAD>
<BODY>
Here is the output of my program:

<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25></APPLET>

</BODY></HTML>
```

Klassenbeispiel:

```
import java.awt.Point;
public class Rechteck [extends java.lang.Object] {

  private Point aufPunkt;         // (*)
  private int breite=0, höhe=0;  // Umlaute sind erlaubt

  Rechteck () {
    super ();                  // Parent-Constructor aufrufen
    aufPunkt=new Point(0,0);}  // auch gleich bei (*) möglich

  punlic Rechteck (int b, int h) {
    this();          // eigenen Constructor Rechteck() aufrufen
    breite=b; höhe=h;}

  public Rechteck (int x, int y, int breite, int höhe) {
    this(breite, höhe);  // die Parameter werden verwendet
    aufPunkt=new Point(x,y);}

  public void moveTo (Point neuerAufPunkt) {
    aufPunkt=neuerAufPunkt; // alter aufPunkt wird automatisch
gelöscht }

  public void moveTo (int x, int y) {  //überladene Methode
    aufPunkt=new Point(x,y);}

  public String toString () {  //überschriebene Methode
    return "Von "+aufPunkt.toString()+" Breite "+breite+" Höhe
"+höhe;}

  private normalisiere () { ... }

  public static int area (Rechteck r) {
    return r.breite*r.höhe;}
}
```

© Chris Hübsch, 1997

Interface-Verwendung:

```
interface Composeable {
     public void putElement(Object anObject);
     public void putElementAt(Object anObject, int anIndex);
     public boolean hasElements ();
     public Object getElementAt(int anIndex);
     public java.util.Enumeration elements();
   }

class Tabelle implements Composeable {
     public Tabelle (int Zeilen, int Spalten) { ... };
     public void putElement(Object anObject) { ... };
     public void putElementAt(Object anObject, int anIndex) {};
     public boolean hasElements () { ... };
     public Object getElementAt(int anIndex) { ... };
     public java.util.Enumeration elements() { ... };
     public String toString () { ... };
   }
```

GridBagLayout:

```
private void addItem(GridBagLayout gbl, Component c,int x, int y,
int b, int h, double dx,double dy) {
    GridBagConstraints gbc=new GridBagConstraints();
    gbc.fill=GridBagConstraints.BOTH;
    gbc.insets= new Insets(5,5,5,5);
    gbc.gridx=x; gbc.gridy=y;
    gbc.gridwidth=b; gbc.gridheight=h;
    gbc.weightx=dx; gbc.weighty=dy;
    gbl.setConstraints(c, gbc);
    add(c);
  }
```

Eine HandleEvent:

```
public boolean handleEvent(Event e) {
  if (e.id==Event.MOUSE_MOVE) {
    doMouseMove(); //Maus über Component bewegt  -- besser
mouseMove() überschreiben
    return true;
  }
  if (e.id==Event.ACTION_EVENT) {
    if (e.target == button1) {
      doButton1(); //button1 geklickt-besser action() überschreiben
      return true;
    }
    if (e.arg.equals("Exit")) {
      doExit(); //ein Objekt mit der Aufschrift Exit feuert Action-
Event
      return true;
    }
  }
  return super.handleEvent(e);
}
```