Inhaltsverzeichnis

1 Die Klasse Main

1.1 Template der Mainklasse

Die Mainklasse steuert das Programm. Sie ist eine Erweiterung der Klasse `Grafiktakt` und kann in BlueJ mit dem Template `grafiktaktMain` erzeugt werden:

```
import grafiktakt.*;
import java.awt.*;

public class Main extends Grafiktakt
{
    public void start() {
    }
    public void zeichnen() {
    }
    public void metronom() {
        repaint();
    }
    public void taste(int tastencode) {
        if(tastencode == ESC) {
            beenden();
        }
    }
    public static void main() {new Main().init(500, 500);} // Fenstergröße
    public static void main(String[] args) { main(); }
}
```

1.2 Start des Programms

public static void **main**() und public static void **main**(String[ ] args)

Die beiden `main`-Methoden am Ende erzeugen das Programmfenster und bleiben unverändert. Lediglich die Fenstergröße kann angepasst werden. Bei negativer Fenstergröße wird das Programm im Vollbildmodus gestartet.

1.3 Methoden zur Steuerung des Programmablaufs, die bereits im Template enthalten sind

public void **start**()

Die Methode wird beim Start des Programms aufgerufen. Hier können z.B. Variablen initialisiert und Objekte erzeugt werden.

public void **zeichnen**()

Die Methode wird aufgerufen, wenn das Bild neu gezeichnet werden muss.

public void **metronom**()

Die Methode wird alle $\frac{1}{100}$ Sekunden aufgerufen.

public void **taste**()

Die Methode wird aufgerufen, wenn eine Taste gedrückt wurde. `tastencode` nimmt folgende Werte an:

`'a'`, `'B'`, `'?'` , `'3'` ...	Standardtasten (in einfachen Anführungszeichen),
`LINKS`, `RECHTS`, `OBEN`, `UNTEN`	vordefinierte Konstanten für die Pfeiltasten,
`ESC`, `ENTER`	vordefinierte Konstanten für `ESC` und `ENTER`,
negative Werte	weitere Sondertasten wie `F1` oder `Pos1`.

1.4 Weitere Methoden zur Steuerung des Programmablaufs

public void **maus**(int mauscode, double x, double y)

Wird diese Methode eingefügt, wird sie bei einer Mausaktion aufgerufen:

```
public void maus(int mauscode, double x, double y) {
    if (mauscode == MLDOWN) {
        System.out.println("Die Mausposition ist "+x+" / "+y);
    }
}
```

`mauscode` nimmt folgende Werte an:

`MRDOWN` / `MLDOWN`	Maustaste gedrückt,
`MRUP` / `MLUP`	Maustaste losgelassen,
`MAUSRECHTS` / `MAUSLINKS`	Maustaste gedrückt und losgelassen,
`MMOVE` / `MDRAGG`	Maus bewegt ohne / mit Tastendruck.

public void **stopp**()

Wird diese Methode eingefügt, wird sie beim Schließen des Fensters aufgerufen. In ihr können z.B. Daten gesichert werden.

public boolean **stopp**()

Die Methode kann auch einen Wert vom Typ `boolean` zurückgeben. Ist dieser gleich `false`, wird das Schließen des Fensters verhindert und das Programm fortgesetzt.

1.5 Methoden der Klasse Grafiktakt, die verwendet werden können

void **beenden**()

Das Fenster wird geschlossen.

void **repaint**()

Das Betriebssystem wird darüber informiert, dass das Bild neu gezeichnet werden muss. Diese Methode muss z.B. aufgerufen werden, wenn sich Objekte im Bild bewegt haben.

boolean **tasteGedrückt**(int tastencode)

Es wird überprüft ob eine spezielle Taste gerade gedrückt ist. `tastencode` nimmt die bereits oben bei der Methode `taste(int tastencode)` beschriebenen Werte an. Mit dieser Methode kann überprüft werden, ob mehrere Tasten gleichzeitig gedrückt werden.

1.6 MP3Player

new **MP3Player**(String dateiname);

Erzeugt einen mp3-Player für die Datei *dateiname*. Diese muss im mp3-Format vorliegen.

void **play**(boolean loop);

Spielt das Musikstück einmal (*loop* == *false*) oder immer wieder (*loop* == *true*).

void **stopp**();

Beendet den mp3-Player.

2 Grafikbefehle der Klasse B

2.1 allgemeine Einstellungen

void **B.bereich**(double links, double unten, double rechts, double oben);
Das Koordinatensystem wird so eingestellt, dass die Ränder des Bildes bei den angegebenen Koordinaten liegen. Einer der Werte kann gleich der Konstante `B.AUTO` sein. Dieser Rand wird dann automatisch eingestellt, so dass die Einheiten auf der x- und der y-Achse gleich groß sind.

void **B.verschiebung**(double dx, double dy);
Der zuvor eingestellte Bereich wird um *dx* nach rechts und um *dy* nach oben verschoben.

void **B.verschiebung**();
Ab jetzt wird das Bild nicht mehr verschoben.

void **B.streckung**(double s);
Alle folgenden Objekte werden um den Faktor *s* gestreckt.

void **B.streckung**(double sx, double sy);
Für die x- und die y-Richtung können unterschiedliche Streckfaktoren angegeben werden.

void **B.streckung**();
Ab jetzt werden die Objekte nicht mehr gestreckt.

double **B.randLinks**(), **B.randUnten**(), **B.randRechts**(), **B.randOben**();
Berechnet die Koordinaten der Ränder im aktuellen Koordinatensystem.

int **B.breite**(), **B.höhe**();
Berechnet die Breite und die Höhe des Bildes in Pixeln.

double **B.pixelBreite**(), **B.pixelHöhe**();
Berechnet die Breite und die Höhe eines Pixels in Einheiten des aktuellen Koordinatensystems.

double **B.yFaktor**();
Berechnet um welchen Faktor ein Pixel in den Einheiten des aktuellen Koordinatensystems höher ist als breit.

void **stift**(double x);
Die Dicke der im folgenden gezeichneten Linien wird festgelegt.

void **farbe**(Color farbe), **farbe**(int rgb), **farbe**(int r, int g, int b), **farbe**(String name);
In `zeichnen()` wird die aktuelle Zeichenfarbe festgelegt, außerhalb von `zeichnen()` die Hintergrundfarbe.

void **font**(String name, int style, int size), **font**(int size);
Stellt die Schriftart ein. Z.B.: `B.font("Arial", Font.BOLD, 20)`. `style` nimmt die Werte `Font.PLAIN`, `Font.BOLD` oder `Font.ITALIC` an.

void **transparent**(boolean b);
Stellt ein, ob der Hintergrund unter einem Text gelöscht wird ($b = false$) oder nicht ($b = true$, Standard).

double **zeilenabstand**();
Berechnet den Zeilenabstand der aktuellen Schriftart in Einheiten des aktuellen Koordinatensystems.

2.2 Befehle zum Zeichnen

Jedem der folgenden Befehle kann ein optionaler erster Parameter farbe übergeben werden, der hier blau markiert ist. Und es kann ein optionaler letzter Parameter stift übergeben werden, der hier rot markiert ist. Sowohl farbe als auch stift können bei bei jedem der folgenden Befehle weggelassen werden.

Wird farbe angegeben, ersetzt diese für dieses eine Objekt die aktuelle Zeichenfarbe. Ist stift gleich 0, wird das Objekt mit der aktuellen Stiftdicke gezeichnet. Ist stift größer als 0, ersetzt der Wert die aktuelle Stiftdicke. Wird stift nicht angegeben, wird das Objekt ausgefüllt gezeichnet. Beispiele:

`B.kreis("blau", 0, 0, 50, 2)`	ein 2 Pixel dicker blauer Kreisring
`B.kreis("blau", 0, 0, 50, 0)`	ein blauer Kreisring mit der aktuell eingestellten Stiftdicke
`B.kreis("blau", 0, 0, 50)`	ein ausgefüllter blauer Kreis
`B.kreis(0, 0, 50)`	ein ausgefüllter Kreis in der aktuell eingestellten Zeichenfarbe

void **B.punkt**(farbe, x, y, stift);
Ein Punkt wird gezeichnet.

void **B.linie**(farbe, x1, y1, x2, y2, stift);
Eine Linie wird gezeichnet.

void **B.rechteck**(farbe, x1, y1, x2, y2, stift);
Ein Rechteck wird gezeichnet.

void **B.ellipse**(farbe, x1, y1, x2, y2, stift);
Eine Ellipse wird gezeichnet, die in das Rechteck mit den angegebenen Eckpunkten hineinpasst.

void **B.kreis**(farbe, mx, my, radiusx, stift);
Ein Kreis wird gezeichnet.

void **B.quadrat**(farbe, mx, my, radiusx, stift);
Ein Quadrat wird gezeichnet. Parameter wie bei Kreis: Mittelpunkt und Abstand von diesem bis zum rechen Rand.

void **B.sektor**(farbe, x1, y1, x2, y2, start, winkel, stift);
void **B.segment**(farbe, x1, y1, x2, y2, start, winkel, stift);
void **B.bogen**(farbe, x1, y1, x2, y2, start, winkel, stift);
Ein Sektor, ein Segment oder ein Bogen wird als Teil der Ellipse gezeichnet, die in das Rechteck mit den angegebenen Eckpunkten hineinpasst. *start* und *winkel* sind Winkel im

Gradmaß.

void **B.sektorRad**(...);
void **B.segmentRad**(...);
void **B.bogenRad**(...);
Bogenmaß-Version.

void **B.ecke**(x, y);
Eine Ecke eines Vielecks wird definiert.

void **B.vieleck**(farbe, stift);
void **B.vieleck**(farbe, x1, y1, x2, y2, x3, y3, ..., , stift);
Ein Vieleck wird gezeichnet. Entweder aus den zuvor definierten oder mit den direkt angegebenen Eckpunkten.

void **B.streckenzug**(farbe, stift);
void **B.streckenzug**(farbe, x1, y1, x2, y2, x3, y3, ..., , stift);
Der Unterschied zum Vieleck ist hier, dasss der erste und der letzte Punkt nicht verbunden werden.

2.3 Texte

Allen Textbefehlen kann ein optionaler erster Parameter farbe übergeben werden, der hier blau markiert ist. farbe kann weggelassen werden. Wird der Parameter angegeben, ersetzt die Farbe die aktuelle Zeichenfarbe für diesen einen Text.

Die `text(...)`-Befehle schreiben einen Text vom Typ String auf den Bildschirm, die `zahl(...)`-Befehle schreibe Zahl vom Typ double.

void **B.text**(farbe, t, x, y);
void **B.zahl**(farbe, z, x, y);
Der Text `t` oder die Zahl `z` werden an der angegebenen Position auf den Bildschirm geschrieben.

void **B.text**(farbe, t);
void **B.zahl**(farbe, z);
Es wird hinter dem zuletzt ausgegebenen Text weitergeschrieben.

void **B.text**(farbe, t, x1, y1, x2, y2);
void **B.zahl**(farbe, z, x1, y1, x2, y2);
Der Text oder die Zahl werden in dem durch die zwei Eckpunkte angegebenen Rechteck zentriert

void **B.text**(farbe, t, x1, y1, x2, y2, position);
void **B.zahl**(farbe, z, x1, y1, x2, y2, position);
`position` nimm die Werte `B.LINKS`, `B.OBEN`, `B.RECHTS` oder `B.UNTEN` für die Position in dem angegebenen Rechteck an.

void **B.doubleFormat**(String format);
z.B. `"000,00"`. Legt das Format für die Ausgabe von Dezimalzahlen mit dem Befehl `zahl` fest.

2.4 Bilder

Image **B.ladeBild**(String filename);

Lädt ein Bild aus einer Datei.

Image **B.definiereBild**(Color farbe, daten[]);

`daten[]` ist ein Array vom Typ `long` oder `int` und enthält zeilenweise die Pixeldaten für das zu definierende Bild.

boolean **B.bild**(Image img, x, y);
boolean **B.bild**(Image img, x, y, int position);
boolean **B.bild**(Image img, x1, y1, x2, y2);
boolean **B.bild**(winkel, boolean spiegeln, strecken, Image img, x, y);
boolean **B.bild**(winkel, boolean spiegeln, strecken, Image img, x, y, int position);
boolean **B.bild**(winkel, boolean spiegeln, Image img, x1, y1, x2, y2);

Ein Bild wird gezeichnet und eventuell gespiegelt, um den Faktor `strecken` gestreckt und um `winkel`° gedreht. `winkel` und / oder `spieglen` und `strecken` können weggelassen werden.

Es kann ein Punkt `(x / y)` gemeinsam mit `position` wie beim Befehl `text` angegeben werden, oder ein Rechteck. Dann wird das Bild gestreckt bzw. gestaucht, dass es in Rechteck passt.

boolean **B.bildRad**(...)

Alle Methoden mit `winkel` haben eine Bogenmaß-Version.