Eingabe und Ausgabe
Ein erster Überblick

Einführung in die Programmierung 1

Wintersemester 21/22

Überblick

- Ausgabe auf der Konsole
- Eingabe
- Grafische Ausgabe

Ausgabe auf der Konsole

Ausgabe (Wiederholung)

- Ausgabe auf Kommandozeile (Konsole)
  - `System.out.print()`
  - `System.out.println()`
- Kann für unterschiedliche Datentypen verwendet werden

String-Klasse (Wiederholung)

- Klasse String ist in Java vordefiniert
- Ähnlichkeiten zu einfachen Datentypen
  - String – Literale wie z. B. `"TestString"`
  - Es gibt einen Operator: + (und auch +=)
  - Implizite Typkonvertierung

F4 → zu Erklärung

Klassen und Objekte (ein einfacher Ausblick)

Klasse

- Beschreibt die gemeinsamen Eigenschaften und Operationen einer Menge gleichartiger Objekte
- Stellt einen Konstruktionsplan für Objekte dar

Objekt

- Ist eine konkrete Instanz einer Klasse
- Es kann mehrere Objekte von einer Klasse geben

Strings (String-Objekte) erzeugen

- Strings erzeugen (Beispiele)

```
String s1 = "Hello";
String s2 = new String("Hello");
String s3 = new String(s1);
```

Bisher verwendet

Anderes Objekt erzeugen

Anderes Objekt erzeugen

Datentypen in Java

Einfache (primitive) Datentypen

- byte, short, int, long, float, double, char, boolean

Referenztypen

- Klassen
- Interfaces
- Arrays

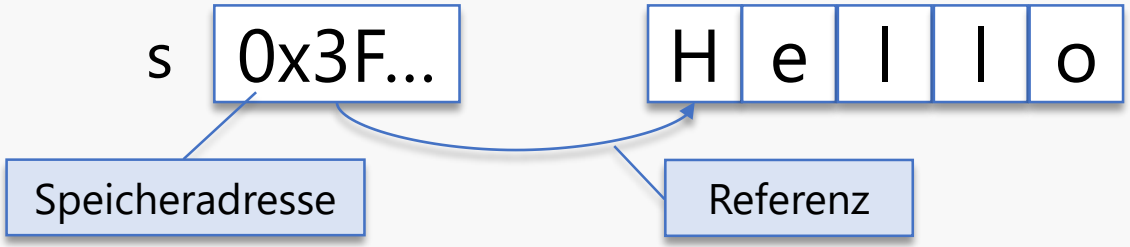
Referenztypen

- Variablen haben in Java einen Datentyp
- Datentyp gibt an, wie die Daten im Speicher interpretiert werden sollen
  - Beispiel: int-Variable mit 32-Bit -> 32 Bit als ganze Zahl auffassen
- Und bei einer Variable eines Referenztyps?
  - Im Speicher der Variable existiert nur eine Referenz auf die Daten
  - Daten liegen an einer anderen Stelle im Speicher

Beispiel mit int und String

- Beispiel `int`-Variable (schematisch)
  - `int x = 1;`

- Beispiel mit String Variable (schematisch)
  - `String s = new String("Hello");`



Besonderheit bei String

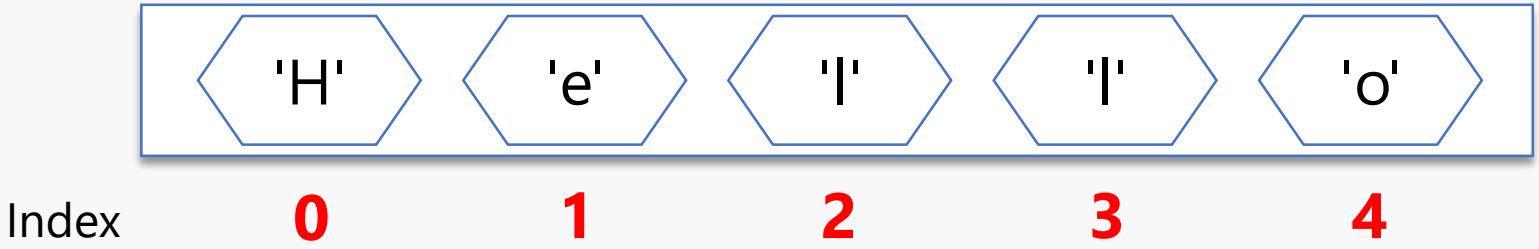
- String-Objekte sind nach dem Anlegen **nicht** mehr veränderbar
  - Bei jeder Konkatenation mit + wird eine neuer String (neues String-Objekt) erzeugt
- String-Literal kommt nur einmal im Programm vor

```
String s1 = "Hello";
String s2 = "Hello";
```

Beide Strings verweisen auf denselben Speicherbereich mit dem Inhalt „Hello“

Struktur eines Strings

- Ein String ist eine Zeichenfolge
- Beispiel String **`"Hello"`**



- Index
  - Jedes Zeichen besitzt einen Index (Position im String)
  - Von **0** bis **(Länge des Strings) - 1**
  - Ermöglicht Zugriff auf einzelne Zeichen

Operationen auf Strings

- + ist eine Ausnahme
  - Keine weitere Operatoren möglich
- Weitere Operationen?
  - Methoden, die auf ein Objekt aufgerufen werden können
- Java-API
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html
  - Aufrufschema
    - Stringvariable gefolgt von einem „.“ und danach kommt ein Methodenname (eventuell Parameter)
  - Beispiel

```
String s = "This is a test string!";
System.out.println(s.charAt(1));
```

Einige hilfreiche String-Methoden

```java
public class StringTest {

    public static void main(final String[] args) {
        String s = "This is a test string!";
        System.out.println(s.length());
        System.out.println(s.isEmpty());
        System.out.println(s.charAt(2));
        System.out.println(s.startsWith("T"));
        System.out.println(s.indexOf('s'));
        System.out.println(s.lastIndexOf('s'));
        System.out.println(s.indexOf('s', 5));
        System.out.println(s.lastIndexOf('s', 14));
        System.out.println(s.indexOf("est"));
        System.out.println(s.replace('i', 'e'));
        System.out.println(s.substring(8));
        System.out.println(s.substring(5, 15));
        System.out.println(s.toLowerCase());
    }

}
```


```
22
false
i
true
3
15
6
12
11
Thes es a test streng!
a test string!
is a test
this is a test string!
```

Eingabe

Motivation

- Variablen immer mit fixen oder zufälligen Werten belegen
  - Nicht praxistauglich
  - Oft möchten wir Daten selbst eingeben und damit die Daten in unserem Programm (teilweise) bestimmen
- Eingabe von Daten
  - Unterschiedliche Möglichkeiten
  - Nachfolgend wird die Eingabe über Kommandozeile mittels Scanner beschrieben

Scanner

- Scanner
  - Einlesen aus dem Eingabefenster über `System.in`
- Aufgabe des Scanners
  - Unterteilt Eingabetext in sogenannte Tokens
- Tokens
  - Primitive Datentypen oder Strings
  - Standardmäßig durch Leerzeichen getrennt
  - Beispiel mit Strings

```
Das ist ein Test
```

4 Tokens durch drei Leerzeichen getrennt

Anlegen einer Scanner-Variable

- Scanner-Klasse benutzen
  - Klasse Scanner muss dafür auch importiert (import) werden
  - Ist nicht automatisch bekannt

```
import java.util.Scanner;

public class ScannerTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        …
    }

}
```

Genaue Bezeichnung von Scanner

Objekt der Klasse Scanner (auf System.in = Eingabe auf der Konsole) anlegen

Beispiel (Einlesen und Ausgeben einer Zahl)

```
import java.util.Scanner;

public class ScannerTest {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        System.out.println(i);
    }

}
```

Methode (Operation) nextInt aufrufen

Beispiel (Zeitumrechnung)

```java
import java.util.Scanner;

public class TimeCalc {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Please insert number of seconds: ");
        int seconds = input.nextInt();
        System.out.println(seconds + " seconds are "
                + (seconds / 3600) + " hour(s), "
                + (seconds % 3600 / 60) + " minutes and "
                + (seconds % 60) + " seconds");
    }

}
```

```
Please insert number of seconds: 1234
1234 seconds are 0 hour(s), 20 minutes and 34 seconds
```

Beispiel (Durchschnitt von drei Zahlen)

```java
import java.util.Scanner;

public class AverageOfThreeNumbers {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int a, b, c;
        System.out.print("First number: ");
        a = input.nextInt();
        System.out.print("Second number: ");
        b = input.nextInt();
        System.out.print("Third number: ");
        c = input.nextInt();
        System.out.println("Average = " + (a + b + c) / 3.0);
    }

}
```

```
First number: 10
Second number: 30
Third number: 40
Average = 26.666666666666668
```

Scanner – hilfreiche Methoden (1)

- `hasNextInt`
  - Überprüfen, ob der nächste Token eine ganze Zahl vom Typ `int` ist
- `nextInt`
  - Nächsten Token als ganze Zahl einlesen
  - Wenn der Token keine ganze Zahl ist, dann wird eine Ausnahme geworfen und das Programm abgebrochen
- Gleiches Schema für einige andere primitive Datentypen
  - Beispiel: `hasNextLong`, `nextLong`, `hasNextFloat`, `nextFloat` ….

Scanner – hilfreiche Methoden (2)

- Den nächsten Token bzw. Zeile als String überprüfen/einlesen
  - `hasNext`, `hasNextLine`
  - `next`, `nextLine`

Java-API

- Wie bekommen wir Informationen zu den einzelnen Methoden?
- Java API für Scanner
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html

Beispiel (Einlesen, verbessert)

```java
import java.util.Scanner;

public class ReadInteger {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please enter a number: ");
        if (scanner.hasNextInt()) {
            int i = scanner.nextInt();
            System.out.println(i);
        } else {
            System.out.println("Input is not a number!");
        }
        scanner.close();
    }

}
```

Schließt den Scanner, wenn der Scanner nicht mehr benötigt wird.
Aber Achtung:
Schließt in diesem Fall auch den verwendeten Stream System.in!
Kann in neueren Java-Versionen auch alternativ implementiert werden!

Beispiel (wiederholtes Einlesen/Ausgeben)

```java
import java.util.Scanner;

public class ReadIntegers {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Please enter an integer (exit with ^D):");
        while (scanner.hasNext()) {
            if (scanner.hasNextInt()) {
                int number = scanner.nextInt();
                System.out.println(number);
            } else {
                String token = scanner.next();
                System.out.println("Wrong Input: " + token);
            }
        }
        System.out.println("Goodbye!");
    }

}
```

Wiederholtes Einlesen

STRG + D – damit wird die Eingabe beendet

Beispiel (Einlesen/Ausgeben von Zeilen)

```java
import java.util.Scanner;

public class Echo {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int lineNumber = 0;
        System.out.println("Please enter 10 lines: ");
        while (lineNumber < 10 && scanner.hasNextLine()) {
            String line = scanner.nextLine();
            System.out.println(lineNumber + ": " + line);
            lineNumber = lineNumber + 1;
        }
        System.out.println("Goodbye!");
    }

}
```

Grafische Ausgabe

Grafische Ausgabe

- In Java möglich, aber komplizierter
- In diesem Kurs – CodeDraw
  - Eigene Entwicklung (basierend auf anderen Bibliotheken für den Unterricht)
- Kurze Übersicht
  - https://github.com/Krassnig/CodeDraw
- Einführung
  - https://github.com/Krassnig/CodeDraw/blob/master/INTRODUCTION.md
- API
  - https://krassnig.github.io/CodeDrawJavaDoc/

CodeDraw benutzen

- Benutzung in IntelliJ
  - `CodeDraw.jar` muss eingebunden werden
    - File -> Project Structure -> Libraries
  - Ist bei unseren TUWEL-Projekten immer dabei
- Muss bei jeder Klasse importiert werden
  - Siehe nachfolgende Beispiele

Grafische Ausgabe

- Eigenes Ausgabefenster
  - Wird am Anfang erzeugt
  - Größe wird beim Erzeugen festgelegt
- Koordinatensystem
  - Punkt (0, 0) befindet sich links oben (siehe Processing)
  - X-Achse nach rechts
  - Y-Achse nach unten
- Befehle für das Zeichnen von Punkten, Linien, Kreisen etc. (siehe Beschreibung der API)
  - Standardfarbe ist Schwarz

Beispiel (Einfache Figuren in CodeDraw)

```
import codedraw.*;

public class DrawTest1 {

    public static void main(String[] args) {
        CodeDraw canvas = new CodeDraw(400, 300);
        canvas.drawSquare(100, 100, 50);
        canvas.setColor(Palette.FOREST_GREEN);
        canvas.drawCircle(200, 200, 50);
        canvas.setColor(Palette.DARK_BLUE);
        canvas.fillRectangle(200, 50, 100, 50);
        canvas.show();
    }

}
```

CodeDraw-Fenster mit der Größe 400 mal 300 erzeugen

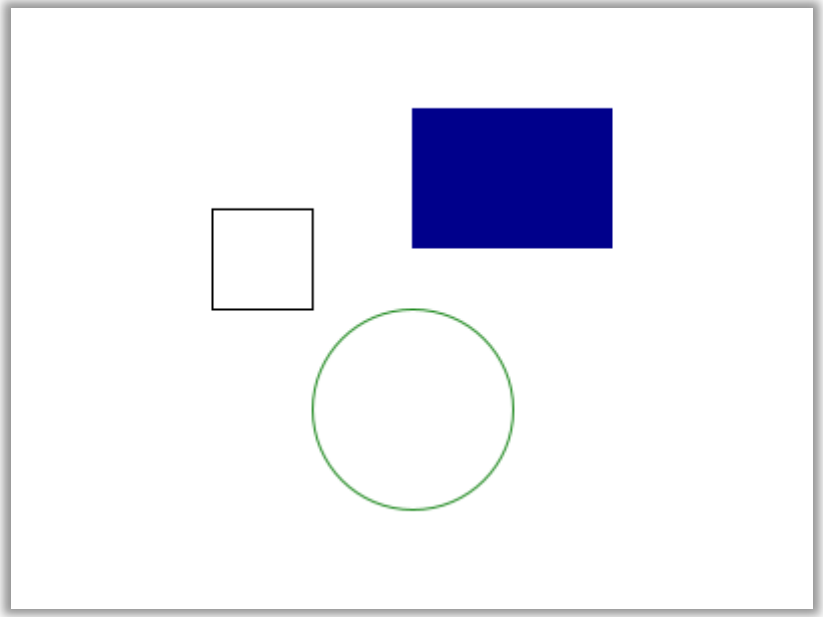
Zeichenfarbe für nachfolgende Operationen setzen

Objekt zeichnen (nicht ausgefüllt)

Objekt zeichnen (ausgefüllt)

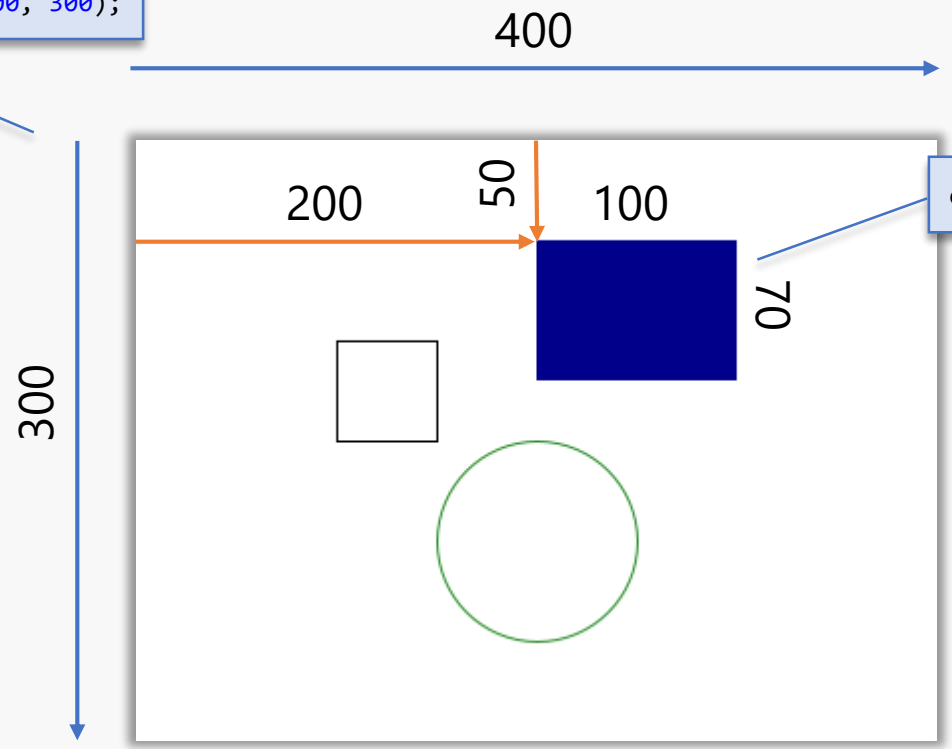
Beispiel (Ausgabe)

```
import codedraw.*;

public class DrawTest1 {

    public static void main(String[] args) {
        CodeDraw canvas = new CodeDraw(400, 300);
        canvas.drawSquare(100, 100, 50);
        canvas.setColor(Palette.FOREST_GREEN);
        canvas.drawCircle(200, 200, 50);
        canvas.setColor(Palette.DARK_BLUE);
        canvas.fillRectangle(200, 50, 100, 70);
        canvas.show();
    }

}
```



Beispiel (Koordinatensystem)

```
CodeDraw canvas = new CodeDraw(400, 300);
```

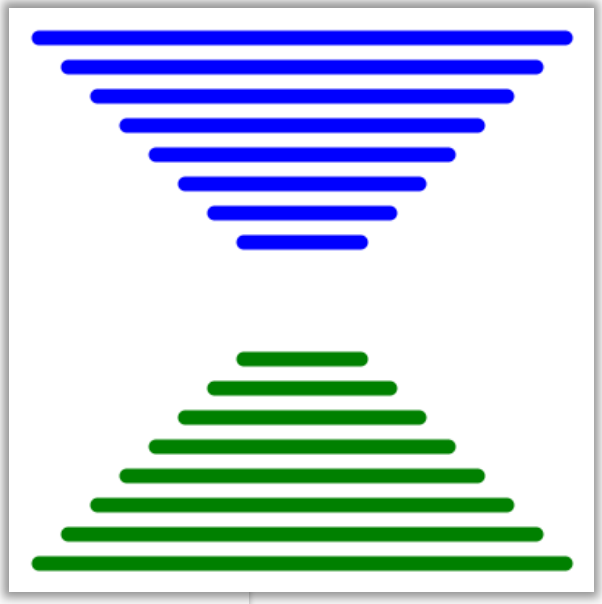



```
canvas.fillRectangle(200, 50, 100, 70);
```

Beispiel (Zeichnen mit while-Schleife)

```
import codedraw.*;

public class PatternTest {

    public static void main(String[] args) {
        int size = 400;
        CodeDraw canvas = new CodeDraw(size, size);
        canvas.setLineWidth(10);
        canvas.setCorner(Corner.ROUND);
        int numberOfLines = 8;
        int step = 20;
        int i = 1;
        while (i <= numberOfLines) {
            int spacing = i * step;
            int oppositeSpacing = size - spacing;
            canvas.setColor(Palette.BLUE);
            canvas.drawLine(spacing, spacing, oppositeSpacing, spacing);
            canvas.setColor(Palette.GREEN);
            canvas.drawLine(spacing, oppositeSpacing, oppositeSpacing, oppositeSpacing);
            i++;
        }
        canvas.show();
    }

}
```

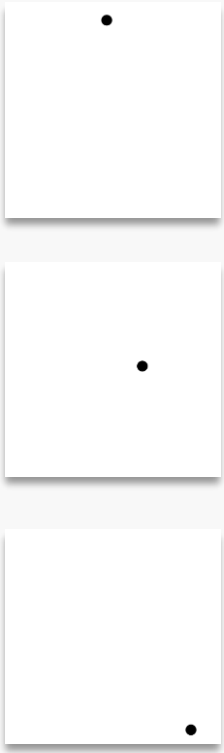
Dickere Linien (Stärke 10) und abgerundete Enden der Linien



Beispiel (mit for-Schleife – Ausblick)

```
import codedraw.*;

public class PatterTest2 {

    public static void main(String[] args) {
        int size = 400;
        CodeDraw canvas = new CodeDraw(size, size);
        canvas.setLineWidth(10);
        canvas.setCorner(Corner.ROUND);
        int numberOfLines = 8;
        int step = 20;
        for (int i = 1; i <= numberOfLines; i++) {
            int spacing = i * step;
            int oppositeSpacing = size - spacing;
            canvas.setColor(Palette.BLUE);
            canvas.drawLine(spacing, spacing, oppositeSpacing, spacing);
            canvas.setColor(Palette.GREEN);
            canvas.drawLine(spacing, oppositeSpacing, oppositeSpacing, oppositeSpacing);
        }
        canvas.show();
    }

}
```

Beispiel (Animation mit Endlosschleife)



```
import codedraw.*;

public class BouncingBall {
    public static void main(String[] args) {
        int width = 400;
        int height = 400;
        CodeDraw canvas = new CodeDraw(width, height);
        double rx = width / 2.0, ry = height / 3.0;
        double vx = 3.0, vy = 5.0;
        double radius = 10;
        while (true) {
            if (rx + vx > width - radius || rx + vx < radius) {
                vx = -vx;
            }
            rx = rx + vx;
            if (ry + vy > height - radius || ry + vy < radius) {
                vy = -vy;
            }
            ry = ry + vy;
            canvas.clear();
            canvas.fillCircle(rx, ry, radius);
            canvas.show(20);
        }
    }
}
```

Endlosschleife

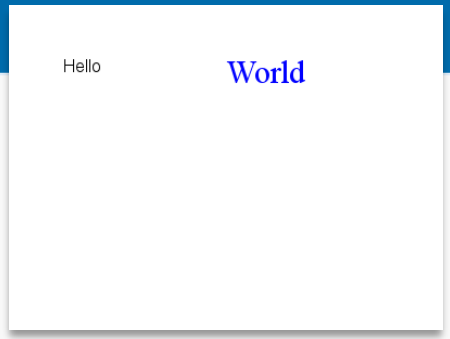
Wenn der „Rand" (rechts bzw. links) erreicht wird, ändert sich die X-Richtung

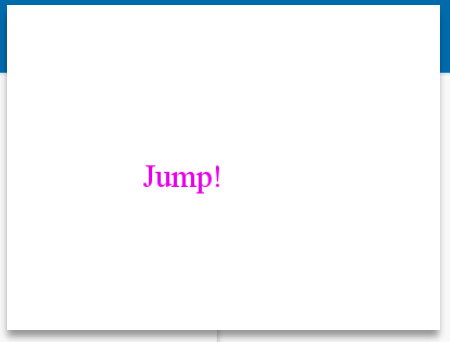
Wenn der „Rand" (unten bzw. oben) erreicht wird, ändert sich die Y-Richtung

Bildschirm löschen

Für 20 Millisekunden den aktuellen Zustand anzeigen

Beispiel mit Text






```java
import codedraw.*;
import codedraw.textformat.*;

public class TextTest {

    public static void main(String[] args) {
        int width = 400;
        int height = 300;
        CodeDraw canvas = new CodeDraw(width, height);
        canvas.drawText(width / 8.0, height / 6.0, "Hello");
        TextFormat format = canvas.getTextFormat();
        format.setFontSize(30);
        format.setFontName("Times New Roman");
        canvas.setColor(Palette.BLUE);
        canvas.drawText(width / 2.0, height / 6.0, "World");
        canvas.show(5000);
        for (int i = 0; i < 256; i++) {
            canvas.clear();
            canvas.setColor(Palette.fromRGB(i, 0, i));
            canvas.drawText(Math.random() * (width / 2.0), Math.random() * (height / 2.0), "Jump!");
            canvas.show(100);
        }

    }

}
```

Text „zeichnen“ (ausgeben)

Textformat-Objekt erzeugen und für die nächste Ausgabe das Format (andere Schrift, andere Schriftgröße) einstellen

Text 5 Sekunden anzeigen

Methode fromRGB in der Klasse Palette aufrufen und damit die Farbe setzen (RGB-Werte vorgeben)

Text wiederholt an anderen Positionen anzeigen

Zusammenfassung

Ausgabe auf der Konsole

Eingabe über Scanner

Grafische Ausgabe mit CodeDraw