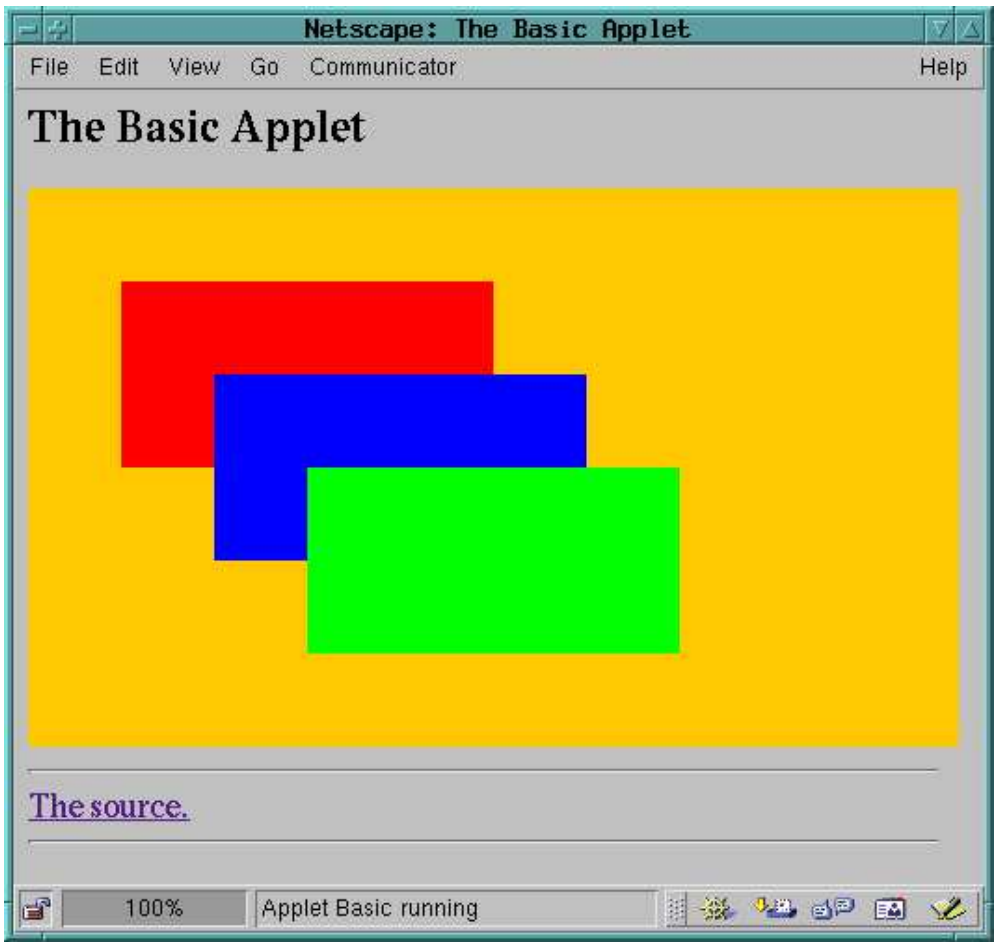
Achtung:

- Verfügbarkeit toller technischer Möglichkeiten bedeutet noch lange nicht, dass das Ergebnis beeindruckend ist ...
  ↑Kreativität
- Applets werden aus dem Internet gezogen:
  - Es gibt keine Garantie, dass sie nicht bösartig sind, d.h. versuchen, unerwünschte Effekte hervorzurufen ...
  - Applets werden darum innerhalb des Browsers ausgeführt und haben nur begrenzten Zugang zu System-Resourcen. ↑Sicherheit
- Statt im Browser können Applets auch mithilfe des Programms `appletviewer` betrachtet werden.

Ein Beispiel:



```
<html><head><title>The Basic Applet </title>
            <!-- Created by: Helmut Seidl -->
    </head>
    <body><h1>The Basic Applet</h1>
    <applet   codebase="."
              documentbase="pictures"
              code=Basic.class
              width=500
              height=300>
    Your browser is completely ignoring the &lt;APPLET&gt; tag!
    </applet>
    <hr>
    <a href="Basic.java">The source.</a>
    <hr>
    </body>
</html>
```

- HTML-Seiten bestehen aus (möglicherweise geschachtelten) Elementen, Kommentaren und Text.
- Ein Kommentar beginnt mit `<!--` und endet mit `-->`.
- Nicht-leere Elemente `b` beginnen mit einem Start-Tag `<b>` und enden mit einem End-Tag `</b>`.

Beispiele:

`html`	—	die gesamte Seite;
`head`	—	der Kopf der Seite;
`title`	—	der Titel der Seite;
`body`	—	der Rumpf der Seite;
`a`	—	ein Link;
`h1`	—	eine Überschrift;
`applet`	—	ein Applet.

- Bei leeren Elementen kann das End-Tag auch fehlen.

Beispiele:

`br` — Zeilen-Umbruch;

`hr` — horizontale Linie.

- Sonderzeichen beginnen mit `&` und enden mit `;` (Character references).

Beispiele:

`<` — “<”

`>` — “>”

`&auml;` — “ä”

- Start-Tags von Elementen können Attribute enthalten, die Zusatz-Informationen für die "Darstellung" des Elements enthalten.
- Der Wert eines Attributs kann (u.a.) ein String sein.
- Das Attribut `href` des Elements `a` gibt eine Internet-Adresse an:

```
<a href="Basic.java">The source.</a>
```

Einige Attribute des Elements `applet`:

- `codebase` — gibt das Verzeichnis an, in dem das Applet liegt;
- `documentbase` — gibt ein weiteres Verzeichnis an;
- `code` — gibt die Datei an, in der ausführbare Java-Code abgespeichert ist;
- `width` — gibt die Breite der verfügbaren Fläche an;
- `height` — gibt die Höhe der verfügbaren Fläche an.

... und jetzt das Applet selber:

```
import java.applet.Applet;
import java.awt.*;
public class Basic extends Applet {
   public void init() {
      showStatus("... no variables to initialize!");
   }
   public void start() {
      setBackground(Color.orange);
   }
   public void stop() {
      showStatus("... stopping the Basic Applet!");
   }
   public void destroy() {
      showStatus("... destroying the Basic Applet!");
   }
   ...
```

- Ein neues Applet erweitert die Klasse `java.applet.Applet`.
- Ein Applet braucht nicht über eine Klassen-Methode `main()` zu verfügen.
- Aufrufen des Applets durch das Element `applet` einer HTML-Seite führt die folgenden Aktionen aus:

  1. Auf der Seite wird dem Applet eine Fläche zur Verfügung gestellt, die entsprechend den Attribut-Werten `width` und `height` dimensioniert ist.
  2. Ein Objekt der Applet-Klasse (hier der Klasse: `Basic`) wird angelegt.
  3. Zur Initialisierung des Objekts wird die Objekt-Methode `init()` aufgerufen.
  4. Dann wird die Objekt-Methode `start()` aufgerufen.
  5. ...

```
    ...
    public void paint(Graphics g) {
        g.setPaintMode();
        g.setColor(Color.red);
        g.fillRect(50,50,200,100);
        g.setColor(Color.blue);
        g.fillRect(100,100,200,100);
        g.setColor(Color.green);
        g.fillRect(150,150,200,100);
        showStatus("... painting the Basic Applet!");
    }
} // end of Applet Basic
```

- Die Klasse `java.awt.Graphics` ist eine abstrakte Klasse, um Bilder zu erzeugen.
- Jede Implementierung auf einem konkreten System muss für diese Klasse eine konkrete Unterklasse bereitstellen ↑Portierbarkeit.
- Mit dem Applet verknüpft ist ein Objekt (der konkreten Unterklasse) der Klasse `Graphics`.
- Nach Ausführen der Methode `start()` wird das `Graphics`-Objekt `page` des Applets auf dem Fenster sichtbar gemacht.
- Auf dem sichtbaren `Graphics`-Objekt kann nun gemalt werden ...

Achtung:

- Wenn die Applet-Fläche verdeckt wurde und erneut sichtbar wird, muss die Applet-Fläche neu gemalt werden. Dazu verwaltet Java eine Ereignis-Schlange (event queue).
- Verändern der Sichtbarkeit der Fläche erzeugt ein `AWTEvent`-Objekt, das in die Ereignis-Schlange eingefügt wird.
- Das erste Ereignis ist (natürlich :-) die Beendigung der `start()`-Methode ...
- Das Applet fungiert als Consumer der Ereignisse.
- Es consumiert ein Ereignis "Fenster wurde sichtbar(er)", indem es den Aufruf `paint(page)` (für das aktuelle Applet) ausführt.

  Dadurch wird das Bild (hoffentlich) wiederhergestellt ...

Weitere Ereignisse:

- Modifizieren des Browser-Fensters $\Longrightarrow$ die `start()`-Methode wird erneut aufgerufen.
- Verlassen der HTML-Seite $\Longrightarrow$ die `stop()`-Methode wird aufgerufen.
- Verlassen des Browsers $\Longrightarrow$ die `destroy()`-Methode wird aufgerufen.

Nützliche Objekt-Methoden der Klasse `applet`:

- `public void showStatus(String status)` schreibt den String `status` in die Status-Zeile des Browsers.
- `public void setBackground(Color color)` setzt die Hintergrundsfarbe.
- `public Color getBackground()` liefert die aktuelle Hintergrundsfarbe.
- `public void setVisible(boolean b)` macht das `Graphics`-Objekt sichtbar bzw. unsichtbar.
- `public boolean isVisible()` teilt mit, ob Sichtbarkeit vorliegt.

21.1 Malen mit der Klasse `Graphics`

- Die Klasse `Graphics` stellt eine Reihe Methoden zur Verfügung, um einfache graphische Elemente zu zeichnen, z.B.:
  - `public void drawLine(int xsrc, int ysrc, int xdst, int ydest);` — zeichnet eine Linie von `(xsrc, ysrc)` nach `(xdst, ydst);`
  - `public void drawRect(int x, int y, int width, int height);` — zeichnet ein Rechteck mit linker oberer Ecke `(x,y)` und gegebener Breite und Höhe;
  - `public void drawPolygon(int[] x, int[] y, int n);` — zeichnet ein Polygon, wobei die Eckpunkte gegeben sind durch `(x[0],y[0])`,..., `(x[n-1],y[n-1])`.

- Diese Operationen zeichnen nur die Umrisse.
- Soll auch das Innere gezeichnet werden, muss man statt `drawXX(...);` die entsprechende Methode `fillXX(...);` benutzen.

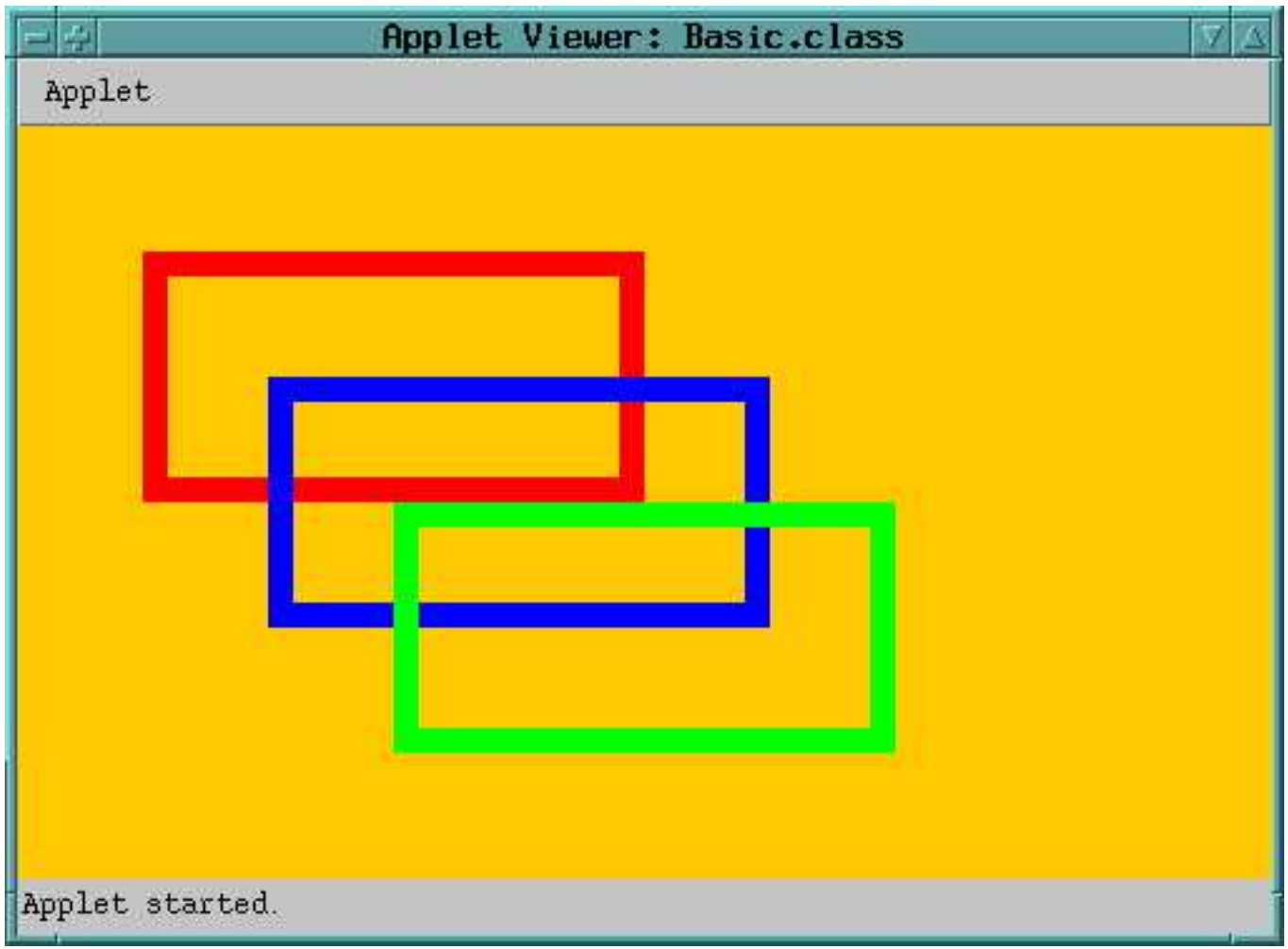
Achtung:

- Die gemalten Elemente werden sequentiell vor dem Hintergrund des Applets abgelegt.
- Wurde die `Graphics`-Methode `public void setPaintMode();` aufgerufen, überdeckt das spätere Element das frühere.
- Die Farbe, in der aktuell gemalt wird, muss mit der `Graphics`-Objekt-Methode `public void setColor(Color color):` gesetzt werden.

Beispiel: Rechteck mit Rand

```
import java.awt.*;
public class AuxGraphics {
   private Graphics page;
   public AuxGraphics (Graphics g) { page = g;}
   public void drawRect(int x, int y,
                        int w, int h, int border) {
      page.fillRect(x,y,border,h);
      page.fillRect(x+border,y,w-2*border,border);
      page.fillRect(x+border,y+h-border,w-2*border,border);
      page.fillRect(x+w-border,y,border,h);
   }
}
```

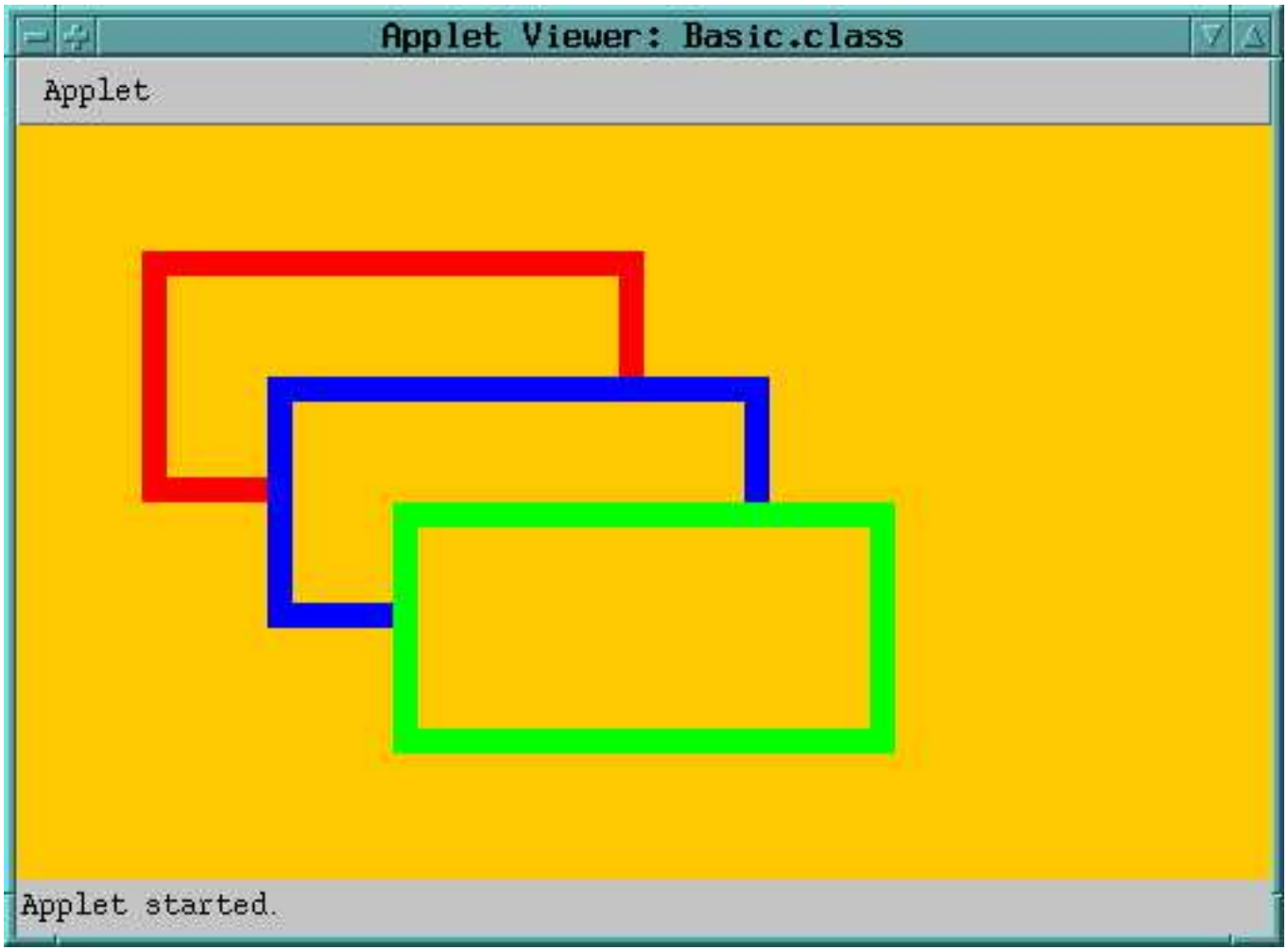
Der Rand wird aus vier kleineren Rechtecken zusammen gesetzt ...


Applet Viewer: Basic.class
Applet
Applet started.

- Man könnte auch auf die Idee kommen, das Innere des Rechtecks durch ein Rechteck in Hintergrunds-Farbe zu übermalen:

```
public void drawRect(int x, int y, int w, int h, int border) {
   Color col = page.getColor();
   page.fillRect(x, y, w, h);
   page.setColor(app.getBackground());
   page.fillRect(x+border,y+border,w-2*border,h-2*border);
   page.setColor(col);
}
```

... mit dem Effekt:


Applet Viewer: Basic.class
Applet
Applet started.

- Seit der Java-Version 1.2 ist das `Graphics`-Objekt, auf dem das Applet malt, aus der Unterklasse `Graphics2D`.
- Diese Klasse bietet tolle weitere Mal-Funktionen :-).
- Insbesondere macht sie den beim Malen verwendeten Strich (Stroke) der Programmiererin zugänglich.
- Statt dem Konzept "Farbe" gibt es nun das Interface `Paint`, das es z.B. gestattet, auch farbliche Abstufungen und Texturen zu malen :-))