Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Animationen ohne Multithreading

Ziel: Kennenlernen elementarer Möglichkeiten einfache Animationen in Java darzustellen und erkennen, welche Probleme sich ergeben, wenn keine Multithreading verwendet wird.

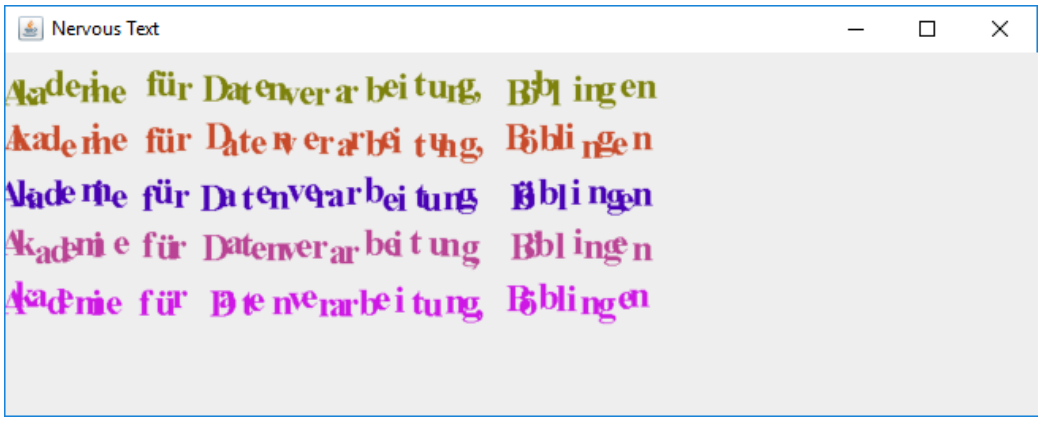
1.1. Die Klasse `TumblingDuke` soll erweitert werden, dass `ManyTumblingDukes` daraus werden:

- Bei einer Fensterbreite von 1000 Pixel und einer Höhe von 700 Pixel, sollen Reihen von „tumbling Dukes“ ihre animierten „Räder“ gleichzeitig ausführen. Die Breite des aktuellen JFrames kann man mit `this.getWidth()` ermitteln, die Höhe entsprechend mit `this.getHeight()`.
- Ein einzelnes Anmationsbild hat die Maße 130x80 (Breite x Höhe). Die Anzahl der gleichzeitigen Animationen ergibt sich also aus der Größe des JFrame und der Größe der Bilder, wobei immer nur ganze Bilder angezeigt werden sollen.
- Nach einem Durchlauf der Animation soll das „Rad“ gleich anschließend in die andere Richtung „geschlagen“ werden, dann eine kurze Verschnaufpause – und so weiter … und so weiter…

Die Grundversion von `TumblingDuke` finden Sie in Moodle, so dass diese nur entsprechend erweitert werden muss.

```
public class I13ManyTumblingDukes extends JFrame
{
    // array vom Typ Image zur Speicherung der Adressen der 17 Einzelbilder
    private Image bilder[] = new Image[17];

    public I13ManyTumblingDukes()
    {
        setBackground(Color.white);
        for (int i = 1; i <= 17; i++)
        {
            try // die Methode 'read' von 'ImageIO' verlangt eine sog. "Ausnahmebehandlung" (try-catch)
            {
                bilder[i - 1] = ImageIO.read(getClass().getResource("../TumblingDuke/T" + i + ".gif"));

            }
            catch (IOException e)
            {
                e.getMessage();     // ==> Fehlernachricht wird im Konsolenfenster angezeigt
            }
        }
    }

    public void paint(Graphics g)
    {
        int pos = 0;                    // Positionszähler für array
        int pauseNachAnimation = 2000;  // Pause von 2000 Milli-Sekunden, d.h. 2 Sek.
        int pauseNachBild = 100;        // Pause von 0,1 Sek. nach Einzelbild

        boolean bRichtung = false;  // true: vorwärts, false: rückwärts

        for (int i=0;;i++)
        {
            // Da die Koordinaten (0/0) im Rand des JFrames liegen, wird noch eine
            // zusätzliche Verschiebung um (30/10) vorgenommen
           for (int iYPos = 0; iYPos < this.getHeight() - 30 - 80; iYPos += 80)
           {
               for (int iXPos = 0; iXPos < this.getWidth() -10 - 130; iXPos += 130)
               {
                   g.drawImage(bilder[pos], iXPos + 10, iYPos + 30, this);
               }
           }

            try
            {
                if (pos == 16 || (pos == 0 && bRichtung))
                    Thread.sleep(pauseNachBild + pauseNachAnimation);
                else Thread.sleep(pauseNachBild);
            }
            catch (InterruptedException e)
            {
                System.exit(0);
            }
            if (pos == 16) bRichtung = true;
            else if (pos == 0) bRichtung = false;
            if (bRichtung) pos--;
            else pos++;
        }
    }

    public static void main(String[] args)
    {
        I13ManyTumblingDukes fenster = new I13ManyTumblingDukes();
        fenster.setSize(1000, 700);
        fenster.setVisible(true);
    }
}
```

1.2. Erstellen Sie folgende Java-Swing-Animation, die einen „nervösen“ Text ausgibt:



- Größe des JFrames: 1000/400
- Der RGB-Wert der Farbe wird zufällig bestimmt. (Siehe Methode setColor() in 28_Java_GUI_Graphics).
- Die Schrift ist "TimesRoman", fett, Schriftgröße 36 (Siehe Methode setFont() in 28_Java_GUI_Graphics).
- Jeder neue Buchstabe ist um 15 Pixel + einem zufälligen Wert von 0-10 Pixel nach rechts verschoben
- Die Zeilen erscheinen jeweils 50 Pixel untereinander, die 1. Zeile bei (0/50); jeder Buchstabe ist um einen zufälligen Wert von max. 10 Pixel nach unten versetzt.
- Die Buchstaben sollen in einem zeitlichen Abstand von 0,1 Sek. erscheinen.
- Zum Zeichnen der Buchstaben verwendet man die Methode drawChars() (siehe Java Doku: Klasse Graphics)

```
public class NervousText extends JFrame
{
    private char buchstaben[];
    private String s = "Akademie für Datenverarbeitung, Böblingen";
    private int x = 0, y = 0;

    NervousText()
    {
        super.setTitle("Nervous Text");
        buchstaben = s.toCharArray();
        setFont(new Font("TimesRoman", Font.BOLD, 36));
    }

    public void paint(Graphics g)
    {
        int zeilenabstand = 50;

        g.clearRect(0, 0, 1000, 400);

        for (int zeile = 1; zeile <= 5; zeile++)
        {
            int rotAnteil = (int) (Math.random() * 255);
            int gruenAnteil = (int) (Math.random() * 255);
            int blauAnteil = (int) (Math.random() * 255);
            Color farbe = new Color(rotAnteil, gruenAnteil, blauAnteil);
            g.setColor(farbe);
            for (int i = 0; i < buchstaben.length; i++)
            {
                x = (int) (Math.random() * 10 + 15 * i);
                y = (int) (Math.random() * 10 + 36) + zeilenabstand;
                g.drawChars(buchstaben, i, 1, x, y);
                try
                {
                    Thread.sleep(100);
                }
                catch (InterruptedException e)
                { }
            }
            zeilenabstand += 50;
        }
    }

    public static void main(String[] args)
    {
        NervousText fenster = new NervousText();
        fenster.setSize(1000, 400);
        fenster.setVisible(true);
    }
```