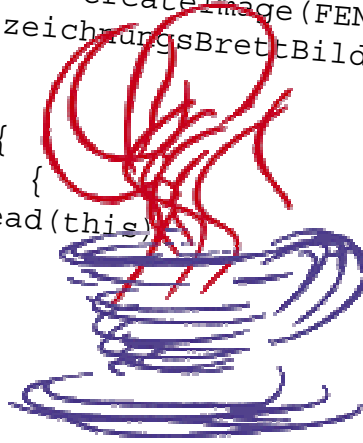


Java

```
public Image zeichnungsBrettBild = null;
private Thread thread;
private boolean isRunning = false;

public void init() {
    thread = null;
    zeichnungsBrettBild = createImage(FENSTERMAX_I, FENSTERMAX_
    zeichnungsBrett = zeichnungsBrettBild.getGraphics();
}

public void start() {
    if (thread == null) {
        thread = new Thread(this)
    }
    isRunning = true;
    thread.start();
}
```



JavaKara -> Java

Ablauf in JavaKara:

1. Programm schreiben
2. Kompilieren
3. Programm starten



Ablauf in Java gleich

1. Programm schreiben
2. Kompilieren
3. Programm starten

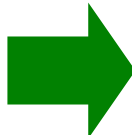
1. Programm schreiben

Texteditor statt JavaKara-Editor

```
import ch.edareth.k2.javakara.JavaAppProgram;
import ch.edareth.k2.javakara.FindenApp;

public class Finden extends FindenAppProgram {
    // hier können Sie eigene Methoden definieren

    protected void appProgramm() {
        // hier kommt das Hauptprogramm hin, zB:
        while (!Kack.isDone()) {
            kack.nimm();
        }
    }
}
```



```
public class FindenApp {
    private Thread thread;
    private boolean isRunning = false;

    public void init() {
        // Wird zu Beginn (vom Appletviewer) aufgerufen, wenn das A
        //
        thread = null;
        zeichnungsbrettBild = createImage(FENSTERMAX_X, FENSTERMAX_Y,
            zeichnungsbrett, zeichnungsbrettBild.getGraphics());
    }

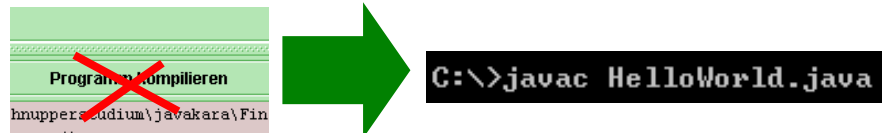
    public void start() {
        /* Erzeugt ein neuen Thread, der nur für unser HauptProgram
        * Der "alte" Thread bleibt auch aktiv, er sorgt dafür, das
        * wenn nötig neu dargestellt wird
        */
        if (thread == null) {
            thread = new Thread(this);
            isRunning = true;
            thread.start();
        }
    }

    public void run() {
        /* Diese Methode wird durch thread.start() ausgelöst. Sie r
        * Methode Hauptprogramm(); auf und beendet den Thread aus
        */
        Hauptprogramm();
        isRunning = false;
    }

    public void stop() {
        /* Wenn der Appletviewer beendet wird, so soll auch der Th
        * Neuzustand zurückgesetzt werden.
        */
        if (thread != null && thread.isAlive()) {
            thread.interrupt();
            thread = null;
        }
    }
}
```

2. Kompilieren

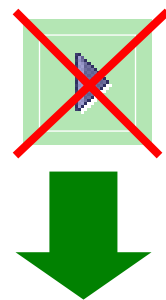
**Befehl in MS-DOS-Eingabeaufforderung
anstatt bequemer Knopf**



javac steht für „**java compiler**“

3. Programm starten

**Befehl in MS-DOS-Eingabeaufforderung
anstatt bequemer Knopf**



```
C:\>appletviewer HelloWorld.html
```

TextPad

Wir verwenden hier TextPad
(www.textpad.com)



- **Vorteil 1:**
Farben erleichtern das Lesen der Programmzeilen
- **Vorteil 2:**
Kompilieren und Start des Applets mit je einem Menu-Befehl

```
int as = a * 60;  
int hs = h * 60 * 60;  
// Clear the background.  
g.setColor(bgcolor);  
g.fillRect(start.x, start.y, size);  
// Draw the clock circle.  
g.setColor(ccolor);  
g.fillOval(start.x, start.y, dian);  
// Draw the clock numbers.
```

Ausführen...

Java kompilieren	Strg+1
Java-Programm starten	Strg+2
Java-Applet starten	Strg+3

TextPad

- **Wir brauchen nur wichtigste Funktionen des TextPad:**
 - Datei öffnen
 - Datei speichern
 - Rückgängig-Befehl
 - Java kompilieren
 - Java-Applet starten
- **Siehe TextPad-Anleitung in den Unterlagen**

Warum HelloWorld.html ?

- **2 Arten Java Programme**
 - Java Applikationen
 - Java Applets
 - Kann/muss man in Webseiten einbauen
 - Einfachere Programme als bei Applikationen
- **Applets: Man benötigt Webseite (.html) und ein Betrachtungsprogramm**

```
C:\>appletviewer HelloWorld.html
```

Wie sieht die .html-Datei aus?

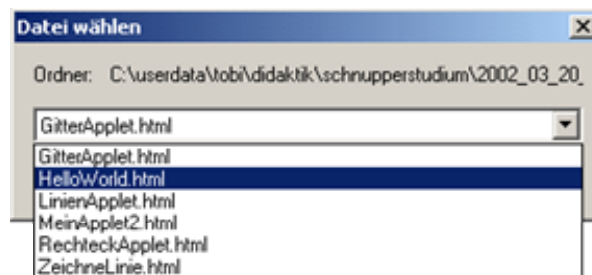
```
HelloWorld.html - Editor
Datei Bearbeiten Format ?
<BODY>
<APPLET code="Helloworld.class" width=400 height=100>
</APPLET>
</BODY>
```

Dateiname der
kompilierten Java-Klasse

Grösse des Programm-
Fensters

TextPad-Problem

- **Um Java-Applet aus dem TextPad zu starten, muss die .html-Datei bereits existieren**



Also nochmals...

1. **Programm schreiben -> HelloWorld.java**
2. **Kompilieren -> HelloWorld.class
(javac HelloWorld.java)**
3. **HTML-Datei (z.B. HelloWorld.html)
erstellen und Programm starten
(appletviewer HelloWorld.html)**

Ein einfaches Java-Applet

```
import java.awt.*;
import java.applet.*;
public class HelloWorld extends Applet
{
    public void paint(Graphics screen)
    {
        screen.drawString("hello world", 10, 10);
    }
}
```

Wichtig bei Java-Programmen



Goldene Regel

- **Dateiname = Klassenname**
HelloWorld.java <-> (...) class HelloWorld (...)
- **Kompilieren nicht vergessen!**
(Sonst startet man eine alte „HelloWorld.class“-Version)

Das Spezielle an Java

- **Java stellt viele Funktionen zur Verfügung**
- **Durch diese „Fertig-Bauelemente“ ist man schneller am Ziel**
- **Aber: Man muss sich an die Richtlinien halten**



public void paint(...)

- **Wird von Java aufgerufen, sobald ein Bildschirmbereich neu gezeichnet werden muss**
- **Von Java festgelegter Name**
- **Ideal, um Grafik-Sachen zu zeichnen.**



Ablauf bei Java-Programmen

- **In Java programmiert man Ereignisse**
- **Ereignis: „Führe bei Mausklicks die Methode xy aus“**
- **Nicht Ereignis-gesteuert:
Viel komplizierter!**



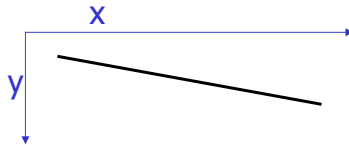
Ablauf bei Java

Von Java definierte Methoden

- **Beim Start der Ausführung:**
`void init()`
- **Immer, wenn es etwas zu zeichnen gibt:**
`void paint(Graphics g)`
- **Vor dem Programmende:**
`void destroy()`
- **Und noch weitere wie**
`void start(), void stop(), void update()`

Linie zeichnen

```
import java.awt.*;
import java.applet.*;
public class ZeichneLinie extends Applet
{
    public void paint(Graphics screen)
    {
        screen.drawLine(10,10,200,50);
    }
}
```



Graphics...

```
import java.awt.*;
import java.applet.*;
public class ZeichneLinie extends Applet
{
    public void paint(Graphics screen)
    {
        screen.drawLine(10,10,200,50);
    }
}
```

Graphics...

An Programm:
starte dein
paint(...) !



Java

Okay. Aber wo
soll ich denn
hinzeichnen?



Programm

Graphics...

Hier kriegst du
ein Blatt
Papier vom
Typ Graphics



Java

Danke. Ich
nenne das Blatt
"screen"



Programm

Graphics

Zusammengefasst

- Der Methode `paint(...)` wird ein Objekt vom Typ `Graphics` übergeben:

```
public void paint (Graphics screen)
{
    screen.drawLine(10,10,200,50);
}
```

- Dem Objekt geben wir einen Namen
- Objekte vom Typ `Graphics` stellen viele Methoden bereit, z.B. `drawLine(...)`

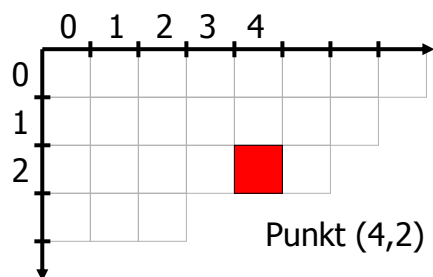
Graphics-Befehle

```
Graphics screen;
```

- **Text ausgeben:**
`screen.drawString("Java ist cool", 10, 20);`
- **Linie zeichnen:**
`screen.drawLine(5, 10, 250, 40);`
- **Punkt zeichnen:**
`screen.drawLine(10, 70, 10, 70);`

Hinweise zu Grafik

- **Koordinaten beginnen bei (0,0)**
- **Fenster der Grösse 400x200**
 - Punkt oben-links: (0,0)
 - Punkt unten-rechts: (399,199)



Graphics-Blatt "weitergeben"

- **Methode `zeichneRechteck(...)` soll man mit `screen.drawLine(x1, y1, x2, y2);` eine Linie zeichnen können**
- **Wir geben das Blatt an die Methode weiter**

```
public void paint (Graphics blatt)
{
    zeichneRechteck(10, 10, 200, 50, blatt);
}

void zeichneRechteck ( ... , Graphics screen)
{
    screen.drawLine(x1, y1, x2, y2);
}
```

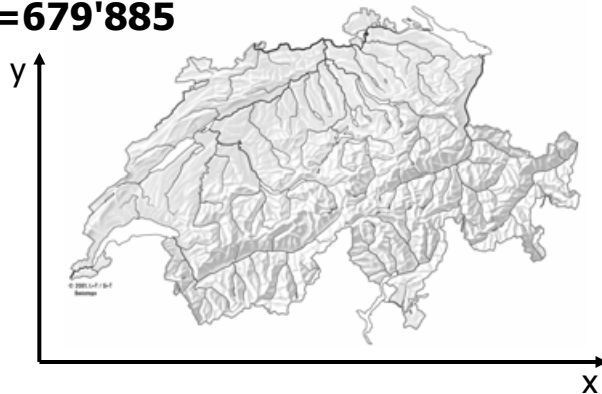
Schweizerkarte

- Schweizerkarte auf dem Bildschirm anzeigen
- Stadt Bern und Stadt Zürich einzeichnen

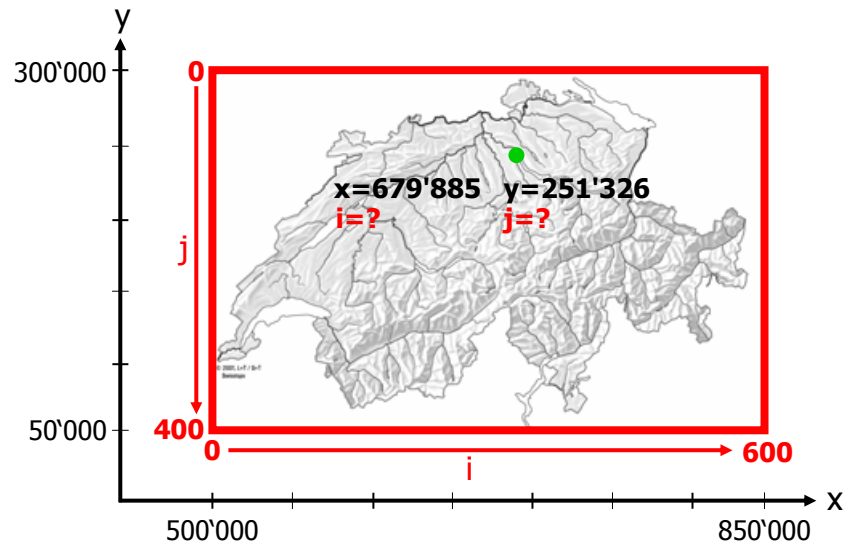


Wo einzeichnen?

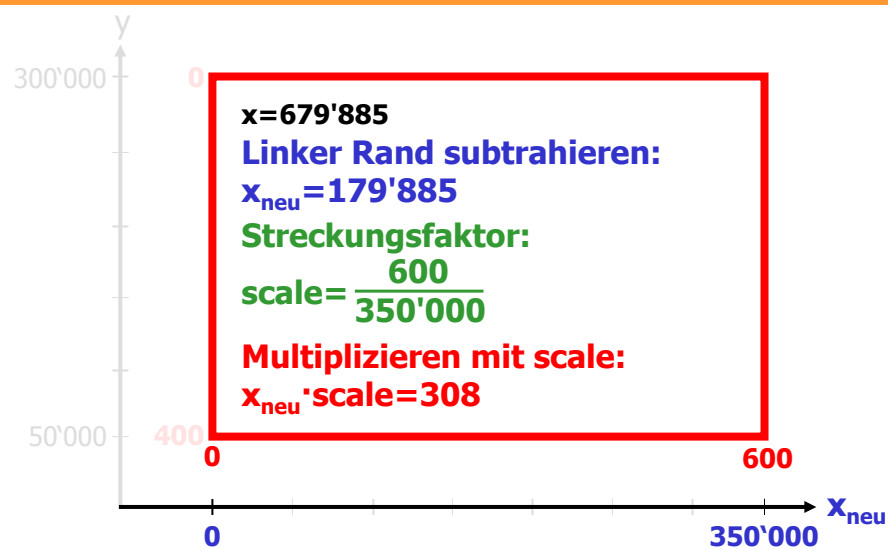
- Koordinaten von Bern:
 $y=200'000$, $x=600'000$
- ETH Zürich, Höggerberg:
 $y=251'326$, $x=679'885$



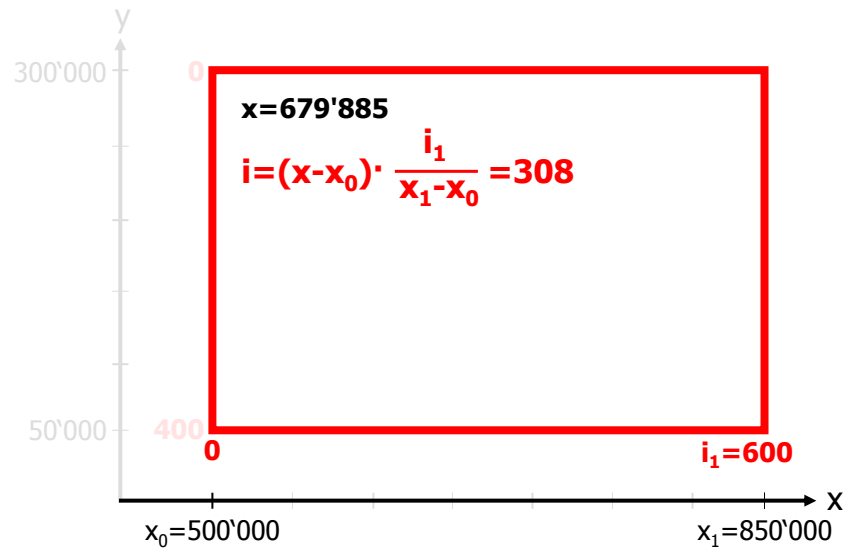
Wo ist Zürich?



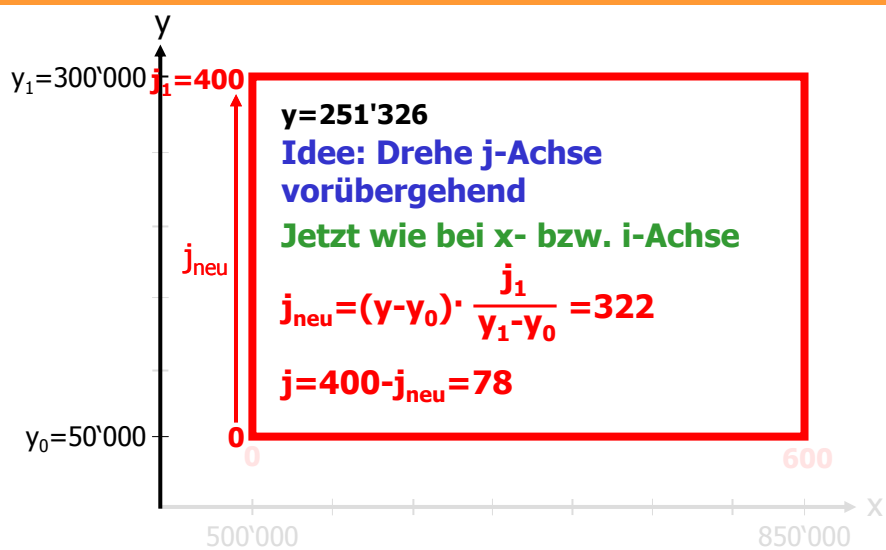
Für x bzw. i



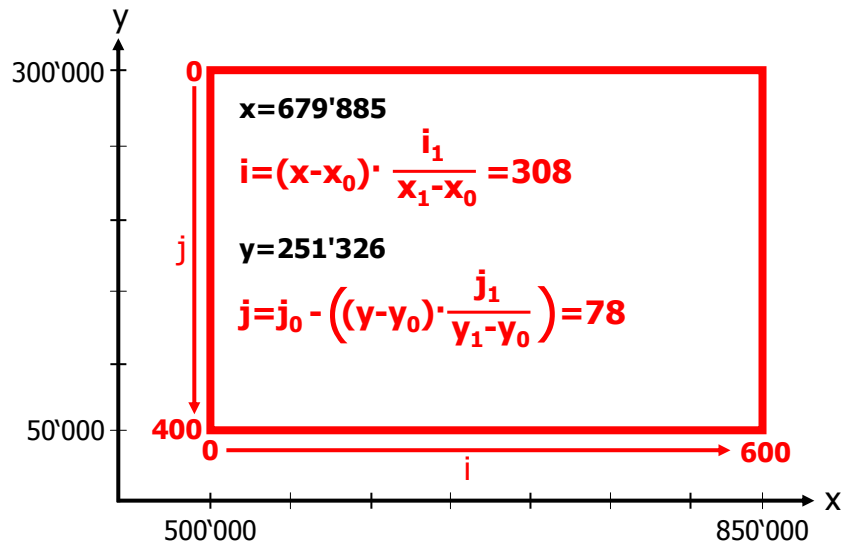
Auf einen Blick



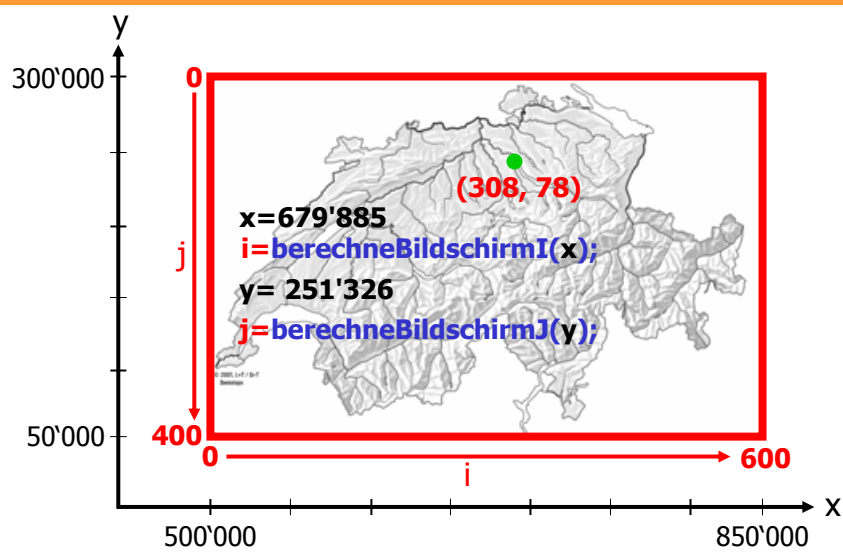
Für y bzw. j



Auf einen Blick



Wo ist nun Zürich?



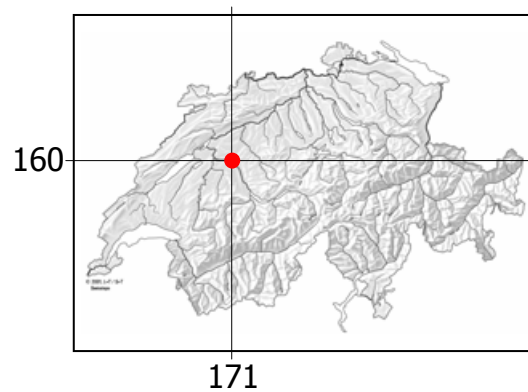
Nun in Java...

```
int berechneBildschirmI (double x) {  
    double bildschirmI;  
    bildschirmI = (x-XMIN) * FENSTERMAX_I / (XMAX-XMIN);  
    return (int) bildschirmI;  
}  
  
int berechneBildschirmJ (double y) {  
    double bildschirmJ;  
    double jverkehrt;  
    jverkehrt = (y-YMIN) * FENSTERMAX_J / (YMAX-YMIN);  
    bildschirmJ = FENSTERMAX_J - ( jverkehrt );  
    return (int) bildschirmJ;  
}
```

Bern

i=berechneBildschirmI(600000);

j=berechneBildschirmJ(200000);



Arbeiten mit TextPad

TextPad ist ein Textprogramm, mit dem wir arbeiten werden. TextPad kann man vom Web herunterladen: <http://www.textpad.com/> .

TextPad kann sehr viel. Wir werden nur einen kleinen Teil davon gebrauchen. Hier sind nur die wichtigsten Funktionen beschrieben. Ausführliche Hilfe gibt es in TextPad im Menüpunkt "Hilfe".

Datei öffnen:

Unter "Datei" und "Öffnen" kann eine Datei auf der Festplatte gesucht und geöffnet werden.

Ganz unten im Datei-Menü sind jene Dateien aufgelistet, die man kürzlich mit TextPad bearbeitet hat. So muss man meistens die Datei nicht lange suchen.

Datei speichern:

Im Menüpunkt Datei ist alles, was man braucht, um die Arbeit zu speichern. Wir werden Java-Applets schreiben. Der Filename muss deshalb die Endung ".java" haben, z.B. "MyFirstApplet.java".

Im Menüpunkt "Speichern unter" wird nach einem Dateinamen gefragt. Dies ist sehr nützlich, wenn man die Datei zwischendurch einmal unter einem anderen Namen speichern will.

Sonst werden jeweils die alten Versionen überschrieben.

Oooooops- versehentlich etwas gelöscht?

Macht nichts. Unter "Bearbeiten" gibt es den äusserst nützlichen Menüpunkt "Rückgängig". Damit kann die zuletzt gemachte Änderung rückgängig gemacht werden. Was man versehentlich soeben gelöscht hat, kommt wieder hervor.

Die Tastenkombination "Ctrl" und "z" bewirkt dasselbe. Die Rückgängig-Funktion gibt es übrigens bei den meisten Programmen!

Kompilieren

Unser Java-Programm muss übersetzt werden, damit der Computer es versteht. Der Compiler übersetzt das Java-Programm in einen Bytecode, der dann vom Computer interpretiert werden kann. Im Menü "Extras" gibt es den Punkt "Java kompilieren". Falls das Programm fehlerfrei war, gibt es nur ein kleines Knackgeräusch.

Andernfalls werden im grossen Fenster die Fehlermeldungen angezeigt. Im Fensterchen oben links erscheint der Eintrag "Programmausgabe". Um wieder zurück in den Programmcode zu wechseln, muss oben links auf den Namen der eigenen Programmdatei geklickt werden.

Das Kompilieren kann man auch mit der Tastenkombination "Ctrl" und "1" auslösen.

Applet starten

Im Menü "Extras" gibt es den Punkt "Java Applet starten", den man auch mit der Tastenkombination "Ctrl" und "3" auslösen kann. TextPad fragt nach dem Namen einer Webpage. Aus dieser heraus wird das Applet gestartet. Gibt es keine Webpage im aktuellen Ordner, so erstellt Textpad eine. Gibt es bereits eine, so kann nur diese ausgewählt, nicht aber eine neue erstellt werden.

Das Applet wird in einem eigenen Fenster dem sogenannten Appletviewer dargestellt.

Das Fensterchen oben links

Im kleinen Fenster oben links wird angezeigt, was man im grossen Fenster gerade sieht. Entweder ist das unser Programmcode, der mit dem Namen der Datei gekennzeichnet ist. Oder es ist die sogenannte Programmausgabe, was ein netter Name ist für die Fehlermeldungen vom letzten Mal kompilieren. Man kann zwischen den beiden Fensterinhalten hin und her wechseln, indem man im Fensterchen oben links auf die entsprechende Zeile klickt.

Viel Spass mit TextPad!

Java Programme kompilieren (ohne TextPad)

Um ein Java Programm zu kompilieren, musst du eine MS-DOS-Eingabeaufforderung starten. Je nach Betriebssystem versteckt sich das Programm an einer anderen Stelle

- Du findest unter „Start“ → „Programme“ das Programm „Eingabeaufforderung“
- Du findest unter „Start“ → „Programme“ → „Zubehör“ das Programm „Eingabeaufforderung“
- Wenn du das Programm mit den oberen Verweisen nicht gefunden hast, kannst du auch mit „Start“ den Befehl „Ausführen...“ wählen, darin „cmd.exe“ eintippen.

Als nächsten Schritt musst du ins Laufwerk und Verzeichnis (auch Ordner genannt) wechseln, worin du dein Java-Programm gespeichert hast.

Z.B. ins Laufwerk „h:“ wechseln:

```
h:<Enter>
```

Ins Verzeichnis „java“ wechseln:

```
cd java<Enter>
```

Jetzt kannst du dein Java-Applet kompilieren. Dazu gibt es den Befehl javac, was für den „java compiler“ steht. Für das Programm HelloWorld tippst du z.B.

```
javac HelloWorld.java<Enter>
```

Wenn du keine Fehler im Programm gemacht hast, kommt wieder eine neue Zeile mit der Eingabeaufforderung, ansonsten eine Liste mit allen Fehlern.

Behalte das Fenster mit der Eingabeaufforderung in jedem Fall offen. So kannst du sie später zum Starten des Applets und für das nächste Mal „wiederverwenden“.

Java-Applet starten (ohne TextPad)

Um dein Java-Applet zu starten, benötigst du eine HTML-Datei, wo drin steht, welches Java-Applet geladen werden soll. Dies erscheint auf den ersten Blick ein wenig stumpfsinnig. Java-Applets werden immer aus einer Webseite (eben einer HTML-Datei) gestartet.

Am besten legst du dir eine Datei an, die gleich heisst wie dein Java-Programm, einfach mit Endung „.html“ anstatt „.java“. Um möglichst wenig zu schreiben, sieht der Inhalt der HTML-Datei folgendermassen aus:

```
<BODY>
<APPLET code="HelloWorld.class" width=400 height=100>
</APPLET>
</BODY>
```

Anstelle von „HelloWorld.class“ musst du den Namen deines kompilierten Programmes angeben. Wenn dein Programm z.B. *MeinErstesProgramm.java* heisst, dann gibst du *MeinErstesProgramm.class* an. Die HTML-Datei heisst dann *MeinErstesProgramm.html*.

Um dein Programm *MeinErstesProgramm* zu starten, tippst du nun in der Eingabeaufforderung folgendes ein:

```
appletviewer MeinErstesProgramm.html<Enter>
```

Erstes Java Programm: HelloWorld.java

Das erste Java Programm schreibt eine Meldung „hello world“ auf den Bildschirm. Starte UltraEdit und erstelle dein eigenes HelloWorld Programm. Den Text kannst du auch abändern, wenn du möchtest. Speichere das Programm, kompiliere es und lasse es Laufen. Du hast 20 Minuten Zeit dazu. Falls du noch Zeit übrig hast, probiere das zweite Programm aus oder verändere das Programm nach Lust und Laune.

```
import java.awt.*;
import java.applet.*;
public class HelloWorld extends Applet
{
    public void paint (Graphics screen)
    {
        screen.drawString("hello world", 10,10);
    }
}
```

Zweites Java Programm: Text mit Rechteck

Dieses Java Applet zeichnet neben dem Text noch ein Rechteck (4 Linien) um den Text herum. Beachte, dass die Datei nun *MeinApplet2.java* heißen muss.

```
import java.applet.*;
import java.awt.*;

public class MeinApplet2 extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.red);
        g.drawLine( 10, 10, 330, 10);
        g.drawLine( 330, 10, 330, 40);
        g.drawLine( 330, 40, 10, 40);
        g.drawLine( 10, 40, 10, 10);

        g.setColor(Color.black);
        g.drawString("Das ist schon ein wenig komplizierter. ", 20, 30);
    }

}
```

Grafik-Befehle (Methoden von Graphics)

Linie zeichnen

```
void drawLine(int x1, int y1, int x2, int y2);  
z.B. screen.drawLine(10, 10, 300, 100);
```

Text ausgeben

```
void drawString(String str, int x, int y);  
z.B. screen.drawString("Grossartig!", 20, 30);
```

Rechteck zeichnen

```
void drawRect(int x, int y, int width, int height);  
z.B. screen.drawRect(20, 20, 270, 70);
```

Ausgefülltes Rechteck zeichnen

```
void fillRect(int x, int y, int width, int height);  
z.B. screen.fillRect(25, 25, 260, 60);
```

Ellipse (bzw. Kreis) zeichnen

```
void drawOval(int x, int y, int width, int height);  
z.B. screen.drawOval(30, 50, 20, 20); // zeichne Kreis mit Radius 10
```

Ausgefüllte Ellipse zeichnen

```
void fillOval(int x, int y, int width, int height);  
z.B. screen.fillOval(100, 50, 30, 20);
```

Zeichnungsfarbe ändern

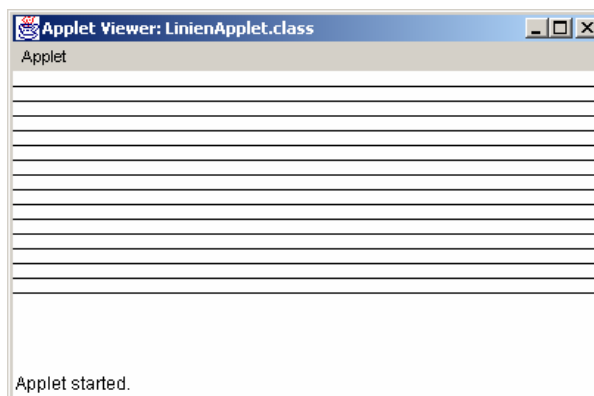
```
void setColor(Color c);  
z.B. screen.setColor(Color.pink);  
Weitere Farben sind Color.black, Color.blue, Color.cyan, Color.darkGray, Color.gray,  
Color.green, Color.lightGray, Color.magenta, Color.orange, Color.pink, Color.red,  
Color.white und Color.yellow.
```

Schrift ändern

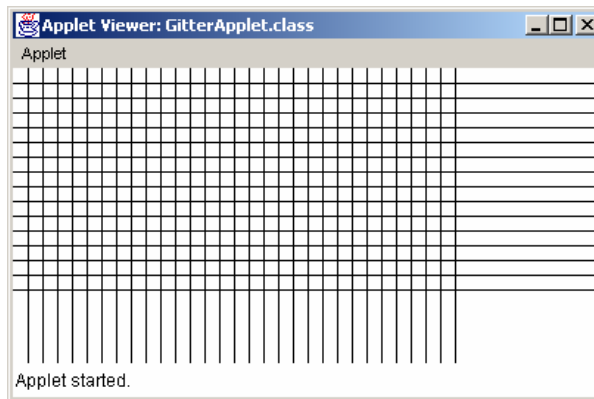
```
void setFont(Font font);  
z.B. screen.setFont(new Font("Helvetica", Font.BOLD, 36));  
Die Zahl 36 entspricht der Grösse. Font.BOLD bedeutet, dass die Schrift fett ist.  
Font.PLAIN ist Normalschrift, Font.ITALIC ist kursive Schrift.
```

Grafik-Aufgaben

1. Java stellt in der Klasse Graphics eine Methode
`void drawRect(int x, int y, int width, int height)`
zur Verfügung, die es einem erlaubt, Rechtecke zu zeichnen. Erstelle ein Java Applet Programm „RechteckApplet.java“, das eine Grösse von 400 mal 200 Bildschirmpunkte hat. Zeichne in die Mitte des Programmfensters ein Rechteck der Grösse 200 mal 100 Bildschirmpunkte.
2. Erweitere dein Proramm und zeichne die Diagonalen des Rechtecks ein. Du kannst die Änderungen in „RechteckApplet.java“ eintragen und brauchst keine neue Datei zu erstellen.
3. Mit der Methode `drawLine`, die wir schon länger kennen, können wir 4 Linien zu einem Rechteck verbinden. Schreibe eine Methode
`void zeichneRechteck(int x, int y, int width, int height, Graphics screen)`
die das gleiche Resultat liefert wie die Methode `drawRect`. Dieser Methode musst man das Graphics-Objekt als Parameter übergeben, damit du in der Methode `screen.drawLine(...)` aufrufen kannst.
Du kannst die Methode in dein Programm „RechteckApplet.java“ reintippen.
Hinweis: In Java dürfen sich Bildschirmpunkte überlappen. Ganz im Gegensatz zu Kara, der kein Kleeblatt legen durfte, wenn schon ein Kleeblatt am Boden lag. Es ist daher einfacher, wenn sich die 4 Linien in den Ecken überlappen.
4. Für diejenigen, die Aufgabe 3 geschafft haben:
Ersetze nun in der `paint`-Methode den Aufruf der Java-Methode `drawRect` durch deine Rechteck-Methode `zeichneRechteck`. Jetzt siehst du, ob du Aufgabe 3 richtig gelöst hast.
Bemerkung: Vergiss nicht zu speichern und zu kompilieren, sonst schaust du dir die alte Version an.
5. Erstelle nun eine neues Java Programm mit dem Namen „LinienApplet.java“ (wieder gleiche Grösse wie vorhin). Erzeuge in der `paint`-Methode 15 horizontale Linien der vollen Breite, jeweils im Abstand von 10 Bildpunkten. Verwende dazu eine `for`-Schleife. Das Resultat sollte etwa wie folgt aussehen:



6. Erstelle nochmals ein neues Java Programm „GitterApplet.java“. Wiederum in der Grösse 400 mal 200 Bildpunkte. Am besten kopierst du das vorhergehende Programm. Du sollst jetzt zusätzlich noch vertikale Linien darüberlegen. Wie vorhin im Abstand von 10 Bildpunkten zueinander, aber nun 30 Stück. Dein Resultat sieht etwa so aus:



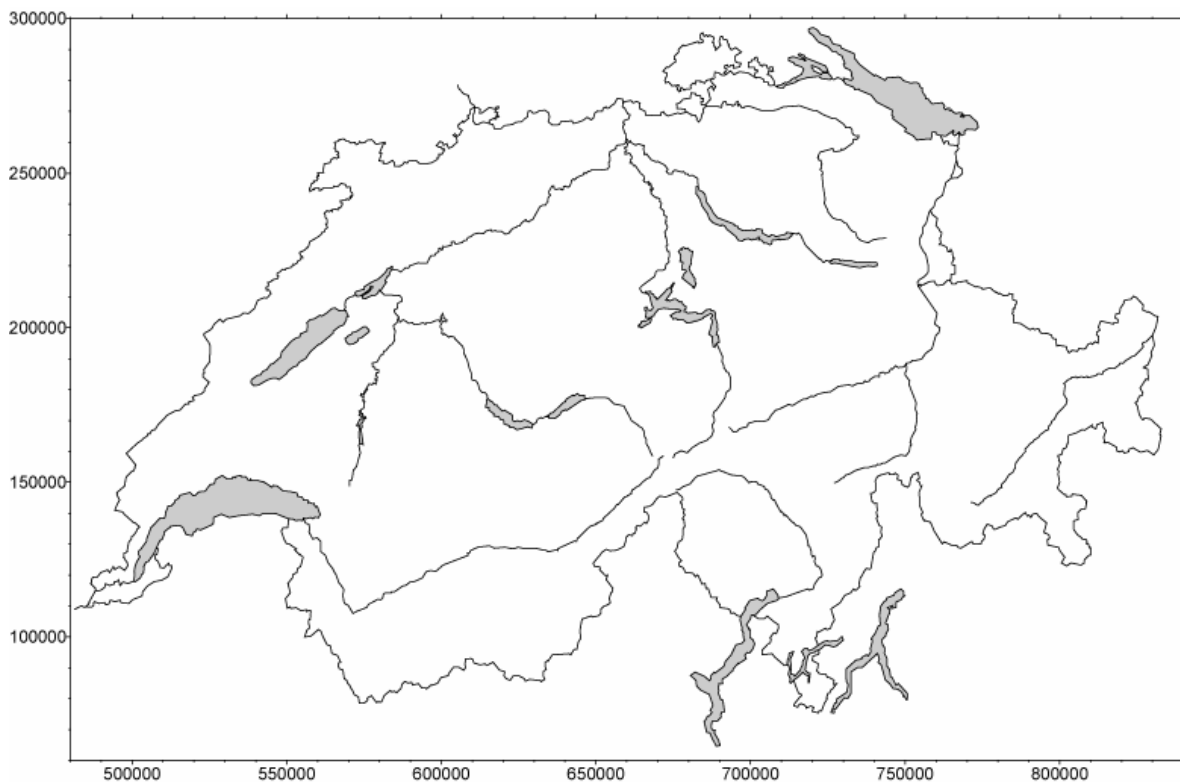
7. Zusatzaufgabe für die ganz schnellen:
Verändere dein GitterApplet nun so, dass jeweils die ersten Linien horizontal und vertikal in roter Farbe gezeichnet wird, die restlichen Linien aber wie bisher in Schwarz.
Wenn das Programm wie gewünscht läuft, betrachte das Bild ein wenig. Welche Farben haben die Schnittpunkte einer roten mit einer schwarzen Linie?

Koordinatensysteme

Wenn wir Orte von einer Schweizerkarte auf dem Bildschirm darstellen wollen, müssen wir die Koordinaten der Orte in der Welt (bzw. auf der Schweizerkarte) in die des Bildschirmes umrechnen.

Das Koordinatensystem auf den Schweizer Karten ist etwa im folgenden Bereich (Angaben sind Meter):

$$x_{\min} = 500'000 < x < 850'000 = x_{\max}$$
$$y_{\min} = 50'000 < y < 300'000 = y_{\max}$$



Die Koordinaten beim Bildschirm bewegen sich in viel kleineren Dimensionen. Bei einem Programmfenster von 600 auf 400 Bildschirmpunkten haben wir ein Koordinatensystem mit folgendem Bereich:

$$i_{\min} = 0 \leq i < 600 = i_{\max}$$
$$j_{\min} = 0 \leq j < 400 = j_{\max}$$

Um Verwechslungen vorzubeugen, nennen wir die x-Koordinate beim Bildschirm die i-Koordinate, der y-Koordinate des Bildschirm geben wir den Namen j-Koordinate.

Aufgepasst: Beim Bildschirm liegt der Ursprung am linken oberen Eckpunkt, die j-Koordinaten werden gegen unten grösser.

Wir möchten die Stadt Bern mit Koordinaten $x = 600'000$ / $y = 200'000$ auf dem Bildschirm einzeichnen. Welche (i,j)-Koordinaten hat Bern?

Die x-Koordinaten bewegen sich von 500'000 bis 850'000. Die Stadt Bern liegt etwa 100'000 Meter vom linken Kartenrand entfernt ($x - x_{\min}$). Der Abstand zwischen linkem und rechten

Rand ($x_{\max} - x_{\min}$) ist 350'000. Durch Dividieren erhalten wir den „prozentualen“ Anteil vom linken Rand nach Bern im Verhältnis zur totalen Breite. Das sind bei uns 28.6%. Wir brauchen somit nur noch mit der Breite des Bildschirmfensters ($i_{\max} - i_{\min}$) zu multiplizieren und wir erhalten das korrekte Resultat $i = 171$.

Es lässt sich also folgende Formel ableiten:

$$i = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (i_{\max} - i_{\min})$$

oder in Zahlen:

$$i = \frac{600'000 - 500'000}{850'000 - 500'000} \cdot (600 - 0) \cong 171$$

Die Formel für die j-Koordinate ist ähnlich. Durch den Umstand, dass die j-Koordinate vom oberen Rand her gemessen wird, müssen wir den prozentualen Anteil umkehren.

$$j = \left(1 - \frac{y - y_{\min}}{y_{\max} - y_{\min}} \right) \cdot (j_{\max} - j_{\min})$$

oder in Zahlen:

$$j = \left(1 - \frac{200'000 - 50'000}{300'000 - 50'000} \right) \cdot (400 - 0) = 160$$

Applet

Ein Applet ist ein abgespecktes Java-Programm. Ein Applet allein läuft nicht. Es läuft nur in einem Browser oder in einem Appletviewer. Ein Appletviewer ist eine Umgebung speziell um Applets darzustellen. Applets werden oft im Zusammenhang mit dem Internet verwendet.

Applets müssen die Methode paint haben. Diese wird vom Browser oder Appletviewer aufgerufen, sobald das Applet gestartet wird. Die paint Methode darf deshalb nicht umbenannt werden.

Typisches Rahmenprogramm:

```
import java.awt*;    /* alle Klassen, die es für die Grafik braucht */
import java.applet.*; /* alles, was es für ein Applet braucht */

public class MyApplet extends Applet { /* mein erstes Applet ist eine Erweiterung
    eines allgemeinen Applets. Achtung: Der Name der Klasse MyApplet muss identisch sein mit dem
    Namen des Java-Files, also MyApplet.java. */

    ... /* Konstanten oder Variablen je nach Bedarf definieren */

    public void paint (Graphics screen) {
    ... /* lokale Konstanten oder Variablen definieren */
        /* Anweisungen in der paint Methode */
    } /* Ende von paint*/

    ... /* eigene Methoden definieren: Falls in einer eigenen Methode gezeichnet werden soll, so
        muss der Graphics screen als Parameter übergeben werden. Z.B.
        public void Linie(double x1, double y1, double x2, double y2, Graphics screen) { ... } */
    } /* Ende vom Applet */
```

Die fett gedruckten Teile müssen vorhanden sein. Kommentare sind kursiv. Die Paint-Methode wird automatisch ausgeführt sobald das Applet in einem Browser oder Appletviewer gestartet wird. Die Paint-Methode liefert den Bildschirmausschnitt (Graphics screen), in dem wir zeichnen können. Dieser muss deshalb an alle Zeichnungsmethoden weitergegeben werden.

Aufruf des Applets

Ein Applet wird aufgerufen, indem man eine Webpage schreibt, in der auf das Applet verwiesen wird. Die vier unten folgenden Zeilen werden in ein eigenes File geschrieben. Der Name dieses Files muss die Endung ".html" haben, also z.B. MyApplet.html

```
<BODY>
<APPLET code="MyApplet.class" width=400 height=200>
</APPLET>
</BODY>
```

Bei den obigen vier Zeilen handelt es sich um eine ganz einfache HTML-Seite. Für diejenigen, die HTML kennen: Selbstverständlich können beliebige weitere HTML-Befehle vor oder nach dem Applet-Aufruf hinzugefügt werden. Allerdings darf zwischen den Tags <APPLET...> und </APPLET> nichts eingefügt werden.

Zeichnen auf dem Bildschirm

In Java gibt es viele Sammlungen von nützlichen Methoden (die sogenannten Klassen-Bibliotheken oder Libraries). Eine davon heisst Graphics oder genauer java.awt.Graphics.

Wenn man diese Zeichenbefehle verwenden will, muss man ganz am Anfang des Programms die Zeile `import java.awt.*;` einfügen. Damit wird alles was innerhalb von awt ist, (also auch Graphics) ins Programm importiert.

Awt ist übrigens die Abkürzung für Abstract Windowing Toolkit, also eine Sammlung von nützlichen Dingen, um mit Fenstern auf dem Bildschirm zu arbeiten.

Bildschirmkoordinaten

Das Koordinatensystem auf dem Bildschirm ist etwas speziell. Der Nullpunkt ist oben links.

Zunehmende x-Werte wachsen nach rechts (wie man es sich von den Koordinatensystemen aus der Mathematik gewöhnt ist). Achtung: Zunehmende y-Werte wachsen von oben nach unten - also gerade umgekehrt wie in der Mathematik.

Hier ein paar nützliche Befehle:

```
screen.drawLine(int x1, int y1, int x2, int y2);  
/* Zeichnet eine Linie bzw. Punkt*/
```

Die Angabe von screen am Anfang deutet an, dass die Linie auf den Bildschirm gezeichnet wird. Die Punkte (x1|y1) und (x2|y2) werden durch eine Linie verbunden.

Beachte: Um einen Punkt zu zeichnen, gibst du einfach für x1 und x2 und für y1 und y2 die gleichen Werte an. Z.B. `screen.drawLine(300,145,300,145);`

```
screen.drawOval(int x, int y, int width, int height);  
/* Zeichnet ein Oval bzw. Kreis */
```

Achtung: das Oval bzw. der Kreis wird durch ein umschliessendes Rechteck bzw. Quadrat beschrieben. Der Punkt (x|y) gibt die linke ober Ecke dieses Vierecks an. Die beiden weiteren Parameter geben die Breite (width) und Höhe (height) des Vierecks an.