Übungsstunde 6

Einführung in die Programmierung

Was machen wir heute?

Nachbesprechung

Theorie: Klassen & Objekte

Theorie-Prüfungsaufgaben :)

Vorbesprechung

(Extra-Aufgaben)

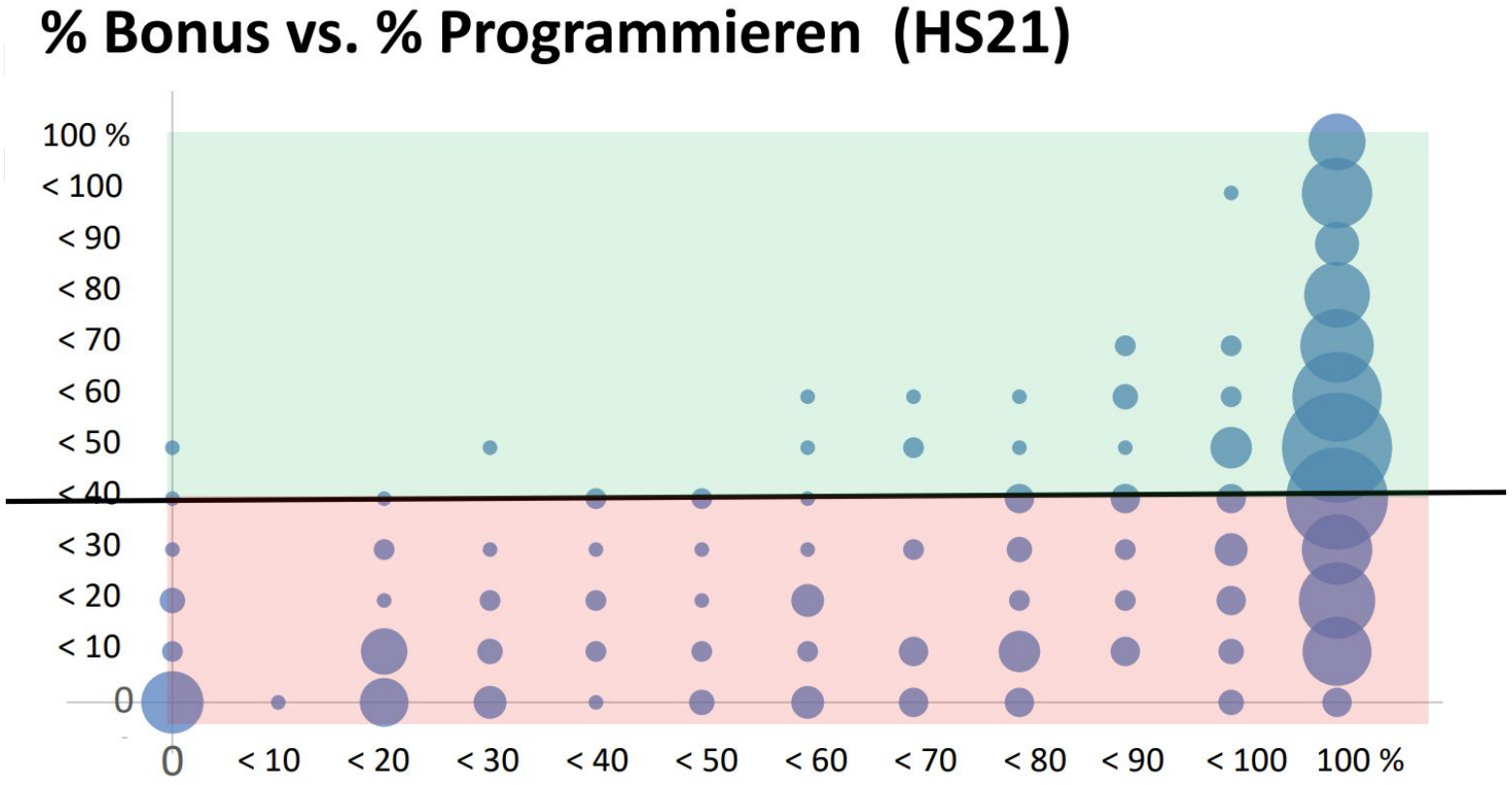
Kahoot

Allgemeines

E-Mails bitte an beide TAs

Nicht mehr so viele Abgaben wie auch schon

Allgemeines



Fragen zur Bewertung Bonusaufgabe 4

Nachbesprechung Übung 5

Aufgabe 1: Wörter Raten

```
Tipp? e
Das Wort enthält nicht "e"!
Tipp? a
Das Wort endet mit "a"!
Tipp? j
Das Wort beginnt mit "j"!
Tipp? v
Das Wort enthält "v"!
Tipp? java
Das Wort ist "java"!
Glückwunsch, du hast nur 5 Versuche benötigt!
```

```
String hinweis = "dummy";

while(!hinweis.equals("ist")) {
    …
    hinweis = hinweis(wort, tipp);
    …
};
```

do-while benötigt keinen Dummy-Wert für **hinweis**

```
String hinweis;

do {
    …
    hinweis = hinweis(wort, tipp);
    …
} while(!hinweis.equals("ist"));
```

Aufgabe 2: Datenanalyse

In dieser Aufgabe werden Sie die Körpergrössen einiger Personen analysieren, welche für eine Studie[1] in Kalifornien erhoben wurden. Im Programm "DatenAnalyse.java" sind schon ein paar (leere) Methoden zu Ihrer Hilfe vorgegeben.

```
[140,143)
[143,146)
[146,149) |
[149,152) ||||
[152,155) ||||||||||||
[155,158) ||||||||||||||||||||
[158,161) |||||||||||||||||||||||||||||||||||||||||
[161,164) |||||||||||||||||||||||||||||||||||||||||||||||||||
[164,167) |||||||||||||||||||||||||||||||||||||||||
[167,170) |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
[170,173) |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
[173,176) |||||||||||||||||||||||||||||||||||||||||||||||||||||||
[176,179) |||||||||||||||||||||||||||||||||||||||||||||||||||||
[179,182) |||||||||||||||||||||||||||||||||||||||
[182,185) |||||||||||||||||||||||||
[185,188) ||||||||||||||
[188,191) ||||||||||||||||||||
[191,194) ||||||
[194,197)
[197,200) ||
```

Aufgabe 3: Bonusaufgabe

Erreichte Punktzahl nach der Korrektur direkt per Git

Lösungsvorschlag von Studenten

Aufgabe 4: Schweizerfahne (GUI)

- Instanziere die Window-Klasse und benutze die Methoden setColor() und fillRect()
- RGB-Farbwerte sind zwischen 0 und 255
- Die (x,y)-Koordinaten beginnen in der linken oberen Ecke mit (0,0)

Das Verhältnis von Breite und Länge der Kreuzarme beträgt also 6:7, von Breite und Gesamtlänge des Kreuzbalkens 6:20.
Das Grössenverhältnis des Kreuzes zum Quadrat ist nicht definiert.
(wikipedia)

```
Scanner s = new Scanner(System.in);
int size = s.nextInt(); // input the flag size

Window window = new Window("Fahne", size, size);

window.setColor(255, 0, 0); // red
window.fillRect(0, 0, size, size);

window.setColor(255, 255, 255); // white
window.fillRect(0.2 * size, 0.4 * size, 0.6 * size, 0.2 * size);
window.fillRect(0.4 * size, 0.2 * size, 0.2 * size, 0.6 * size);

window.open();
window.waitUntilClosed();
```

wiederverwendbar

Aufgabe 5: Chaos Game

```
public static void main(String[] args) {
      int size = 800;
      int stepsPerIter = 50;
      double[] cornersX = {size/2, 50, size-50};
      double[] cornersY = {100, size-100, size-100};
      double x = Math.random() * size;
      double y = Math.random() * size;

      Window window = new Window("Chaos Game", size, size);
      window.open();
      while(window.isOpen()) {
            for(int i = 0; i < stepsPerIter; i++) {
                  int index = (int) (Math.random() * 3);
                  x = (x + cornersX[index]) / 2;
                  y = (y + cornersY[index]) / 2;
                  window.fillRect(x, y, 1, 1);
            }
            window.refresh();
      }
}
```

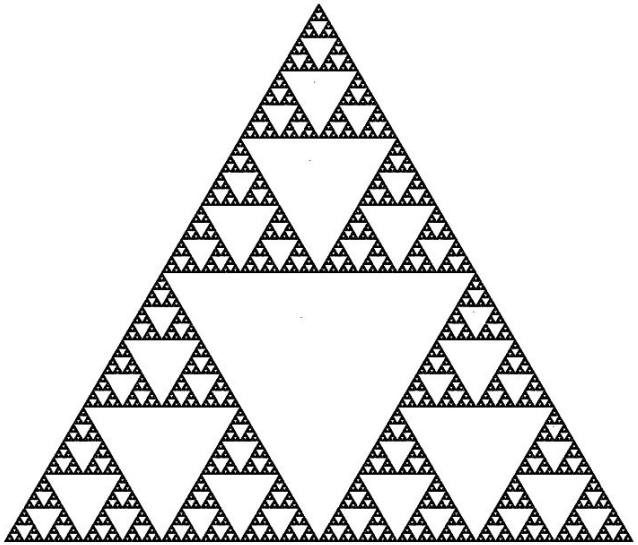
Eckpunkte des Dreiecks

x,y-Koordinaten eines zufälligen Punktes im Fenster

Wahl eines Eckpunktes

Berechne die Koordinaten des Mittelpunkts

Aufgabe 5: Chaos Game



Sierpinski-Dreieck

Wacław Sierpiński

Aufgabe 6: EBNF

In dieser Aufgabe erstellen Sie erneut verschiedene EBNF-Beschreibungen. Speichern Sie diese wie gewohnt in der Text-Datei "EBNF.txt", welche sich in Ihrem "u04"-Ordner, bzw. im "U04 <*N-ETHZ-Account*>"-Projekt befindet. Sie können die Datei direkt in Eclipse bearbeiten.

1. Erstellen Sie eine Beschreibung `<pyramid>`, welche alle Zahlen zulässt, in denen die Ziffern zuerst strikt auf- und dann strikt absteigen. Beispiele sind: 14, 121, 1341.
   Sie dürfen annehmen, dass das leere Wort auch zugelassen wird. (Als Challenge können Sie probieren, das leere Wort auszuschliessen.)

2. Erstellen Sie eine Beschreibung `<digitsum>`, welche alle natürlichen Zahlen als Symbole zulässt, deren Quersumme eine gerade Zahl ist.

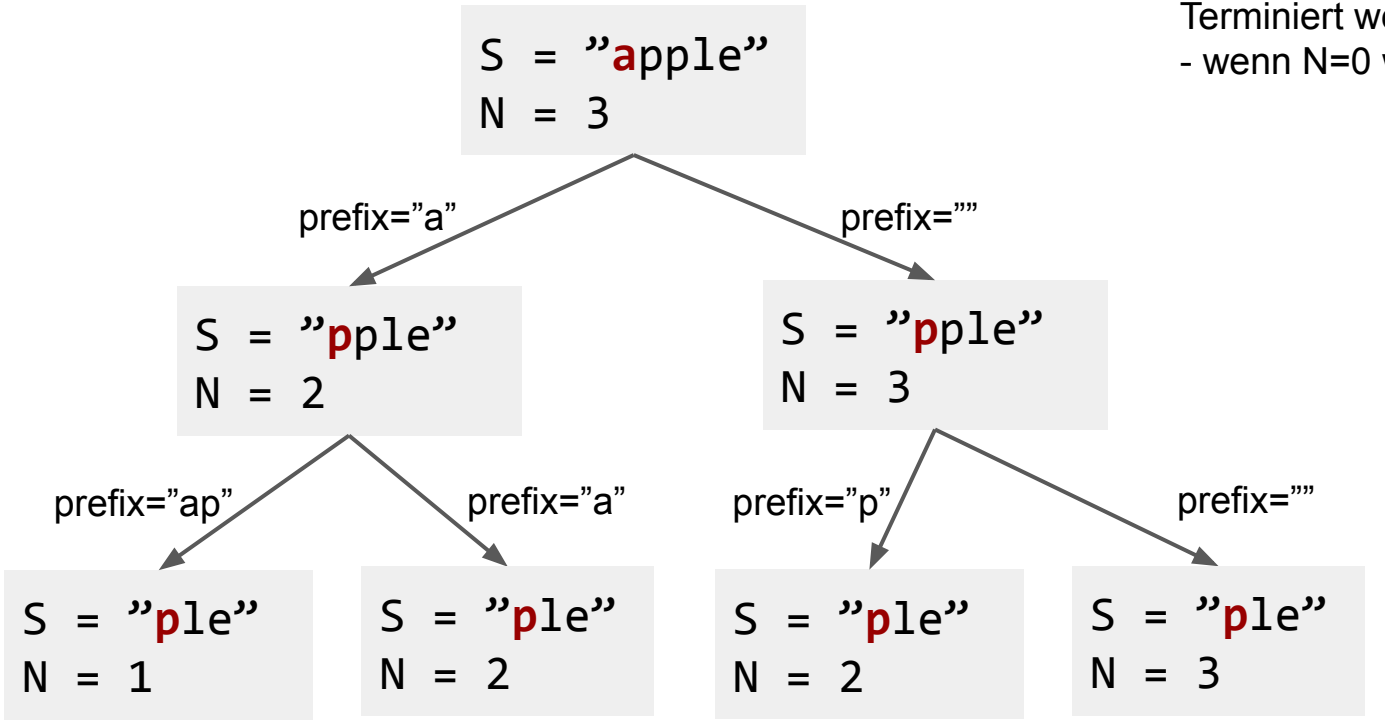
3. Erstellen Sie eine Beschreibung `<xyz>`, welche alle Wörter zulässt, die aus X, Y und Z bestehen und bei welchen jedes X mindestens ein Y im Teilwort links und rechts von sich hat. Beispiele sind: Z, YXY, YXXY, ZYXYY.

4. Erstellen Sie eine Beschreibung `<term>`, welche alle wohlgeformten arithmetischen Terme, besthend aus ganzen Zahlen, Variablen (x, y, z), Addition und Klammern zulässt. Geklammerte Terme müssen mindestens eine Addition enthalten. Beispiele sind: 1 + 4, (1 + 4), 1 + (3 + 4), (1 + 1) + x + 5.

Aufgabe 7: Teilfolgen

```java
public static void subSequences(String prefix, String s, int n) {
        if (n == 0) {
                // "prefix" ist eine Teilfolge der gewünschten Länge
                System.out.println(prefix);
        } else if (!s.isEmpty()) {
                // Man nimmt den ersten Buchstaben...
                subSequences(prefix + s.charAt(0), s.substring(1), n-1);
                // ... oder man nimmt den Buchstaben nicht
                subSequences(prefix, s.substring(1), n);
        }
}
```

Rekursion: Teilprobleme identifizieren



Terminiert wenn:
- wenn N=0 wird

Theorie-Input

Themenübersicht

EBNF

Java

- Methoden
- Typen und Variablen
- if-else (Verzweigungen)
- Schleifen
- Arrays

Hoare-Tripel, Pre- & Postconditions
In- & Output, GUI

Jetzt: Klassen und Objekte

Klassen und Objekte

Klassen sind wie ein eigener Datentyp

- Können Attribute speichern (Zustand)
- Haben Methoden (Verhalten)

Objekte sind eine Instanz einer Klasse

=> Klasse gibt die Struktur der Objekte vor

=> die Objekte haben dann die eigentlichen Daten

Klassen und Objekte

Beispiel:

Klasse Auto
Hat x Sitzplätze
Kann Fahren

Objekt Lamborghini
Hat 2 Sitzplätze
Kann Fahren

Objekt Audi
Hat 4 Sitzplätze
Kann Fahren

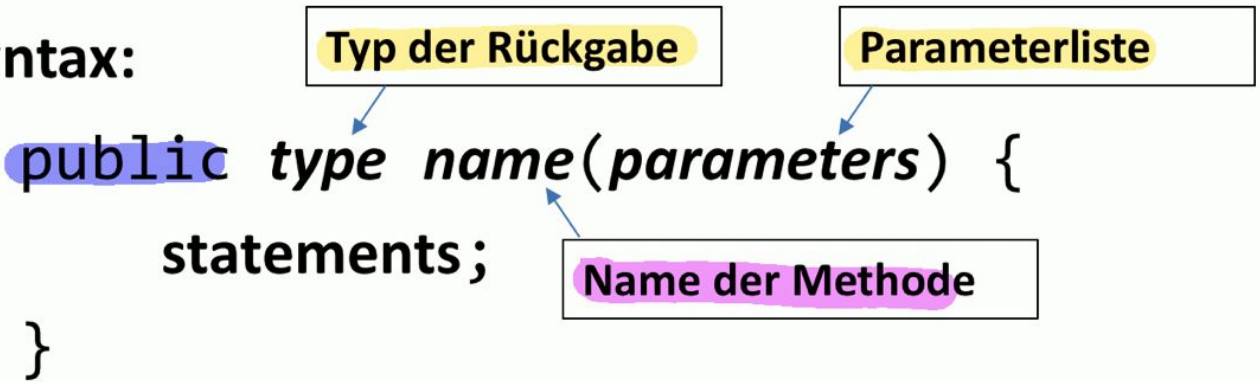
Instanz-Methoden

Klasse Auto
Hat x Sitzplätze
→ **Kann Fahren**

Klassen sollten Zustand und Verhalten kombinieren

Methoden für Exemplare

- Eine Methode (auch «object method») existiert innerhalb jedes Objekt(exemplar)s einer Klasse und beschreibt das Verhalten eines Objektes.
- Syntax:



  - Selbe Syntax wie bei den bisher bekannten Methoden aber *ohne* das Keyword `static`

Sichtbarkeit von Attributen und Methoden

private: Nur innerhalb einer Klasse, keine anderer Klasse/Methode kann zugreifen

public: Überall und jeder

default (oder gar nichts): nur innerhalb eines Package (z.B. src, test)

protected: mehr als default, weniger als public (weniger relevant)

Sichtbarkeit von Attributen und Methoden

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Shadowing

```
public class Point {
     int x;

     public void method1(double x) {
        // Need to set attribute x!
        x = …
     }
}
```

Shadowing & this-Keyword

Mit `this` können Sie auf das Attribut zugreifen

```
public class Point {
    int x;

    public void method1(double x) {
      // Need to set attribute x!
      this.x = (int)(x/2);
    }
}
```

Shadowing & this-Keyword

Verdecken von Variablen («shadowing»)

Poll

```
public class Point {
   int x;
   int y;

   public void method1() {
     // D: x bezieht sich auf Attribut x
   }
   public void method2(int x) {
     // E: x bezieht sich auf Parameter x
   }
   public void method3() {
     int x;
     // F: x bezieht sich auf Variable x
   }
}
```

Konstruktoren

Klasse (“Template”) geschrieben

→ Objekte erstellen

new Operator

Auto Klasse

```
public class Auto {

	int sitze;
	String marke;

	public void fahren() {
		System.out.println("Ein “ + marke + “ mit “ + sitze + “Sitzen fährt.");
	}

}
```

```
public class Main {

    public static void main(String[] args) {

        Auto lambo = new Auto();

        lambo.sitze = 2;

        lambo.marke = “Lamborghini”;

        Auto audi = new Auto();

        lambo.fahren();

    }

}
```

Ein Lamborghini mit 2 Sitzen fährt.

```
public class Auto {

	int sitze;
	String marke;

	public Auto(int sitze, String marke) {
			this.sitze = sitze;
			this.marke = marke;
	}

	public void fahren() {
		System.out.println("Ein “ + marke + “ mit “ + sitze + “Sitzen fährt.");
	}

}
```

```
public class Main {

    public static void main(String[] args) {

        Auto lambo = new Auto(2, “Lamborghini”);

        Auto audi = new Auto();

        lambo.fahren();

    }

}
```

Ein Lamborghini mit 2 Sitzen fährt.

```
public class Auto {

    int sitze;
    String marke;

    public Auto() {}

    public Auto(int sitze, String marke) {
        this.sitze = sitze;
        this.marke = marke;
    }

    public void fahren() {
        System.out.println("Ein “ + marke + “ mit “ + sitze + “Sitzen fährt.");
    }
```

```
public class Main {

    public static void main(String[] args) {

        Auto lambo = new Auto(2, “Lamborghini”);

        Auto audi = new Auto();

        lambo.fahren();

    }

}
```

Ein Lamborghini mit 2 Sitzen fährt.
Ein null mit 0 Sitzen fährt.

Tipp nebenbei: LeetCode

Prüfungsaufgabe
bis 08:55

Vorbesprechung Übung 6

Aufgabe 1: Prefixkonstruktion (Bonus!)

Gegeben seien zwei Strings `s` und `t` und ein Integer `n` mit $n \geq 0$. Schreiben Sie ein Programm, das zurückgibt, ob `s` eine Konkatenation von maximal `n` vielen Prefixen von `t` ist.
Beispiele:

- `s = "abcababc"`, `t = "abc"`, `n = 4`: Das Programm sollte `true` zurückgeben, da `"abc"` und `"ab"` Prefixe von `t` sind und `s` eine Konkatenation von `"abc"`, `"ab"`, `"abc"` ist.
- `s = "abcbcabc"`, `t = "abc"`, `n = 4`: Das Programm sollte `false` zurückgeben, da `"bc"` kein Prefix von `t` ist.
- `s = "abab"`, `t = "abac"`, `n = 2`: Das Programm sollte `true` zurückgeben, da `"ab"` in ein Prefix von `t` ist und `s` eine Konkatenation von `"ab"`, `"ab"` ist.

Implementieren Sie die Methode `isPrefixConstruction(String s, String t, int n)` in der Klasse `PrefixConstruction`. Die Methode hat drei Argumente: die beiden Strings `s` und `t` und der Integer `n`. Sie dürfen davon ausgehen, dass der Integer grösser oder gleich 0 ist. In der Datei "PrefixConstructionTest.java" finden Sie Tests. **Tipp:** Lösen Sie die Aufgabe rekursiv.

Aufgabe 2: Datenanalyse mit Personen

In der letzten Übung haben Sie Körpergrössen analysiert. In dieser Aufgabe werten Sie einen Personen-Datensatz mit mehreren "Spalten" aus. Die Spalten enthalten Werte für Gewicht, Alter, Geschlecht, usw. Um einfach mit diesen Daten zu arbeiten, entwerfen Sie eine Klasse `Person`, welche alle Eigenschaften einer Person als Felder enthält. Weiter schreiben Sie ein Programm "PersonenAnalyse.java", welche die Personendaten aus einer Datei einliest und diese auswertet.

body.dat.txt

```
506
42.9 17.7 28.0 21  65.6 174.0 1
43.7 16.9 30.8 23  71.8 175.3 1
40.1 20.9 31.7 28  80.7 193.5 1
44.3 18.4 28.2 23  72.6 186.5 1
42.5 21.5 29.4 22  78.8 187.2 1
...
```

Spalte	Beschreibung	Java-Typ
0	Schulterbreite in cm	`double`
1	Brusttiefe in cm	`double`
2	Brustbreite in cm	`double`
3	Alter in Jahren	`int`
4	Gewicht in kg	`double`
5	Grösse in cm	`double`
6	Geschlecht (1: männlich, 0: weiblich)	`boolean`

Tabelle 1: Datenbeschreibung von "body.dat.txt"

Einlesen mit:

```
scanner.nextInt()
scanner.nextDouble()
```

a) Vervollständigen Sie die Klasse `Person` (in der Datei "Person.java"), indem Sie sie mit Feldern und Methoden zu ergänzen. Fügen Sie für jede Spalte im Datensatz ein Feld mit dem entsprechenden Java-Typ hinzu und implementieren Sie den (schon vorhandenen) Konstruktor so, dass er diese Felder initialisiert.

Spalte	Beschreibung	Java-Typ
0	Schulterbreite in cm	`double`
1	Brusttiefe in cm	`double`
2	Brustbreite in cm	`double`
3	Alter in Jahren	`int`
4	Gewicht in kg	`double`
5	Grösse in cm	`double`
6	Geschlecht (1: männlich, 0: weiblich)	`boolean`

Tabelle 1: Datenbeschreibung von "body.dat.txt"

```
public class Person {



    /**
     * Erstellt eine neue Person mit den gegebenen Werten
     * für die Eigenschaften. */
    Person(int alter, double gewicht, double groesse,
            boolean istMaennlich, double schulterBreite,
            double brustTiefe, double brustBreite) {


    }
}
```

d) Glücklicherweise sind die Daten ideal um ein weiteres wichtiges Problem zu lösen: Sie ermöglichen uns, zu jeder Person einen geeigneten Trainingspartner fürs Fitnessstudio zu ermitteln. Da die Geräte im Studio immer auf die Person eingestellt sein müssen, sollten Trainingspartner ähnliche "Dimensionen" aufweisen. Wir beschreiben die Ähnlichkeit von zwei Personen p_1 und p_2 betreffend dieses Kriteriums als *Partner-Qualität* $Q(p_1, p_2)$, die wir wie folgt definieren:

$$
\begin{aligned}
\mathit{grössenDiff}(p_1, p_2) &= \mathit{grösse}(p_1) - \mathit{grösse}(p_2) \\
\mathit{brustDiff}(p_1, p_2) &= \mathit{brustTiefe}(p_1) \cdot \mathit{brustBreite}(p_1) - \mathit{brustTiefe}(p_2) \cdot \mathit{brustBreite}(p_2) \\
\mathit{schulterDiff}(p_1, p_2) &= \mathit{schulterBreite}(p_1) - \mathit{schulterBreite}(p_2) \\
Q(p_1, p_2) &= \frac{1}{1 + \mathit{grössenDiff}(p_1, p_2)^2 + \frac{\mathit{abs}(\mathit{brustDiff}(p_1, p_2))}{5} + \frac{\mathit{schulterDiff}(p_1, p_2)^2}{2}}
\end{aligned}
$$

ii) Schreiben Sie nun eine Methode `druckeGuteTrainingsPartner()`, welche die Qualität aller möglichen Paare berechnet. Sofern die Qualität eines Paares 0.8 übersteigt, sollen die Partner-Qualität, sowie die Beschreibungen und Gewichtsklassen beider Personen ausgegeben werden. Achten Sie darauf, dass kein Paar doppelt ausgegeben wird (eine bestimmte Person darf jedoch in mehreren Paaren auftreten). Ein Beispiel eines solchen Paares ist:

```
Person (m, 21 Jahre, 177.8 cm, 79.5 kg), übergewichtig
Person (m, 19 Jahre, 177.8 cm, 76.6 kg), normalgewichtig
Qualität: 1.0
```

Kennen Sie eine bessere Formel für Partner-Qualität?

Aufgabe 3: Black-Box Testing

1	2	3	4	5	6	7

```
BlackBox.rotateArray(values, 2);
```

6	7	1	2	3	4	5

Aufgabe 3: Black-Box Testing

```java
@Test
public void testRotateArray() {
    int[] values = new int[] { 1, 2, 3, 4 , 5 ,6, 7 };
    int[] expected = new int[] { 6, 7, 1, 2, 3, 4, 5 };
    BlackBox.rotateArray(values, 2);
    assertArrayEquals(expected, values);
}

@Test
public void testRotateArray2() {
    int[] values = new int[] { 1, 2, 3, 4 , 5 ,6, 7 };
    int[] expected = new int[] { 3, 4, 5, 6, 7, 1, 2 };
    BlackBox.rotateArray(values, -2);
    assertArrayEquals(expected, values);
}
```

Negative bedeutet nach links

Aufgabe 4: Loop Invariante

Gegeben sind die Precondition und Postcondition für das folgende Programm

```
public int compute(int n) {
  // Precondition:  n >= 0
  int x;
  int res;

  x = 0;
  res = x;

  // Loop Invariante:
  while (x <= n) {
    res = res + x;
    x = x + 1;
  }
  // Postcondition:  res == ((n + 1) * n) / 2
  return res;
}
```

Schreiben Sie die Loop Invariante in die Datei “LoopInvariante.txt”.

Loop Invariante zum Vorlösen

Beispiel

```
public int compute(int n) {

  // Precondition: n >= 0 && n % 2 == 0

  int i;
  int res;

  i = 0;
  res = 1;

  // Loop Invariante: ??

  while (i <= n) {
    res = 1 - res;
    i = i + 1;
  }

  // Postcondition:  res == 0
  return res;
}
```

Aufgabe 5: Analoge Uhr

Zeit ---> Winkel

z.B. 23:10:20

Wie kann man 20 Sekunden in einen Winkel umrechnen?

$\alpha = 20 / 60 * 2 \pi$ (rad)

Achtung: eine Uhr läuft im Uhrzeigersinn

Math Class

cos

```
public static double cos(double a)
```

Returns the trigonometric cosine of an angle. Special cases:

- If the argument is NaN or an infinity, then the result is NaN.

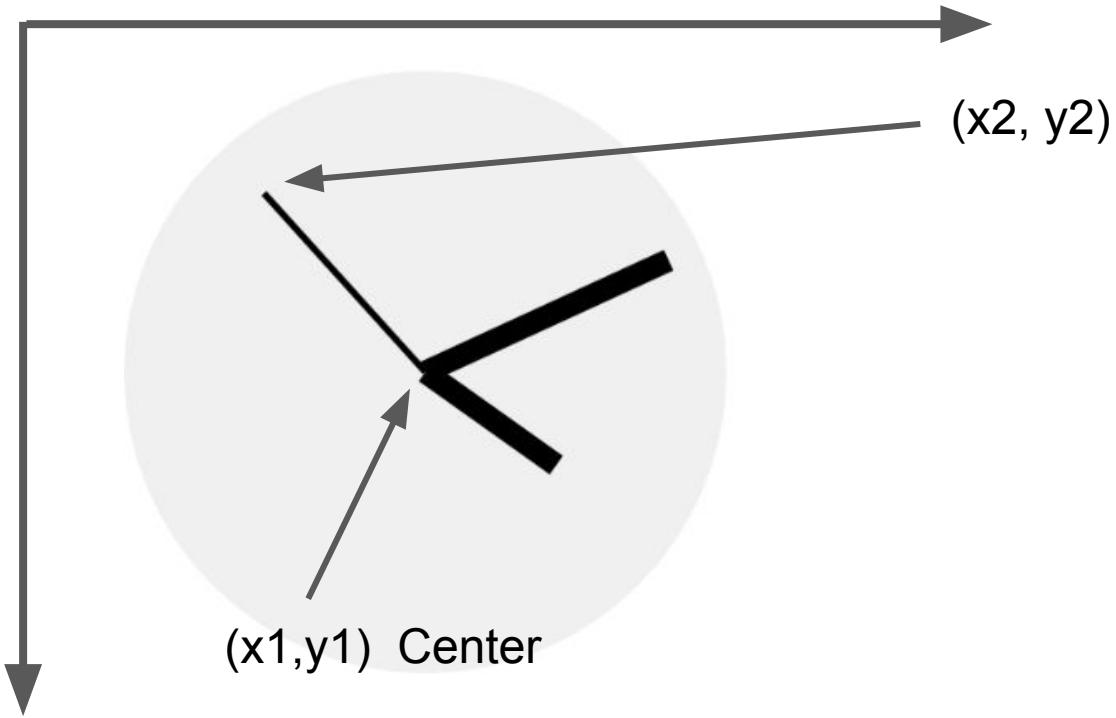
Parameters:

`a` - an angle, in radians.

Returns:

the cosine of the argument.

```
window.drawLine(double x1, double y1,
                double x2, double y2)
```



System.currentTimeMillis()

Millisekunden seit 00:00:00, 01.01.1970

```
System.out.println(System.currentTimeMillis);
```

→ 1507981421

Wie kann man mit dieser Zahl die aktuelle Zeit ausrechnen?

Division und Modulo: `sec = (msec / 1000) % 60`

Kahoot!

Zusatzübungen → auf Webseite

Close Enough

Schreiben Sie eine Methode `closeEnough` die zwei Arrays `a, b` des Typs `int` und einen `int` Wert `eps` als Parameter übernimmt. Die Methode liefert **`true`** wenn für alle Elemente von `a` und `b` gilt:

$$abs(a[i] - b[i]) \leq eps$$

Wenn das nicht der Fall ist, dann ist der Return-Wert **`false`**.

Sorted Int

Schreiben Sie eine Methode `sortedInt,` die zwei **`int`** Parameter `a`, `b` entgegen nimmt und sie der Grösse nach sortiert (aufsteigend) zurück gibt. Also:

```
? ← sortedInt(a, b);
```

- Kann `sortedInt a` und `b` modifizieren?
- Was muss der Return-Wert sein?

Rechenübungen

- Schreiben Sie ein Programm das in der 1. Klasse zum Rechnenueben verwendet werden kann.
  - Es sollen Rechnungen mit 2-5 Zahlen zwischen 1 und 10 geloest werden.
  - Es gibt 1 Punkt fuer eine korrekte Antwort, 0 fuer eine falsche.
  - Das Programm terminiert nach 3 falschen Antworten.

```
4 + 10 + 3 + 10 = 27
9 + 2 = 11
8 + 6 + 7 + 9 = 25
Wrong! The answer was 30
5 + 9 = 13
Wrong! The answer was 14
4 + 9 + 9 = 22
3 + 1 + 7 + 2 = 13
4 + 2 + 10 + 9 + 7 = 42
Wrong! The answer was 32
You earned 4 total points.
```

2D-Array

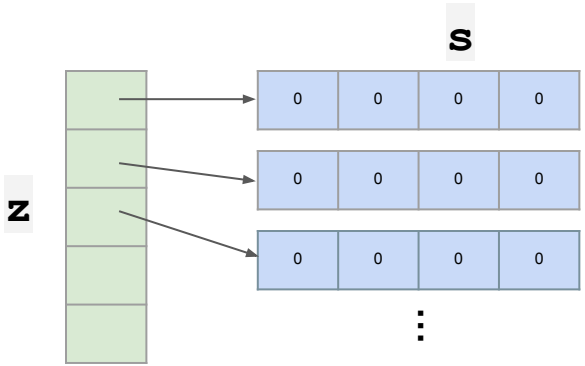
Java hat auch 2-dimensionale Arrays. Ein 2D-Array mit Elementen des Typs `type` wird so deklariert:

```
type [] [] array2d;
```

Der Array muss natürlich mit **`new`** erschaffen werden, z.B.:

```
int [] [] x = new int [zeilen] [spalten];
```

Ein 2D-Array ist ein lineares Array von Zeilen. Der **`new`** Operator initialisiert die Elemente der Zeilen mit dem Nullwert des Typs, z.B. 0 für **`int`**.



`array2d[z][s]` selektiert Zeile **`z`** und darin das **`s`**-te Element (die **`s`**-te Spalte)

2D-Array

Schreiben Sie ein Programm, das einen `int N` einliest und dann einen `N * N int`-Array erschafft und die Elemente entlang der Diagonale auf 1 setzt.

- Wie geben Sie das Array aus?