

Üben 3 - Ein- und Ausgabe, CodeDraw, Debugger, Schleifen

Einführung in die Programmierung 1
Wintersemester 23



Vorbereitung

- TUWEL öffnen, IntelliJ-Projekt „Ueben3.zip“ herunterladen
- Projekt in IntelliJ öffnen
- Laden Sie das bearbeitete Projekt am Ende wieder in TUWEL hoch

Passwords

- Schreiben Sie ein Programm, das überprüft, ob das eingegebene Passwort sicher ist
 - Lesen Sie das Passwort ein (mit einem Scanner)
 - Ein Passwort wird als sicher eingestuft, wenn es mindestens aus 12 Zeichen besteht und einen Großbuchstaben enthält

Hinweis: Verwenden Sie die String-Methoden `length()` und `charAt(int index)`

Geben Sie ein Passwort ein:

afDpMdMxGpBr

Das Passwort ist sicher

Geben Sie ein Passwort ein:

hallowelt

Das Passwort ist nicht sicher

Passwords – 1. Erweiterung (optional)

- Erweitern Sie das Programm so, dass ein Passwort als sicher eingestuft wird, wenn folgende Bedingungen erfüllt sind
 - Das Passwort besteht aus mindestens 12 Zeichen
 - Das Passwort enthält mindestens einen Großbuchstaben, einen Kleinbuchstaben und eine Ziffer

Geben Sie ein Passwort ein:

kxNpD4MaB2fG

Das Passwort ist sicher

Geben Sie ein Passwort ein:

afDpMdMxGpB

Das Passwort ist nicht sicher

Passwords – 2. Erweiterung (optional)

- Erweitern Sie das Programm so, dass es ausgibt, warum ein Passwort als unsicher eingestuft wird
 - Wenn das Passwort aus mehreren Gründen als unsicher betrachtet wird, müssen **alle** nicht geltenden Bedingungen mitgeteilt werden

Geben Sie ein Passwort ein:

halLo

Das Passwort ist nicht sicher

- * Das Passwort ist zu kurz
- * Das Passwort enthält keinen Großbuchstaben
- * Das Passwort enthält keine Ziffer

Passwords – 3. Erweiterung (optional)

- Berechnen Sie die Entropie E des Passwortes:

$$E = L \cdot \frac{\log_{10} N}{\log_{10} 2}$$

Passwortlänge

N ... Größe des Zeichensatzes

Hinweis: Verwenden Sie die statische Methode `log10(double value)` aus der Klasse `Math`, um Logarithmen zur Basis 10 zu berechnen

N	Im Passwort enthaltene Zeichen
10	Nur Ziffern
26	Nur Groß- oder nur Kleinbuchstaben
36	Ziffern und nur ein Typ von Buchstaben
52	Keine Ziffern, aber beide Groß- und Kleinbuchstaben
62	Ziffern, Groß- und Kleinbuchstaben

Passwords – 3. Erweiterung (optional)

- Geben Sie den berechneten Entropiewert E und die entsprechende Sicherheitsstufe (gemäß der folgenden Skala) aus

Entropie	Sicherheitsstufe
$0 \leq E < 36$	Sehr schwach
$36 \leq E < 60$	Schwach
$60 \leq E < 120$	Sicher
$E \geq 120$	Sehr sicher

Geben Sie ein Passwort ein:

password

Das Passwort ist nicht sicher

- * Das Passwort ist zu kurz
- * Das Passwort enthält keinen Großbuchstaben
- * Das Passwort enthält keine Ziffer

Entropie: 37.603517745128734 (schwach)

Geben Sie ein Passwort ein:

f4PfGmk87PMgQaZb67BQX8fb

Das Passwort ist sicher

Entropie: 142.90071144928498 (sehr sicher)

Passwords – Mögliche Lösung

```
import java.util.Scanner;

public class Passwords {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Geben Sie ein Passwort ein:");
        String password = sc.nextLine();

        boolean hasUppercase = false;
        for (int i = 0; i < password.length() && !hasUppercase; i++) {
            char c = password.charAt(i);
            if ('A' <= c && c <= 'Z') hasUppercase = true;
        }

        if (password.length() >= 12 && hasUppercase) {
            System.out.println("Das Passwort ist sicher");
        } else {
            System.out.println("Das Passwort ist nicht sicher");
        }
    }
}
```


Passwords – Mögliche Lösung 1. Erweiterung

```
import java.util.Scanner;

public class Passwords {
    public static void main(String[] args) {
        ...

        boolean hasUppercase = false, hasLowercase = false, hasDigit = false;
        for (int i = 0; i < password.length(); i++) {
            char c = password.charAt(i);
            if ('A' <= c && c <= 'Z') hasUppercase = true;
            else if ('a' <= c && c <= 'z') hasLowercase = true;
            else if ('0' <= c && c <= '9') hasDigit = true;
        }

        if (password.length() >= 12 && hasUppercase && hasLowercase && hasDigit) {
            System.out.println("Das Passwort ist sicher");
        }
        ...
    }
}
```

Passwords – Mögliche Lösung 2. Erweiterung

```
import java.util.Scanner;

public class Passwords {
    public static void main(String[] args) {
        ...

        if (password.length() >= 12 && hasUppercase && hasLowercase && hasDigit) {
            System.out.println("Das Passwort ist sicher");
        } else {
            System.out.println("Das Passwort ist nicht sicher");
            if (password.length() < 12)
                System.out.println("* Das Passwort ist zu kurz");
            if (!hasUppercase)
                System.out.println("* Das Passwort enthält keinen Großbuchstaben");
            if (!hasLowercase)
                System.out.println("* Das Passwort enthält keinen Kleinbuchstaben");
            if (!hasDigit)
                System.out.println("* Das Passwort enthält keine Ziffer");
        }
    }
}
```

Passwords – Mögliche Lösung 3. Erweiterung

```
import java.util.Scanner;

public class PasswordsSolution {
    public static void main(String[] args) {
        ...

        int alphabetSize = 0;
        if (hasUppercase) alphabetSize += 26;
        if (hasLowercase) alphabetSize += 26;
        if (hasDigit) alphabetSize += 10;

        double entropy = alphabetSize == 0 ? 0 : password.length() * Math.Log10(alphabetSize) / Math.Log10(2);
        System.out.print("Entropie: " + entropy + " ");

        if (entropy < 36)
            System.out.println("(sehr schwach)");
        else if (entropy < 60)
            System.out.println("(schwach)");
        else if (entropy < 120)
            System.out.println("(sicher)");
        else
            System.out.println("(sehr sicher)");
    }
}
```

Field

- Korrigieren Sie das Fußballfeld durch Adaptierung und Ergänzung des Codes
- Den meisten Lernerfolg haben Sie, wenn Sie nicht nur ausprobieren, sondern vorher überlegen, wodurch Sie was verändern

```
import codedraw.*;

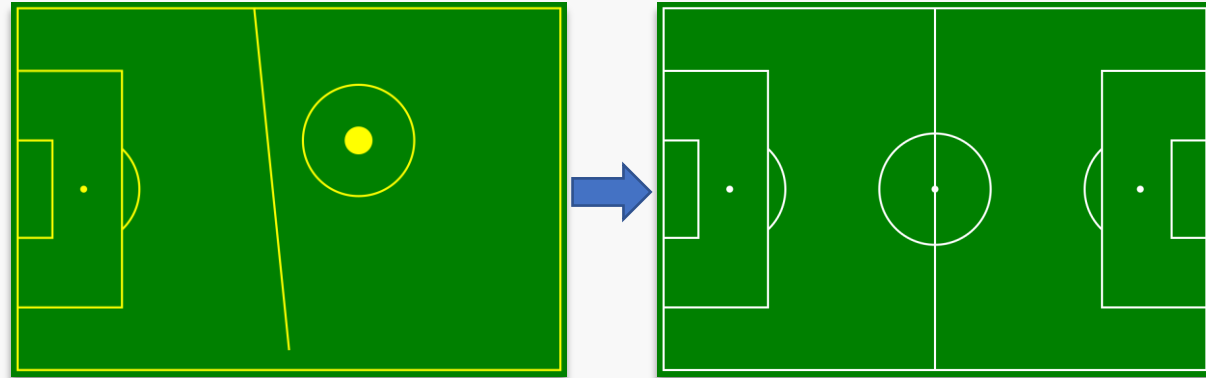
public class Field {
    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(800, 540);
        cd.setColor(Palette.GREEN);
        cd.fillRect(0, 0, 800, 540);

        cd.setColor(Palette.YELLOW);
        cd.setLineWidth(3);
        cd.drawRect(10, 10, 780, 520);

        cd.drawRect(10, 100, 150, 340);
        cd.drawRect(10, 200, 50, 140);
        cd.fillCircle(105, 270, 5);
        cd.drawArc(105, 270, 80,
            Math.toRadians(-45), Math.toRadians(90));

        cd.drawLine(350, 10, 400, 500);
        cd.drawCircle(500, 200, 80);
        cd.fillCircle(500, 200, 20);

        cd.show();
    }
}
```



(Linienfarbe weiß)

<https://github.com/Krassnig/CodeDraw/blob/master/INTRODUCTION.md>
<https://krassnig.github.io/CodeDrawJavaDoc/v4.0.x/>

Field – Mögliche Lösung

```
import codedraw.*;

public class Field {
    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(800, 540);
        cd.setColor(Palette.GREEN);
        cd.fillRect(0, 0, 800, 540);

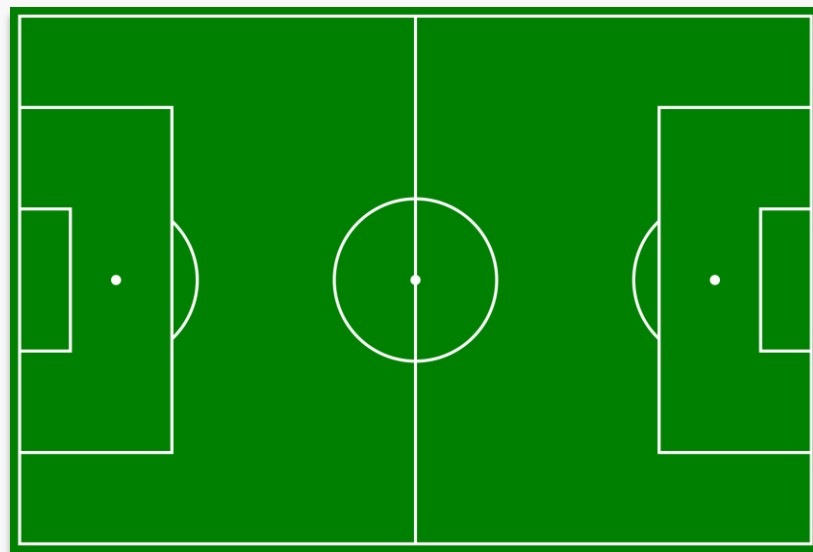
        cd.setColor(Palette.WHITE);
        cd.setLineWidth(3);
        cd.drawRect(10, 10, 790, 530); // Out-Linien

        cd.drawRect(10, 100, 150, 340); // Linker Strafraum
        cd.drawRect(10, 200, 50, 140);
        cd.fillCircle(105, 270, 5);
        cd.drawArc(105, 270, 80, Math.toRadians(-45), Math.toRadians(90));

        cd.drawLine(400, 10, 400, 530); // Mittellinie
        cd.drawCircle(400, 270, 80); // Mittelkreis
        cd.fillCircle(400, 270, 5); // Mittelpunkt

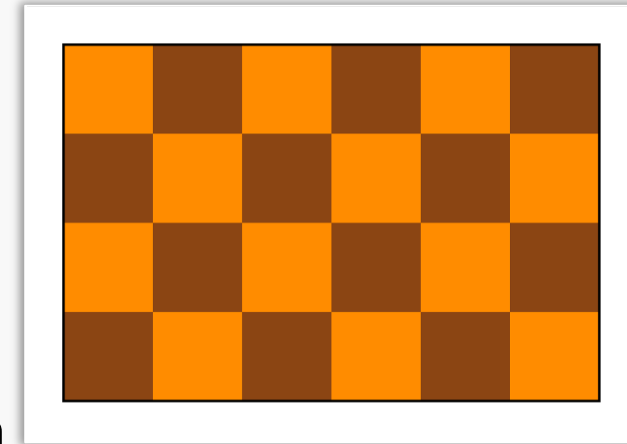
        cd.drawRect(640, 100, 680, 340); // rechter Strafraum
        cd.drawRect(720, 200, 760, 140);
        cd.fillCircle(675, 270, 5);
        cd.drawArc(675, 270, 80, Math.toRadians(135), Math.toRadians(90));

        cd.show();
    }
}
```



CodeDrawDebug

- Der gegebene Code soll die dargestellte Grafik erzeugen, enthält jedoch Fehler
- **Benutzen Sie den Debugger**, um die Fehler aufzuspüren und auszubessern:
 - Setzen Sie einen Breakpoint in Zeile 19
 - Gehen Sie Zeile für Zeile durch das weitere Programm
 - Beobachten Sie die Werte der Variablen
 - *Als Hilfestellung:*
 - Die korrekten x-Koordinaten der oberen linken Ecken der Quadrate sind: 35.5, 107, 178.5, 250, 321.5, 393
- Es dürfen keine Ausgaben mittels `System.out` erzeugt werden!



Debugger

File Edit View Navigate Code Refactor Build Run jGRASP Tools Git Window Help

Schleifen src StarPattern

Project

Schleifen

src

DoWhileTest

ForTest

GamblingSimulation

GuessingGame

Illusion

Multiplication

Pattern

PrimeTest

Schleifen.zip

StarPattern

Test

WhileTest

StarPattern.java

```
1 public class StarPattern {
2
3     public static void main(String[] args) { args: []
4         int n = 5; n: 5
5         for (int i = -n; i <= n; i++) { i: -5 i: -5
6             for (int j = -n; j <= n; j++) { n: 5
7                 if (i * i <= j * j) {
8                     System.out.print("* ");
9                 } else {
10                    System.out.print(" ");
11                }
12            }
13        }
14    }
```

Breakpoint

Debugger einschalten

Debug: StarPattern

Debugger Console jGRASP

Frames

*main"...UNNING

main:6, StarPattern

Variables

args = {String[0]@807} []

n = 5

i = -5

Einzelne Programmschritte ausführen

Variablen inspizieren

Memory Overhead

Class Count Diff

No classes loaded. Load classes

Event Log

All files are up-to-date (a minute ago)

CRLF UTF-8 4 spaces main

Werte von Ausdrücken inspizieren

- Mauszeiger auf gewünschten Ausdruck (Operator) bewegen

The image displays two screenshots of a code editor interface, illustrating how to inspect the value of an expression by hovering over an operator.

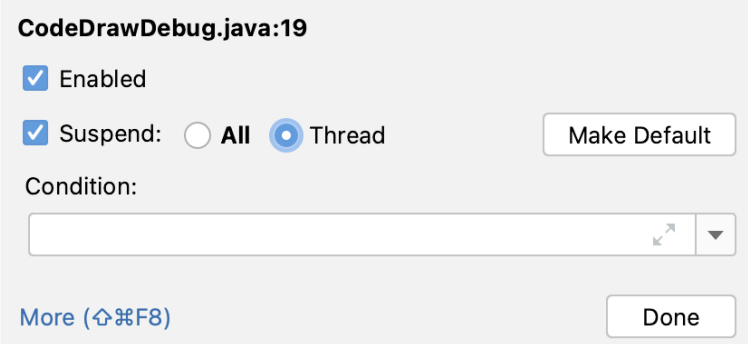
Top Screenshot: The code is `int result = a + b * c; a: 2 b: 3 c: 5`. The mouse cursor is positioned over the multiplication operator `*`. A tooltip shows the value `15`, which is the result of `b * c` (3 * 5). A callout box points to the tooltip with the text "Maus über *".

Bottom Screenshot: The code is the same: `int result = a + b * c; a: 2 b: 3 c: 5`. The mouse cursor is now positioned over the addition operator `+`. A tooltip shows the value `17`, which is the result of `a + b * c` (2 + 15). A callout box points to the tooltip with the text "Maus über +".

Debuggen von CodeDraw-Programmen

- Die Einstellungen von Breakpoints müssen angepasst werden, um die einzelnen Zeichenoperationen sehen zu können

1. Rechtsklicken Sie auf einen gesetzten Breakpoint
2. Wählen Sie neben „Suspend“ die Option „Thread“
3. Klicken Sie auf den Button „Make Default“, um diese Einstellung für alle Breakpoints zu übernehmen



CodeDrawDebug.java:19

☒ Enabled

☒ Suspend: ☐ All ☒ Thread Make Default

Condition:

[More \(⌘F8\)](#) Done

<https://github.com/Krassnig/CodeDraw/blob/master/INTRODUCTION.md#debugging-codedraw>

CodeDrawDebug – Mögliche Lösung

...

```
double startX = (canvasWidth - boxLength) / 2.;
double startY = (canvasHeight - boxHeight) / 2.;
double size = (double) boxHeight / rows;

for (int row = 1; row <= rows; row++) {
    double xOffset = 0;
    boolean brown = row % 2 == 0;
    while (xOffset < boxLength) {
        cd.setColor(brown ? Palette.SADDLE_BROWN : Palette.DARK_ORANGE);
        cd.fillRect(startX + xOffset, startY + (row - 1) * size, size, size);
        xOffset += size;
        brown = !brown;
    }
}

...
```

Lines

```
import codedraw.*;

public class Lines {
    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(400, 200);
        cd.setLineWidth(2);

        cd.setColor(Palette.DARK_TURQUOISE);
        cd.drawLine(0, 0, 200, 200);
        cd.drawLine(40, 0, 200, 160);
        cd.drawLine(80, 0, 200, 120);
        cd.drawLine(120, 0, 200, 80);
        cd.drawLine(160, 0, 200, 40);

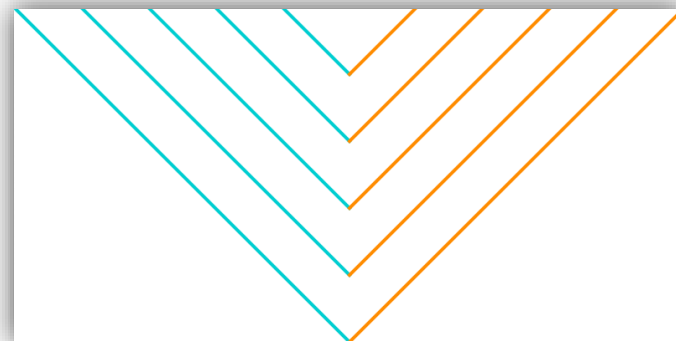
        cd.setColor(Palette.DARK_ORANGE);
        cd.drawLine(400, 0, 200, 200);
        cd.drawLine(360, 0, 200, 160);
        cd.drawLine(320, 0, 200, 120);
        cd.drawLine(280, 0, 200, 80);
        cd.drawLine(240, 0, 200, 40);

        cd.show();
    }
}
```

Ändern Sie den Code so, dass alle gleichfarbige Linien mit einer einzelnen Schleife gezeichnet werden!

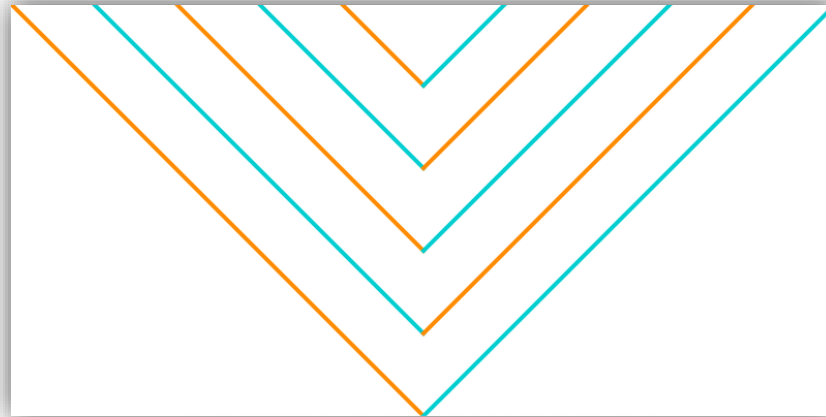
Hilfreiche Überlegungen

1. Wie viele Linien jeder Farbe gibt es?
2. Wo beginnt und endet eine Linie und wie groß ist der Abstand zwischen den Enden zweier aufeinanderfolgenden Linien?



Lines – Optionale Erweiterungen

1. Zeichnen Sie alle Linien mit einer einzelnen Schleife
2. Die Farbe ändert sich mit jeder Linie (beginnend bei Orange)



Lines – Mögliche Lösung

```
import codedraw.*;

public class Lines {
    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(400, 200);
        int distance = 40, middle = cd.getWidth() / 2;

        cd.setLineWidth(2);
        cd.setColor(Palette.DARK_TURQUOISE);
        for (int i = 1; i <= 5; i++) {
            cd.drawLine(middle - i * distance, 0, middle, i * distance);
        }
        cd.setColor(Palette.DARK_ORANGE);
        for (int i = 1; i <= 5; i++) {
            cd.drawLine(middle + i * distance, 0, middle, i * distance);
        }

        cd.show();
    }
}
```

Lines – Mögliche Lösung 1. Erweiterung

```
import codedraw.*;

public class Lines {

    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(400, 200);
        int distance = 40;
        int middle = cd.getWidth() / 2;

        cd.setLineWidth(2);
        for (int i = 1; i <= 5; i++) {
            cd.setColor(Palette.DARK_TURQUOISE);
            cd.drawLine(middle - i * distance, 0, middle, i * distance);
            cd.setColor(Palette.DARK_ORANGE);
            cd.drawLine(middle + i * distance, 0, middle, i * distance);
        }

        cd.show();
    }
}
```

Lines – Mögliche Lösung 2. Erweiterung

```
import codedraw.*;

public class LinesSolutionExt2 {

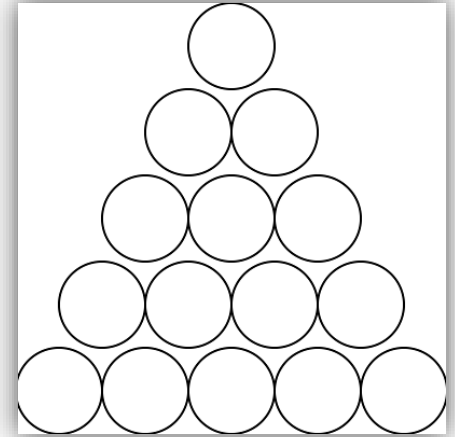
    public static void main(String[] args) {
        CodeDraw cd = new CodeDraw(400, 200);
        int distance = 40;
        int middle = cd.getWidth() / 2;

        cd.setLineWidth(2);
        for (int i = 1; i <= 5; i++) {
            cd.setColor((i % 2 == 0) ? Palette.DARK_TURQUOISE : Palette.DARK_ORANGE);
            cd.drawLine(middle - i * distance, 0, middle, i * distance);
            cd.setColor((i % 2 == 1) ? Palette.DARK_TURQUOISE : Palette.DARK_ORANGE);
            cd.drawLine(middle + i * distance, 0, middle, i * distance);
        }

        cd.show();
    }
}
```

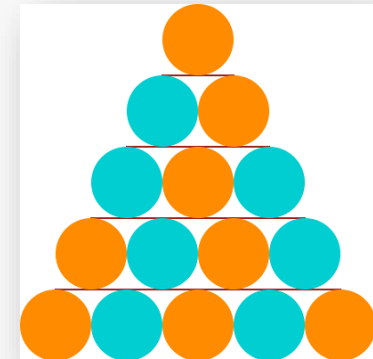
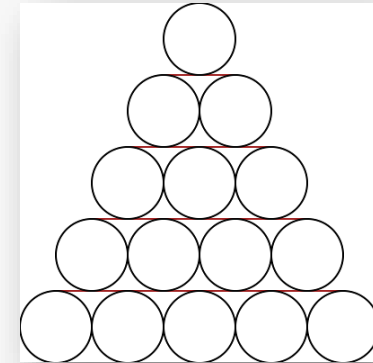
Circles

- Erweitern Sie den Code in der Klasse `Circles`, um die gezeigte Ausgabe zu erzeugen
 - Der Mittelpunkt des oberen Kreises liegt bei $(\text{width} / 2, 40)$
 - Der Radius aller Kreise beträgt 40 Pixel
 - Jede Reihe enthält einen Kreis mehr als die davor und die Kreise einer Reihe sollten sich berühren, aber nicht überlappen
 - Der Mittelpunkt des ersten Kreises einer Reihe liegt 80 Pixel unterhalb und 40 Pixel links vom Mittelpunkt des ersten Kreises in der Reihe darüber
 - Es sollen so viele Kreise gezeichnet werden, wie vollständig in das Zeichenfenster passen



Circles – Optionale Erweiterungen

1. Fügen Sie unter jeder Reihe von Kreisen eine braune Linie mit der gleichen Länge wie die jeweilige Reihe hinzu
2. Die Kreise werden abwechselnd mit Orange oder Türkis gefüllt, von oben beginnend mit Orange
 - Farben: Palette.*DARK_ORANGE*, Palette.*DARK_TURQUOISE*



Circles – Mögliche Lösung

```
import codedraw.*;

public class Circles {

    public static void main(String[] args) {
        int width = 400, height = 400;
        CodeDraw cd = new CodeDraw(width, height);
        cd.setLineWidth(2);

        int radius = 40;
        int yCenter = radius;
        int numCircles = 1;
        while (yCenter + radius <= height) {
            int xCenter = width / 2 - (numCircles - 1) * radius;
            for (int n = 0; n < numCircles; n++) {
                cd.drawCircle(xCenter + 2 * radius * n, yCenter, radius);
            }
            numCircles++;
            yCenter += 2 * radius;
        }
        cd.show();
    }
}
```

Circles – Mögliche Lösung 1. Erweiterung

```
import codedraw.*;

public class Circles {
    public static void main(String[] args) {
        ...

        while (yCenter + radius <= height) {
            int xCenter = width / 2 - (numCircles - 1) * radius;
            int lineLength = 2 * numCircles * radius;

            cd.setColor(Palette.BROWN);
            cd.drawLine(xCenter - radius, yCenter + radius, xCenter - radius + lineLength, yCenter + radius);
            cd.setColor(Palette.BLACK);

            for (int n = 0; n < numCircles; n++) {
                cd.drawCircle(xCenter + 2 * radius * n, yCenter, radius);
            }
            numCircles++;
            yCenter += 2 * radius;
        }
        cd.show();
    }
}
```

Circles – Mögliche Lösung 2. Erweiterung

```
import codedraw.*;

public class Circles {
    public static void main(String[] args) {
        ...
        boolean isNextOrange = true;
        while (yCenter + radius <= height) {
            int xCenter = width / 2 - (numCircles - 1) * radius;
            int lineLength = 2 * numCircles * radius;

            ...

            for (int n = 0; n < numCircles; n++) {
                cd.setColor(isNextOrange ? Palette.DARK_ORANGE : Palette.DARK_TURQUOISE);
                cd.fillCircle(xCenter + 2 * radius * n, yCenter, radius);
                isNextOrange = !isNextOrange;
            }
            numCircles++;
            yCenter += 2 * radius;
        }
        cd.show();
    }
}
```

Danke für's mitmachen

Bitte Projekt zippen und in TUWEL hochladen