

Übungen zur Vorlesung Software-Entwicklung

Java-Technologie: Verstehen und Modifizieren eines Java-Programms.

In dieser Übungsstunde soll die Funktionsweise eines Java-Programms in den Grundzügen verstanden werden. Ohne auf syntaktische Einzelheiten und deren Variationen einzugehen, soll ein Verständnis für einen Programmablauf gewonnen werden, so dass kleinere, lokale Modifikationen durchgeführt werden können, die zu einer sichtbaren Veränderung des Programmverhaltens führen.

Das Ziel dieser Übung ist also das Verstehen der Programmlogik, die zum sichtbaren Programmverhalten führt. Es wird nicht erwartet, dass Sie anschließend Java programmieren können.

1. Voraussetzungen

Bis zum Beginn der Übungsstunde sollten Sie die folgenden Abschnitte über das Programm sorgfältig gelesen haben. Versuchen Sie, so viel wie möglich davon zu verstehen. Machen Sie sich keine Gedanken über syntaktische Einzelheiten wie Kommas und Strichpunkte. Falls zum Programm Fragen offen bleiben, können Sie diese während der Übungsstunden stellen.

2. Das Java-Programm ClickMe

Wir werden uns das Java-Programm ClickMe im Detail anschauen. Ähnlich wie das Applet aus der letzten Übungsstunde erzeugt es einen Punkt als Reaktion auf einen Mausklick. Sie werden danach selbst in der Lage sein, kleine Veränderungen daran vorzunehmen. Hier zunächst das vollständige Programm, bevor wir in die Einzelheiten gehen.

```
import java.awt.*;
import java.awt.event.*;

public class ClickMe extends javax.swing.JFrame {

    public static void main(String args[]) {
        new ClickMe();
    }

    public ClickMe() {
        super("ClickMe");
        setContentPane(new DrawPane());
        setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        setSize(400, 400);
        setVisible(true);
    }
}

class DrawPane extends javax.swing.JPanel implements MouseListener {

    public DrawPane() {
        addMouseListener(this);
    }

    private int x = -1, y = -1;
    private static int RADIUS = 7;

    public void drawBackground(Graphics g) {
        // draw a 2 line blue border on a white background
        g.setColor(Color.white);
        g.fillRect(0, 0, this.getSize().width - 1, this.getSize().height - 1);

        g.setColor(Color.blue);
        g.drawRect(1, 1, this.getSize().width - 3, this.getSize().height - 3);
        g.drawRect(2, 2, this.getSize().width - 5, this.getSize().height - 5);
    }

    public void paint(Graphics g) {
        // draw the spot
        if (x != -1) {
            drawBackground(g);
            g.setColor(Color.red);
            g.fillOval(x - RADIUS, y - RADIUS, RADIUS * 2, RADIUS * 2);
        } else {
            drawBackground(g);
        }
    }

    public void mousePressed(MouseEvent event) {
        x = event.getX();
        y = event.getY();
        this.repaint();
    }

    public void mouseClicked(MouseEvent event) {}
    public void mouseReleased(MouseEvent event) {}
    public void mouseEntered(MouseEvent event) {}
    public void mouseExited(MouseEvent event) {}
}
```

3. Bekanntmachen von Namen

```
import java.awt.*;
import java.awt.event.*;
```

Die ersten zwei Zeilen des Java-Programms sorgen dafür, dass man vordefinierte Java-Programme über ihre sog. einfachen Namen verwenden kann. Das Java-Laufzeitsystem stellt nämlich eine große Anzahl vordefinierter Java-Programme zur Verfügung, die man als Java-Programmierer nutzen kann. Diese Programme sind in Bibliotheken zusammengefasst, die man in Java „Pakete“ nennt. Unter anderem stehen die folgenden Pakete in Java zur Verfügung:

Swing		Java 2D		AWT	Accessibility	
Drag and Drop		Input Methods		Image I/O	Print Service	Sound
IDL	JDBC	JNDI	RMI	RMI-IIOP		Scripting
Beans	Security		Serialization		Extension Mechanism	
JMX	XML JAXP		Networking		Override Mechanism	
JNI	Date and Time		Input/Output		Internationalization	
lang and util						
Math	Collections		Ref Objects		Regular Expressions	
Logging	Management		Instrumentation		Concurrency Utilities	
Reflection	Versioning		Preferences API		JAR	Zip

Diese Pakete enthalten Java-Programme für mathematische Berechnungen (Math), für Datum und Zeit (Date and Time), für Sicherheitsaspekte (Security), für Datenbankzugriffe (JDBC), für Benutzungsschnittstellen (AWT, Swing) u.v.m.

Die ersten beiden mit dem Schlüsselwort „import“ beginnenden Programmzeilen bewirken, dass man im nachfolgenden Programmtext alle Programme aus den Paketen „java.awt“ und „java.awt.event“ über ihre dort definierten Namen verwenden kann.

Natürlich muss man als Java-Programmierer wissen, welche Programme in welchen Paketen existieren. Zu Beginn ist der Zugang zu dieser Information etwas mühsam, aber mit zunehmender Erfahrung kennen sich die Programmierer immer besser aus und können auch immer schneller die notwendigen Informationen aus der Java-Dokumentation extrahieren.

4. Klassen

```
public class ClickMe extends javax.swing.JFrame {
    ...

    class DrawPane extends javax.swing.JPanel implements MouseListener {
        ...
    }
}
```

Java-Programme bestehen aus der Definition sog. Klassen. In unserem Fall besteht das Programm aus zwei Klassen, der Klasse „ClickMe“ und der Klasse „DrawPane“.

Java-Klassen werden oft als Erweiterungen einer bestehenden Java-Klasse definiert. Hier wird ClickMe als Erweiterung der Klasse JFrame (`extends javax.swing.JFrame`) und DrawPane als Erweiterung der Klasse JPanel definiert. JFrame und JPanel sind vordefinierten Java-Klassen aus dem Paket javax.swing. Durch Voranstellen des Paketnamens vor den Klassennamen werden die

Klassennamen eindeutig identifiziert. Der Effekt der Angabe „extends“ besteht darin, dass ClickMe dieselben Eigenschaften wie JFrame und DrawPane dieselben Eigenschaften wie JPanel aufweist. Insbesondere wird sichergestellt, dass sich unser Programm als ein Fenster mit Zeichenfläche verhält.

Die Angabe (`implements MouseListener`) in der Definition der Klasse DrawPane ist wichtig, damit die Zeichenfläche auf Mausklicks reagieren kann. Auch `MouseListener` gehört zu einem Java-Paket. Diesmal haben wir nicht den Paketnamen vorangestellt, da wir diesen Namen aus dem Java-Paket `awt.event` „importiert“ hatten.

5. Methoden

```
public static void main(String args[]) {  
    new ClickMe();  
}
```

Die Klasse ClickMe enthält eine sog. Methode. Methoden sind ausführbare Teilprogramme, die über den Aufruf ihres Namens gestartet werden. Die drei obigen Zeilen definieren eine solche Methode mit dem Namen „main“. Eine Methode dieses Namens wird vom Java-Laufzeitsystem immer dann aufgerufen, wenn das Programm als Ganzes gestartet werden soll. Sie muss also für jedes Java-Programm definiert werden.

Nach der ersten Zeile, in der hauptsächlich der Name der Methode angegeben wird, öffnet sich eine geschweifte Klammer, die 2 Zeilen weiter, am Ende der Methode, wieder geschlossen wird. Alle Texte, die innerhalb der geschweiften Klammern stehen, bilden eine Folge sog. Anweisungen, in denen formuliert ist, was der Computer nacheinander tun soll.

In dieser Methode gibt es nur eine Anweisung. Sie bewirkt, dass von der Klasse ClickMe eine sog. Instanz erzeugt wird. Über Instanzen wird in der Lehrveranstaltung noch ausführlich zu sprechen sein. An dieser Stelle so es genügen zu wissen, dass durch die Erzeugung der sog. Konstruktor der Klasse ClickMe aktiviert wird.

6. Konstruktoren

```
public ClickMe() {  
    super("ClickMe");  
    setContentPane(new DrawPane());  
    setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);  
    setSize(400, 400);  
    setVisible(true);  
}
```

Konstruktoren werden ausgeführt, wenn eine Instanz einer Klasse – wie oben – erzeugt wird. Konstruktoren sind ähnlich wie Methoden. Sie enthalten Anweisungen, die nach Aktivierung nacheinander ausgeführt werden. Unser Konstruktor enthält 5 Anweisen, die dazu führen, dass auf dem Bildschirm ein Fenster mit dem Title „ClickMe“ erscheint, das 400x400 Pixel groß ist. Die Einzelheiten können wir an dieser Stelle getrost ignorieren.

```
public DrawPane() {  
    addMouseListener(this);  
}
```




Dies ist der Konstruktor der Klasse „DrawPane“. Seine einzige ausführbare Anweisung besteht darin, den Inhalt des Fensters für Mausereignisse zu sensibilisieren. Durch ihn wird die Zeichenfläche zum Interessent für Mausereignisse, d.h. sie wird zum „MouseListener“.

7. Variablen

```
private int x = -1, y = -1;  
private static int RADIUS = 7;
```

In diesen beiden Zeilen der Klassendefinition von DrawPane werden sog. Variable definiert. Variable sind Namen für Speicherzellen, in die unser Java-Programm Zahlenwerte abspeichern kann. Über die Namen kann man auf die abgespeicherten Werte wieder zugreifen, um sie an anderen Stellen des Programms zu verwenden. Hier werden drei solche Variablen (`x`, `y` und `RADIUS`) definiert und die Werte -1, -1 und 7 in sie abgespeichert. Diese Werte werden später im Programm für die x- und y-Position und den Radius des zu zeichnenden Punktes verwendet. Der Wert -1 für x und y bewirkt, dass am Anfang beim Starten des Programms das Zeichnen des Punktes unterbleibt. Wir werden das später noch sehen.

7. Die Methode drawBackground

```
public void drawBackground(Graphics g) {  
    //draw a black border and a white background  
    g.setColor(Color.white);  
    g.fillRect(0, 0,    
                  getSize().width,   
                  getSize().height);   
    g.setColor(Color.black);  
    g.drawRect(0, 0,   
               getSize().width-1,  
               getSize().height-1);  
}
```

Die Klasse DrawPane enthält neben dem Konstruktor und den Variablen die Methoden „drawBackground“, „paint“ und „mousePressed“.

Bei der obigen Definition von „drawBackground“ steht nach dem Namen ein Klammerausdruck mit den beiden Elementen „Graphics“ und „g“.

In der Klammer nach dem Methodennamen werden sog. Parametervariablen definiert. Wie bei den oben beschriebenen Variablen `x` und `y` wird auch hier eine Speicherzelle beim System reserviert und mit einem Namen, in diesem Fall „g“, versehen. Wenn die Methode als Teilprogramm abläuft, befindet sich in der Speicherzelle ein Datenobjekt, das es ermöglicht, Grafikausgaben zu machen.

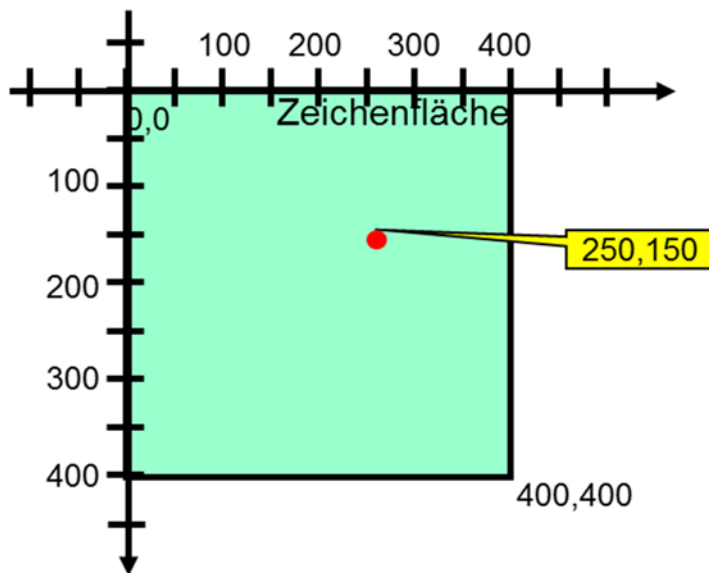
Der Bereich zwischen den geschweiften Klammern besteht hier aus 4 Anweisungen:

Die erste Anweisung besteht aus dem Aufruf einer Methode namens „setColor“ auf das Datenobjekt, das Inhalt der Speicherzelle „g“ ist. Man erkennt Methodenaufrufe syntaktisch an dem Punkt zwischen Datenobjekt und Methodennamen.

In der ersten Anweisung wird also die Methode „setColor“ für das Datenobjekt in der Variablen „g“ aufgerufen. Zusätzlich hat der Methodenaufbau ein sog. Argument. Dies steht in der Klammer dahinter (`Color.white`). Zusammen bewirkt diese Anweisung, dass die Farbe, die sich das System im Datenobjekt „g“ merkt, auf „white“ gesetzt wird.

Die zweite Anweisung erstreckt sich über insgesamt 3 Zeilen. Wie alle Anweisungen endet sie mit einem Semikolon. Diesmal wird die Methode „fillRect“ für das Datenobjekt in „g“ angewendet. Dies bewirkt, dass ein rechteckiger Bereich mit der Farbe aufgefüllt wird, die man vorher angegeben hat, also Weiß. Die beiden Koordinaten für die linke obere Ecke des Rechtecks und für seine Breite und Höhe werden getrennt durch Kommas als Argumente des Methodenaufbaus angegeben.

Die Koordinaten für die linke, obere Ecke des Rechtecks sind die ersten beiden Argumente des Methodenaufrufs „fillRect“, also 0/0. Die weiteren beiden Argumente für die Breite und Höhe des Rechtecks sind selbst Methodenaufrufe. In diesem Fall werden – ohne auf Einzelheiten an dieser Stelle einzugehen – Breite und Höhe von den am Anfang festgelegten Werten übernommen.



Der sichtbare Effekt der beiden ersten Anweisungen der Methode „drawBackground“ besteht im Zeichnen des weißen Hintergrundes.

Die beiden nächsten Zeilen sorgen für das Zeichnen eines schwarzen Randes.

Hierbei wird zunächst die Farbe für das Datenobjekt in der Variablen „g“ auf „Color.black“, also Schwarz gesetzt. Dann wird mit Hilfe der Methode „drawRect“ und denselben Argumenten wie zuvor ein Rechteck gezeichnet. Von der Breite und der Höhe wird allerdings der Wert „1“ abgezogen, damit die Linie innerhalb der Zeichenfläche verläuft.

Man beachte den Unterschied zwischen den Methoden „fillRect“ und „drawRect“: die erste füllt eine Fläche, während die zweite einen Rahmen zeichnet.

8. Die Methode paint

```
public void paint(Graphics g) {  
    drawBackground(g);  
    //draw the spot  
    g.setColor(Color.red);  
    if (x != -1) {  
        g.fillOval(x - RADIUS, y - RADIUS,  
                  RADIUS * 2, RADIUS * 2);  
    }  
}
```

Diese Methode sorgt dafür, dass zunächst die vorher definierte Methode „drawBackground“ aufgerufen und danach ein roter Punkt an einer bestimmten Stelle mit einer bestimmten Größe gezeichnet wird.

Der Aufbau des Methodenkopfs ist derselbe wie bei der Methode „drawBackground“. Einziger Unterschied ist der Methodenname.

Der Methodenkörper zwischen den geschweiften Klammern besteht aus mehreren Anweisungen. Die erste Anweisung ist der Aufruf der Methode „drawBackground“. An dieser Stelle wird also die oben beschriebene Methode „drawBackground“ ausgeführt, was – wie wir wissen – zum Zeichnen des Hintergrunds mit dem Rahmen führt.

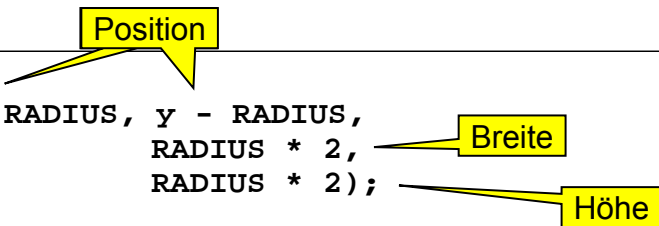
In der nächsten Zeile steht ein Kommentar (//draw the spot). Kommentare haben für die Programmausführung keinerlei Bedeutung. Sie dienen lediglich dem menschlichen Leser eines Programms dazu, Hinweise zum Programmtext zu erhalten.

Danach wird die Farbe des Datenobjekts „g“ zur Vorbereitung der Punktdarstellung auf Rot (Color.red) gesetzt.

Die drei folgenden Zeilen bestehen aus einer sog. bedingten Anweisung. Bedingte Anweisungen werden nur ausgeführt, wenn eine Bedingung erfüllt ist. In diesem Fall wird die Methode „fillOval“ nur aufgerufen, wenn der Wert der Variablen x nicht gleich „-1“ ist (x != -1). Die Variablen x und y wurden ja am Anfang auf diesen Wert gesetzt. Dies bedeutet, dass beim ersten Zeichnen kein(!) roter Punkt gezeichnet wird.

Wir werden beim Beschreiben der nächsten Methode „mousePressed“ sehen, wie die Werte der Variablen x und y durch Drücken der Maustaste verändert werden. Hier wollen wir davon ausgehen, dass dies bereits passiert ist, dass also x und y andere Werte als -1 haben.

Dann wird nämlich die Methode „fillOval“ für das Datenobjekt „g“ aufgerufen. Diese Methode füllt eine ovale Form mit der zuvor angegebenen Farbe. Die Argumente haben dieselbe Bedeutung wie bei den schon beschriebenen Methoden „fillRect“ und „drawRect“. Die ersten beiden Werte sind die Koordinaten der linken oberen Ecke und das dritte und vierte Argument ist die Breite und die Höhe der ovalen Form:



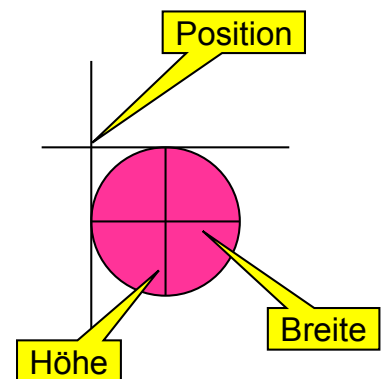
```
...  
    g.fillOval(x - RADIUS, y - RADIUS,  
              RADIUS * 2, RADIUS * 2);  
...
```

Schauen wir uns die Argumente etwas näher an:

Breite und Höhe der ovalen Form berechnen sich aus dem doppelten Radius ($\text{RADIUS} * 2$). Damit wird die ovale Form zum Kreis.

Die Position berechnet sich arithmetisch aus der Differenz von X-Wert und dem Wert der Variablen RADIUS sowie der Differenz des Y-Wertes und der Variablen RADIUS. Warum ist dies so?

Wenn wir uns vorstellen, dass in den Variablen x und y die Position des letzten Mausklicks gespeichert ist, dann müssen wir die Differenzbildung vornehmen, damit der Mittelpunkt des zu zeichnenden Punktes genau an dieser Position zu liegen kommt. Das liegt an den Eigenheiten der Definition grafischer Objekte in Java, die immer von der linken, oberen Ecke und nicht von der Mitte aus gezeichnet werden:



9. Die Methode mousePressed

```
public void mousePressed(MouseEvent event) {  
    x = event.getX();  
    y = event.getY();  
    repaint();  
}
```

Es bleibt zu beschreiben, wie die Variablen x und y ihre Werte bekommen. Dazu dient die Methode mousePressed. Sie wird bei jedem Mausklick innerhalb der Zeichenfläche aufgerufen. Ihr Kopf entspricht dem Kopf der anderen Methoden. Ausnahme ist die Angabe der Variablen in den runden Klammern. Hier wird im System eine Variable eingerichtet, die ein Datenobjekt benennt, das ein sog. Mausereignis darstellt. Mausereignisse beschreiben die Position der Maus und den Zustand der Maustasten.

In den ersten beiden Anweisungen des Methodenkörpers werden die Variable x und y mit den Koordinaten des Mausereignisses gesetzt, d.h. diese Werte werden in die Speicherzelle geschrieben, auf die man mit „x“ und „y“ zugreifen kann.

Anschließend wird die Methode „repaint“ aufgerufen. Diese Methode ist für Zeichenflächen vordefiniert und führt über uns unsichtbare Umwege dazu, dass der Browser die oben beschriebene Methode „paint“ aufruft.

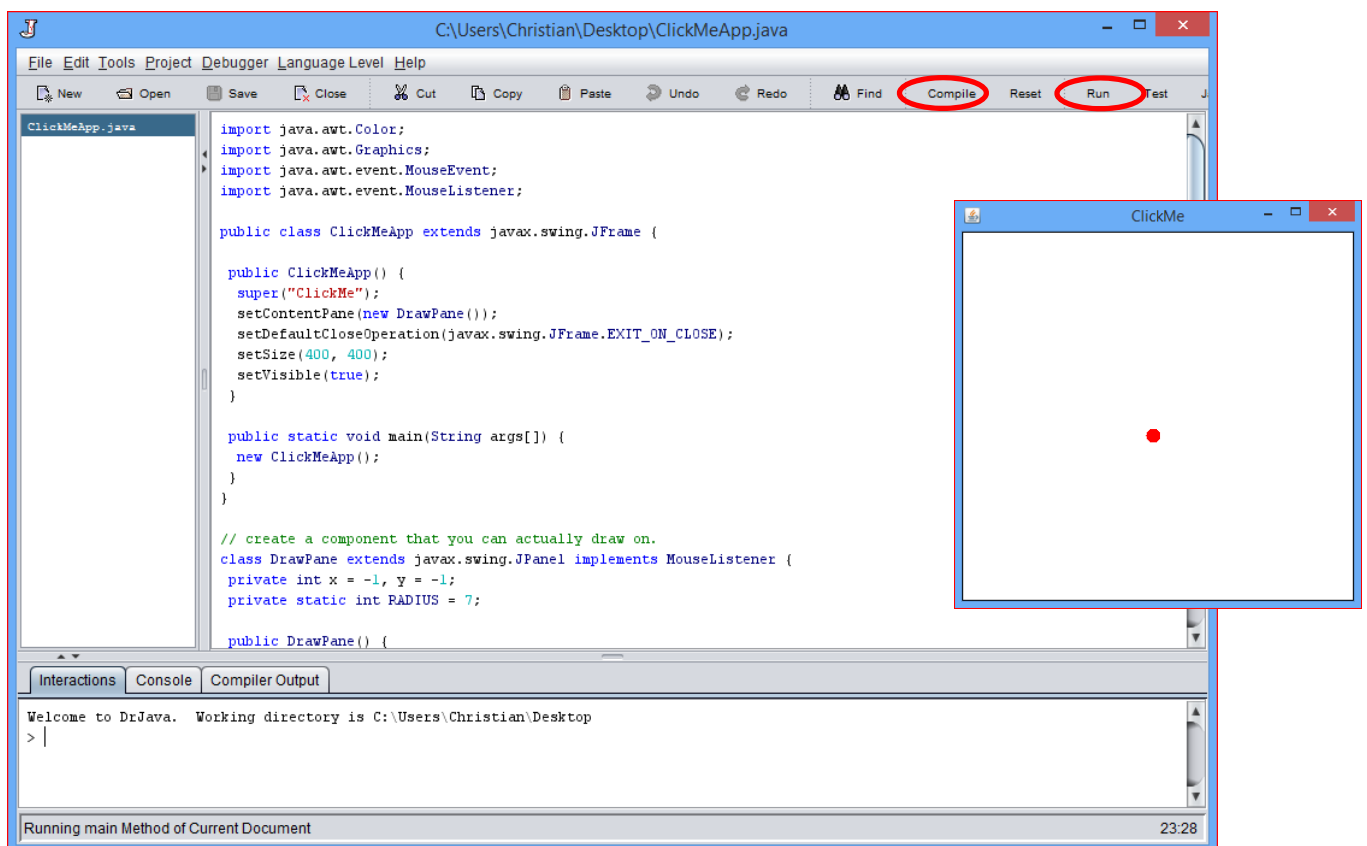
Praktische Übungen

1. Voraussetzungen

- Der Rechner muss angeschaltet und hochgefahren sein.
- Man sollte als Benutzer auf dem Rechner eingeloggt sein.

2. Speichern und Starten des ClickMe-Programms

Speichern Sie die Datei ClickMeApp.java von der Seite des Moodle-Kurses für diese Lehrveranstaltung im Abschnitt „Java-Technologie“. Öffnen Sie die gespeicherte Datei „ClickMeApp.java“ mit Hilfe des Programmeditors DrJava. Erzeugen Sie mit Hilfe von „Compile“ die übersetzte Version. Mit Hilfe von „Run“ starten Sie das Programm und können durch Mausklicks die roten Punkte setzen.



3. Übungen

Beantworten Sie durch Ausprobieren: Welche anderen Farben kann man in Java direkt über ihre Namen verwenden?

Nehmen Sie nacheinander folgende Modifikationen vor und überprüfen Sie jeweils separat das Resultat durch Starten des Programms. Bitte denken Sie daran, vor jeder Programmänderung das ClickMe-Fenster zu schließen und nach jeder Änderung das Programm zu speichern und zu übersetzen.

- Der Hintergrund soll ohne Rahmen erscheinen.
- Der Hintergrund soll in der Farbe Blau erscheinen.
- Statt eines roten Punktes soll ein grünes Quadrat gezeichnet werden.
- Das Quadrat soll einen schwarzen Rand erhalten.
- Das Quadrat soll viermal so groß sein, d.h. die Seitenlänge soll sich verdoppeln.
- Ein zuvor gezeichnetes Quadrat soll bei erneutem Mausklick nicht gelöscht werden.
- Anstelle des ursprünglichen Punktes soll ein roter Punkt in einem grünen Quadrat ausgegeben werden.
- Anstatt des Quadrats soll Ihr Name an der Maus-Position erscheinen.

Hinweis: verwenden Sie dazu die Methode „drawString“. Diese Methode hat drei Argumente: eine Zeichenfolge (in Anführungszeichen eingeschlossen), die x-Koordinate und die y-Koordinate.

- Modifizieren Sie das Programm, so dass es Ihren Namen in Lila ausgibt.

Hinweis: Neue Farbobjekte mit beliebigen, anderen Farben werden durch die Verwendung von `new` und einem Konstruktor der Klasse `Color` erzeugt, d.h. anstatt `g.setColor(Color.black)` schreibt man `g.setColor(new Color(255,255,255))`.

Dabei werden Mischungen von Rot, Grün und Blau verwendet. Diese haben je nach Intensität Werte zwischen 0 und 255. Welche Farbmischung ergibt Lila?