Inhaltsverzeichnis

Doug Engelbart entwickelte in seinem Projekt „Augmentation of Human Intellect“ am Stanford Research Institute (SRI) in den sechziger Jahren das so genannte „On-Line-System“ (NLS). Es enthielt einen von einer Maus gesteuerten Cursor und mehrere Fenster. Engelbart war zum Teil durch die „Memex-Informationsmaschine“ inspiriert worden, ein Tischgerät, das Vannevar Bush 1945 vorgestellt hatte. Im Jahre 1968 wurde von ihm die "Mutter aller Demos" vorgeführt. Über zwei verbundene CDC3100 konnte erstmalig ein Chat mit Video vorgeführt werden. Zusätzlich war eine Copy And Paste Funktion mittels einer graphischen Benutzeroberfläche möglich.

Command buttons from Windows XP, Windows 2000, and OS X

Rechenmaschinen


Facit NTK (1954)


Triumphator CRN1 (1958)


Walther WSR160 (1960)


Olivetti Divisumma 24 (1964)

http://en.wikipedia.org/wiki/Calculator

"Für jede rein gleichförmig sich wiederholende Bewegung, welche keine geistige Tätigkeit erfordert, wird mit der Zeit eine Maschine erfunden; dem Menschen bleibt mehr und mehr die rein geistig leitende und künstlerische Tätigkeit."
Gustav Schmoller, Die Arbeiterfrage

3 Grafiken, Threads und GUIs

Dieses Kapitel ist ein Exkurskapitel. Es zeigt, wie man mit Hilfe von Java GUIs (*Graphical User Interface*, siehe Grafische Benutzeroberfläche (grafische Benutzerschnittstelle) programmieren kann.
Wir beschränken uns zunächst darauf, Applets zu programmieren, die zwar Grafik und Userkommunikation ermöglichen, aber z.B. einen Dateizugriff nicht erlauben.

3.1 Was sind Applets – Theoretischer Hintergrund

Ein **Java-Applet** ist ein kleines Computerprogramm, das in der Programmiersprache Java verfasst wurde.
Java-Applets wurden eingeführt, um Programme in Web-Seiten ablaufen lassen zu können, die im Webbrowser (auf der Client-Seite) arbeiten und direkt mit dem Benutzer interagieren können, ohne Daten über die Leitung zum Server versenden zu müssen.
Java-Applets waren Ende der 1990er Jahre ein Hauptgrund für den Erfolg und die schnelle Verbreitung von Java. (...)
Applets werden auf dem Rechner des Anwenders ausgeführt und stellen daher - wie jedes lokal ausführbare Programm - ein Sicherheitsrisiko dar. Da die Applets in einer abgeschotteten Laufzeitumgebung (Sandbox) laufen, ist dieses Risiko jedoch gut kontrollierbar. Ein Sicherheitsrisiko durch "böswillige" Applets besteht nur, wenn die Sandbox fehlerhaft ist.
http://de.wikipedia.org/wiki/Java-Applet , 03.05.2006

3.1.1 Kennzeichen von Applets

Sie verfügen unter anderen über die folgenden Methoden,

- `init()` - wird genau einmal aufgerufen, wenn das Applet erstmals in den Browser geladen wird.
- `start()` - wird jedesmal aufgerufen, wenn das Applet sichtbar wird.
- `paint(...)` - Zeichenmethode für die Anzeigefunktionen des Applet
- `stop()` - wird jedesmal aufgerufen, wenn das Applet verdeckt wird, z.B. weil das Browser-Fenster von einem anderen Fenster überdeckt wird.
- `destroy()` - wird aufgerufen, wenn das Applet aus dem Hauptspeicher entladen wird.

http://de.wikipedia.org/wiki/Java-Applet , 03.05.2006

3.1.2 Nicht abiturrelevant – nicht klausurrelevant

Dieses Kapitel ist weder abitur- noch klausurrelevant. Aber Programmieren macht erst dann richtig Spaß, wenn die Ergebnisse auch vorzeigbar sind. Das folgende Kapitel bedient diesen Mangel. Der Bankautomat sieht aus wie ein Bankautomat und die Uhr wie eine Uhr.
Leider ist das mit einer Menge Technik verbunden, auf die im Einzelnen nicht eingegangen werden kann, dazu fehlt leider die Zeit. Wer am Programmieren Gefallen gefunden hat, kann jedoch problemlos im Internet Ergänzungen finden.

3.2 paint(Graphics g) - Grafiken und Texte mit Applets

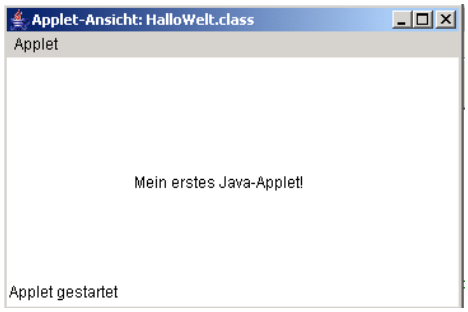
Ein einfaches Beispiel **HalloWelt.java** (hier wird nur *paint()* überschrieben um gezielt ab der Stelle 100,100 den Text ausgeben zu können):

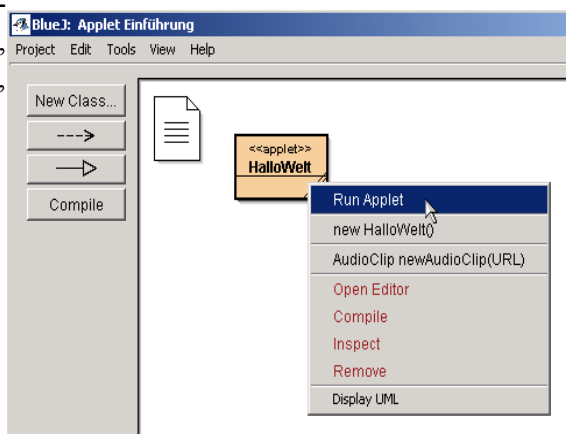
```
import java.awt.*;
import java.applet.Applet;
public class HalloWelt extends Applet
```

```
{
  public void paint( Graphics g )
  {
      g.drawString( "Mein erstes Java-Applet!",100,100 );     // Grafik-Text
  }
}
```

aus: http://de.wikibooks.org/wiki/Java_Standard:_Applets

BlueJ umgeht nettwerweise wieder einige Anfangshürden. Wenn man die oben gezeigte Klasse anlegt, kann man mit „Run Applet“ das Programm starten, ohne selbst eine Internetseite anlegen zu müssen.





Es läuft im Applet-Viewer, kann aber auch in eine Internetseite eingefügt werden. Siehe dazu http://de.wikibooks.org/wiki/Java_Standard:_Applets

Erläuterung:
Auf den Quelltext gehe ich hier nicht zu tief ein. Was die einzelnen Zeilen bedeuten, kann man in jeder Java-Dokumentation nachlesen.
Die Methode paint ermöglicht es uns, grafische Ausgaben zu gestalten. Legt man es in der Weise an, lassen sich allerlei grafische Ausgaben realisieren.

Übung:	
1)	Testen Sie das Applet. Ändern Sie Text und Position der Ausgabe. Fügen Sie eine zweite Ausgabe hinzu.

3.2.1 Einige Methoden der Klasse Graphics

Die Schnittstelle um graphische Objekte wie Shapes, oder aber auch Bilder zu zeichnen, ist in Java das Graphics-Objekt.

Ausgabe eines Strings:

```
g.drawString( "Mein erstes Java-Applet!",100,100 );
```

Ausgabe eines Strings mit anderem Zeichensatz und Farbe

```
g.setFont(new Font("Dialog", Font.PLAIN, 18));
g.setColor(Color.BLUE);
g.drawString("Hallo Welt", 10,40);
```

Ausgabe eines ausgefüllten abgerundeten Rechtecks
g.fillRoundRect(startX,startY,breite,hoeher,radiusAbrundungX,radiusAbrundungY);
Zum Beispiel:
g.fillRoundRect(50,70,20,30,10,20);

aus: http://de.wikibooks.org/wiki/Java_Standard:_Das_Graphics_Object

Übung:	
1)	Experimenieren Sie mit den Ausgabemöglichkeiten des Strings. Was passiert, wenn Sie einen riesigen, was, wenn Sie einen winzigen Buchstaben zeichnen lassen?

3.2.2 Java-Dokumentation – Linien, Rechtecke, Kreise und Farben

In der offiziellen Java-Dokumentation finden Sie alle Methoden der Klasse Graphics hier:
http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Graphics.html

Für den Anfang sinnvoll erscheinen Linienm Rechtecke und Ovale.

drawLine(int x1,int y1,int x2,int y2)
Zieht eine Linie von (x1, y1) nach (x2, y2). (0|0) ist dabei oben links.

drawRect(int x,int y,int width,int height)
Zeichnet ein Rechteck, obere linke Ecke bei (x|y) mit Breite width und höhe height.
Es gibt auch ausgefüllte Rechtecke (**fillRect**) und abgerundete Rechtecke (vgl. oben)

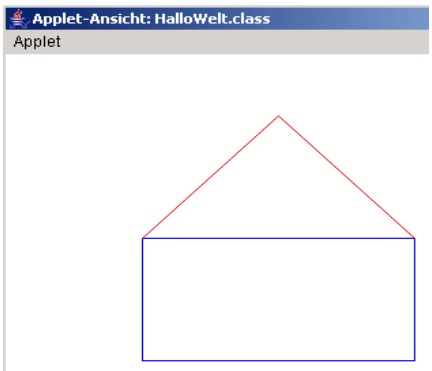
drawOval(int x,int y,int width,int height)
Zeichnet ein Oval, der ein Kreis ist, falls with und height gleich groß sind.

setColor(Color c)
ändert nicht nur bei Stringausgaben die Farbe, sondern auch bei allen anderen Figuren, die ausgegeben werden.
Einige Farben sind vordefiniert: z.B. g.setColor(Color.YELLOW);
Es können aber auch RGB-Werte die Farbe bestimmen.
Siehe dazu: http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Color.html

Übung:	
1)	Welche Methoden vermissen Sie? Schlagen Sie nach, ob es sie gibt.

3.2.3 Ein Haus

```
import java.awt.*;

import java.applet.Applet;
public class HalloWelt extends Applet
{
 public void paint( Graphics g )
 {
    g.setColor(Color.BLUE);
    g.drawRect(100,150,200,100);
    g.setColor(Color.RED);
    g.drawLine(100,150,200,50);
    g.drawLine(300,150,200,50);
 }
}
```

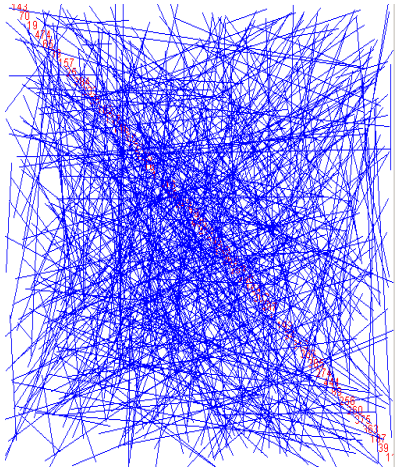


Übung:	
1)	Setzen Sie dem Haus eine Schornstein und ergänzen Sie ein Fenster und eine Sonne.
2)	Zeichnen Sie als Übung zur Grafik ein beliebiges Bild.

3.2.4 Zufallsfigur

Folgende Befehlskombination schafft eine Zufallszahl zwischen 0 und 499:
(int) (Math.random()*500)
Das ist so zu lesen: Math.random ist eine double Zahl zwischen 0 und 1 (z.B. 0.124398234..), die wird mit 500 multipliziert und zu einer ganzen Zahl (durch (int)) gemacht.

```
import java.awt.*;
 import java.applet.Applet;
 public class Zufallsfigur extends Applet
 {
  public void paint( Graphics g )
  {   int i=0;
      g.setColor(Color.BLUE);
      while(i<1000)
      {   i++;
          g.setColor(Color.RED);
          g.drawString(""+(int) (Math.random()*500), i*10,i*10);
          g.setColor(Color.BLUE);
          g.drawLine((int) (Math.random()*500),
                  (int) (Math.random()*500),
                  (int) (Math.random()*500),
                  (int) (Math.random()*500));
      }
   }
 }
```



Übung:	
1)	Mathematische Aufgabe: Warum liegen in der Mitte des Bildes mehr blaue Halme als außen?
2)	Entwerfen Sie ein Zufallsbild mit Kreisen.

3.3 Nebenläufige Prozesse - Threads

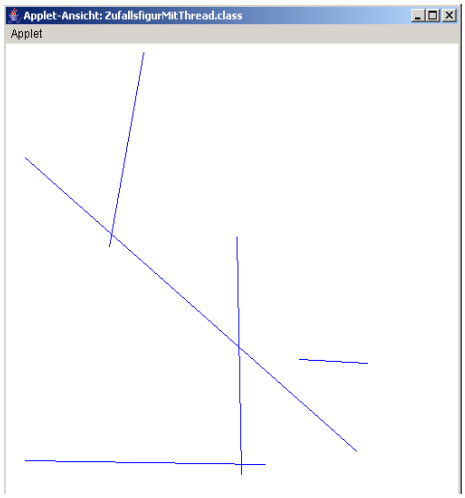
„Nebenläufigkeit (concurrency) ist die Fähigkeit eines Systems zwei oder auch mehrere Aufgaben (scheinbar) gleichzeitig auszuführen. In Java kann die Ausführungsparallelität innerhalb eines Programmes mittels Threads (lightweight processes) erzwungen werden.“
http://de.wikibooks.org/wiki/Java_Standard:_Threads

Wieder kann hier nicht in aller Tiefe auf die Vorgänge eingegangen werden. Vielleicht hilft es etwas, die roten Hervorhebungen und Kommentierungen zu lesen: das ist neu wegen des Threads.
Ich empfehle Ihnen, diesen oder ähnliche Quelltexte als Grundlage für alle Eigenentwicklungen mit

Threads zu nehmen. Es lohnt sich nicht, alles auswendig zu lernen.

Das folgende Applet zeichnet alle halbe Sekunde 5 Linien auf den Bildschirm.

```
import java.awt.*;
import java.applet.Applet;
public class ZufallsfigurMitThread extends Applet implements Runnable
{
  Thread thread;

  public void init()
  {
     thread= new Thread(this); // thread ist ein neuer Thread dieses Applets
     thread.start(); // starte den Thread
  }

  public void destroy()
  {
     thread = null; // Beim Verlassen des Applets setze den Thread auf null
  }

  public void run()
   {
      while (true) // solange true, also immer
      {
         repaint(); // rufe paint auf
         try // und versuche, 500 Millisekunden zu warten
         {
            thread.sleep(500);
         }
         catch (InterruptedException e) // wenn das mit dem Warten nicht geklappt hat,
         {
            System.out.println(e); // dann gib einen Fehler aus.
         }
      }

   }

  public void paint( Graphics g )
  {  int i=0;
    g.setColor(Color.BLUE);
    while(i<5)
    {  i++;
       g.setColor(Color.BLUE);
       g.drawLine((int) (Math.random()*500),
                  (int) (Math.random()*500),
                  (int) (Math.random()*500),
                  (int) (Math.random()*500));
    }
  }
```



```
}
```

vgl. auch
http://www.realapplets.com/tutorial/ThreadExample.html

Übung:	
1)	Ändern Sie die Geschwindigkeit, die Farbe und die Anzahl der Linien.
2)	Entwerfen Sie eine eigene Zufallsgrafikenanimation.

3.4 Der nächste Streich: Beschriftung, Button und ActionListener

Kaum ein modernes Programm kommt heutzutage ohne eine ansprechende GUI aus. Daher hier nun ein Beispiel, wie man mit Beschriftungen und Buttons in Java umgeht.
Hervorgehoben ist alles, was in Verbindung mit dem sogenannten ActionListener in Verbindung steht. Es handelt sich um eine Technik, mit denen man GUI-Elemente wir Buttons mit Aufgaben versehen kann.
Grobe Erklärung der Funktionsweise: Ein neue Button wird angelegt. Dieser bekommt einen ActionListener zugewiesen. Der sorgt dafür, dass actionPerformed aufgerufen wird, wenn der Button gedrückt ist. Da man manchmal mehrere Elemente hat, prüft man mit if (evt.getSource() == button), ob das ActionEvent „evt“ auch wirklich vom Button button stammt.

Vertiefend:
http://de.wikibooks.org/wiki/Java_Standard:_Grafische_Oberfl%C3%A4chen_mit_AWT
http://www.realapplets.com/tutorial/ActionExample.html

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class GUI_1 extends Applet implements ActionListener
{
  Label label;
  Button button;
  int anzahl;

  public void init()
  {
    button = new Button("Schaltfläche");
    button.addActionListener(this);
    add (button);

    label = new Label("Du hast kein mal gedrückt");
    add(label);
  }


   public void actionPerformed(ActionEvent evt)
   {
      if (evt.getSource() == button) {
```

```
                anzahl++;
                label.setText("Du hast "+anzahl+" mal gedrückt");
                repaint();
            }
        }

}
```

Übung:	
1)	Legen Sie ein zweites Label an, was anzeigt, wie oft man noch drücken muss, bis 100 erreicht ist.
2)	Legen Sie einen zweiten Button an, der herunterzählen lässt.

3.5 Textfelder

Die folgende Klasse verwendet neben Buttons und Labels auch Textfelder. In diesen können Benutzer Eingaben tätigen, die mit getText() abgefragt werden können. setText(...) ändert den Text.

Vertiefend:
http://de.wikibooks.org/wiki/Java_Standard:_Grafische_Oberfl%C3%A4chen_mit_AWT#Texteingabefelder

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class GUI_2 extends Applet implements ActionListener
{
  Label label;
  Button button;
  TextField textfield;
  int anzahl;

  public void init()
  {
    textfield = new TextField("niemand");
    add (textfield);
    button = new Button("Begrüße");
```

```
        button.addActionListener(this);
        add (button);

        label = new Label("Ich kenne dich nicht!");
        add(label);
    }


     public void actionPerformed(ActionEvent evt)
     {
         if (evt.getSource() == button) {
             label.setText("Hallo "+textfield.getText());
             repaint();
         }
     }

}
```

3.5.1 Mehrzeilige Texteingabefelder

Mehrzeilige Texteingabefelder können Sie mit der Klasse `TextArea` erstellen. Zusätzlich zu dem Anfangstext gibt man beim Anlegen noch die Zeilenanzahl und die Breite in Buchstaben an. Hinweis: \n wechselt im TextArea die Zeile, führt aber z.B. in einzeiligen Labels zu Problemen!

```
textarea = new TextArea("Hallo Welt!\nWo bist du?", 5,30));
```

Übung:	
1)	Ändern Sie im vorherigen Beispiel oben die Texteingabefeldern (TextField) zu mehrzeiligen TextAreas.

3.5.2 Weitere GUI-Elemente

Wie Menus, Radiobuttons, Listen etc. sowie Tipps zum Layout finden Sie hier:
http://de.wikibooks.org/wiki/Java_Standard:_Grafische_Oberfl%C3%A4chen_mit_AWT

3.6 Version

- 0.1 – 04.05.2006 Angelegt bis GUI

Wikipedia
aus: Quelle

Tabelle Gelb

Übung:	
1)	
2)	
3)	

5