Eingabe und Ausgabe
Ein erster Überblick

Einführung in die Programmierung 1

Sommersemester 21

TU Wien

Überblick

- Eingabe
- Ausgabe auf der Konsole
- Grafische Ausgabe

Eingabe

Scanner (Wiederholung)

- Scanner
  - Z. B. Einlesen aus dem Eingabefenster über `System.in`
- Aufgabe des Scanners
  - Unterteilt Eingabetext in sogenannte Tokens
- Tokens
  - Primitive Datentypen oder Strings
  - Standardmäßig durch Leerzeichen getrennt
- Anlegen einer Scanner-Variable
  - Komplexer als bei einfachen Variablen
  - Klasse Scanner muss dafür extra importiert werden

Beispiel (Einlesen einer Zahl)

```java
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);
    }

}
```

Weist darauf hin, dass (die Klasse) Scanner benutzt wird

Methode (Funktion) nextInt aufrufen

Scanner auf System.in anlegen

Scanner – Hilfreiche Methoden

- Hilfreiche Methoden
  - `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` ….
- 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/15/docs/api/java.base/java/util/Scanner.html

Beispiel (Einlesen, verbessert)

```
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)

```
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!");
    }

}
```

Beispiel (ein kleines Ratespiel)

- Ratespiel
  - Computer wählt zufällig eine Zahl im Bereich von 1 bis 100 aus
  - Person kann eine Zahl raten
    - Eingabe über `System.in`
    - Wenn falsch
      - Ausgabe, ob die Eingabe zu klein, zu groß oder keine korrekte Zahl ist
    - Wenn richtig
      - Ausgabe einer Gewinnmeldung und Ende des Programms
- Code in TUWEL
  - `GuessingGame.java`

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
- Java-API
  - https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/String.html

String erzeugen

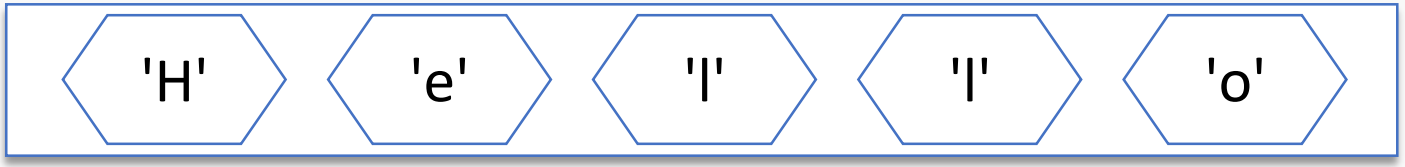
- Strings erzeugen (Beispiele)

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

- Besonderheiten
  - String-Objekte sind nach dem Anlegen **nicht** mehr veränderbar
  - Bei jeder Konkatenation mit + wird eine neuer String (neues String-Objekt) erzeugt

Struktur

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



Index **0** **1** **2** **3** **4**

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

Einige hilfreiche String-Methoden

```
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!
```

Grafische Ausgabe

Grafische Ausgabe

- In Java möglich, aber komplizierter
- Für diesen Kurs
  - Einfache offene Bibliothek vom Web
- stdlib.jar von http://introcs.cs.princeton.edu/java/stdlib/
  - Enthält unterschiedliche Klassen für die Ein- und Ausgabe
  - Einfache Beispiele (ähnlich zu Java-API)
    - StdIn für das Einlesen von der Kommandozeile
    - StdOut für das Ausgeben auf der Kommandozeile
  - `StdDraw` für die grafische Ausgabe
    - http://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html

stdlib.jar benutzen

- Benutzung in IntelliJ
  - `stdlib.jar` muss eingebunden werden
    - File -> Project Structure -> Libraries
  - Ist bei unseren TUWEL-Projekten immer dabei

Grafische Ausgabe

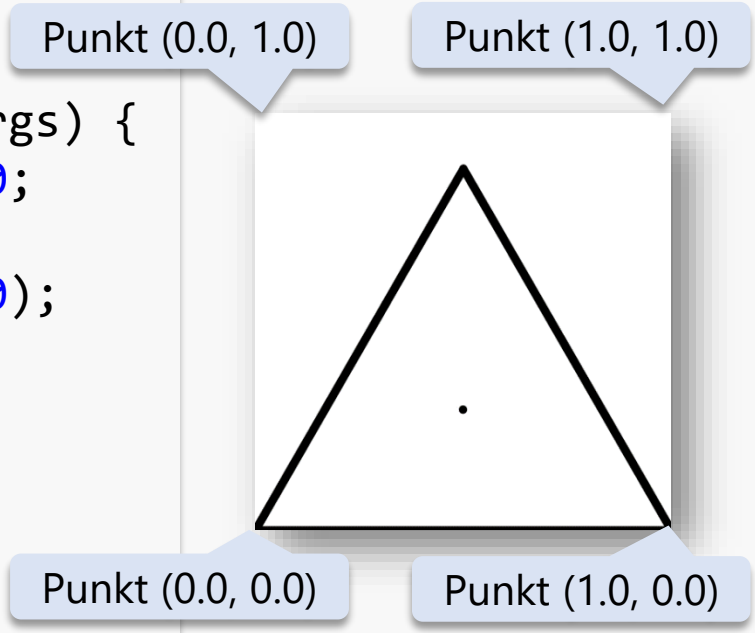
- Eigenes Ausgabefenster
- Standardmäßig 512 × 512 Pixel
  - Koordinatensystem wie in der Mathematik
  - Von 0.0 bis 1.0 (sowohl auf der X- als auch auf der Y-Achse)
- Befehle für das Zeichnen von Punkten, Linien, Kreisen etc. (siehe Beschreibung der API)
- Form für Methodenaufruf
  - `StdDraw.`*`method`*`(..)`

Beispiel (Dreieck mit Punkt)

```
public class DrawTest1 {

    public static void main(String[] args) {
        double t = Math.sqrt(3.0) / 2.0;
        StdDraw.setPenRadius(0.02);
        StdDraw.line(0.0, 0.0, 1.0, 0.0);
        StdDraw.line(1.0, 0.0, 0.5, t);
        StdDraw.line(0.5, t, 0.0, 0.0);
        StdDraw.point(0.5, t / 3.0);
    }

}
```

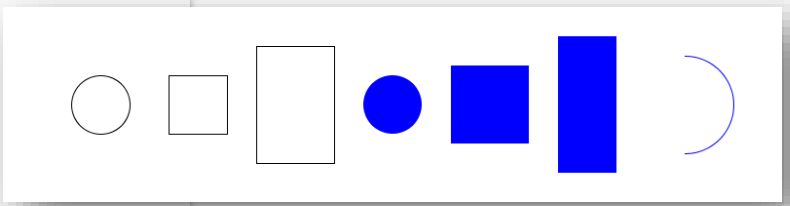


Beispiel (Figuren, angepasste Fenstergröße)

```
public class DrawTest2 {

    public static void main(String[] args) {
        int width = 800;
        int height = 200;
        double middle = height / 2.0;
        StdDraw.setCanvasSize(width, height);
        StdDraw.setXscale(0, width);
        StdDraw.setYscale(0, height);
        StdDraw.circle(100, middle, 30);
        StdDraw.square(200, middle, 30);
        StdDraw.rectangle(300, middle, 40, 60);
        StdDraw.setPenColor(StdDraw.BLUE);
        StdDraw.filledCircle(400, middle, 30);
        StdDraw.filledSquare(500, middle, 40);
        StdDraw.filledRectangle(600, middle, 30, 70);
        StdDraw.arc(700, middle, 50, 270, 90);
    }

}
```



Anpassen der Fenstergröße

Getrenntes Anpassen des X- und Y-Bereichs

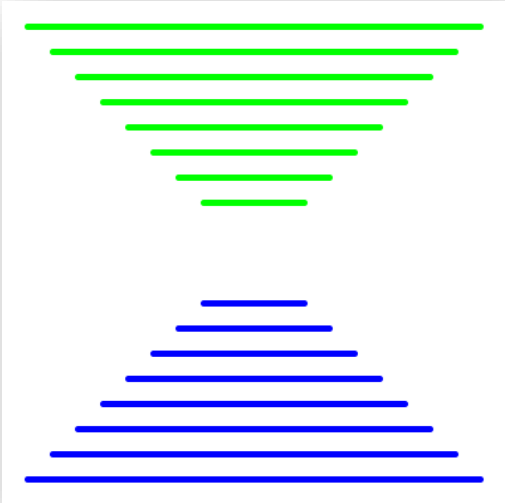
Beispiel (Ausblick auf Schleifen)

```
public class PatternTest {

    public static void main(String[] args) {
        int size = 400;
        StdDraw.setCanvasSize(size, size);
        StdDraw.setScale(0, size);
        StdDraw.setPenRadius(0.01);
        int c = 8;
        int step = 20;
        for (int i = 1; i <= c; i++) {
            int spacing = i * step;
            int oppositeSpacing = size - spacing;
            StdDraw.setPenColor(StdDraw.BLUE);
            StdDraw.line(spacing, spacing, oppositeSpacing, spacing);
            StdDraw.setPenColor(StdDraw.GREEN);
            StdDraw.line(spacing, oppositeSpacing, oppositeSpacing, oppositeSpacing);
        }
    }

}
```

Gemeinsames Anpassen des X- und Y-Bereichs



c-mal den Programmcode mit unterschiedlichen Werten ausführen (Linien werden abwechselnd gezeichnet !)

Beispiel (Animation)

```
public class BouncingBall {
    public static void main(String[] args) {
        StdDraw.setXscale(-1.0, 1.0);
        StdDraw.setYscale(-1.0, 1.0);
        StdDraw.enableDoubleBuffering();
        double rx = 0.480, ry = 0.860;
        double vx = 0.015, vy = 0.023;
        double radius = 0.05;
        while (true) {
            if (Math.abs(rx + vx) > 1.0 - radius) vx = -vx;
            if (Math.abs(ry + vy) > 1.0 - radius) vy = -vy;
            rx = rx + vx;
            ry = ry + vy;
            StdDraw.clear();
            StdDraw.filledCircle(rx, ry, radius);
            StdDraw.show();
            StdDraw.pause(20);
        }
    }
}
```

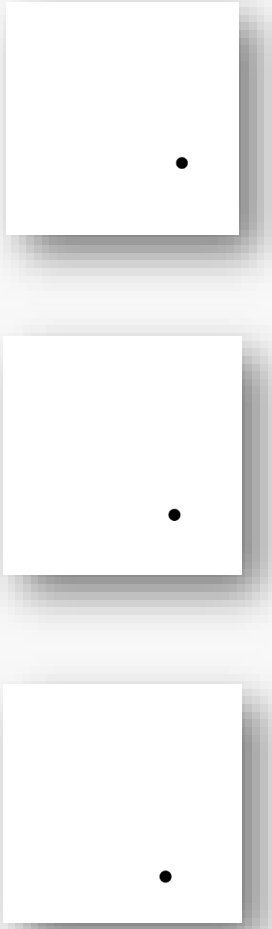
Doppelpufferung = es wird zunächst nur in einen Puffer gezeichnet

Endlosschleife

Bildschirm löschen

Verdeckt (im Puffer) zeichnen

Puffer am Bildschirm anzeigen



Zusammenfassung

- Eingabe
  - Scanner
  - Größeres Beispiel (Ratespiel)
- Ausgabe
  - String-Klasse
  - Ausgewählte String-Methoden
- Grafische Ausgabe
  - Grundlagen der StdDraw-Klasse
  - Animation in StdDraw