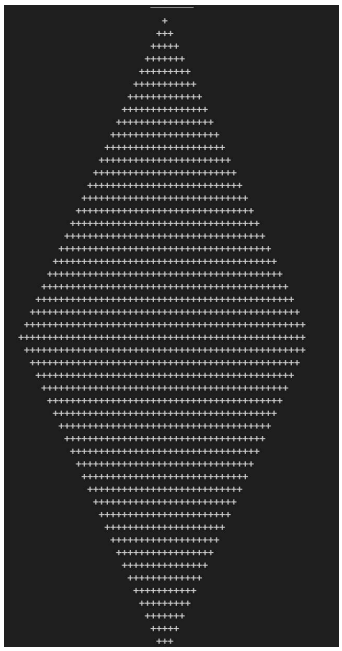
Uebung 4 Patryk Dajos

Aufgabe 1)

1) Man kann die einzelner schritte des codes sich anschauen und herausfinden was falsch an dem code ist!
Man kann in die einzelne Variablen nach jedem Schritt schauen und sich Gedanken machen ob das die Variablen sind die wir haben wollen!

Man hat breakpoints, Step Over, Step Into, Step Out, Restart, Pause, etc. als Funktionen.

2)
Fehler: Das Programm bleibt in der zweiten "for" Schleife stecken, weil es bis "j < 50" geht, die Variable j aber immer 0 bleibt weil der letzte Teil der Schleife wiedermals die Variable "i", die zu der ersten "for" schleife gehört, um 1 erhöht.
Das kommt heraus:



Aufgabe 2)

1)
Code conventions werden auch code standards genannt. Code Standards beschreiben das Erstellen vom Code nach bestimmten Regeln.

2)
Begrenzung der Zeilenlänge, weil kürzere Zeilen leichter lesbar sind.
Variablen, Methoden und Klassen Namen die Sinn machen, damit der Code leichter wartbar und verständlicher ist.
Kommentare vor den Funktionen, Methoden und Klassen, damit das Projekt leichter wartbar und verständlicher ist.

3)
Code conventions machen es leichter in Teams zu arbeiten. Sie machen den Code verständlich, lesbar und leichter wartbar. Diese sind nötig in großen und/oder gut

ausgeplanten Projekten.

Aufgabe 3)
1)
Zwei verschiedene Datentypen sollen in einer Zeile nicht deklariert werden.

Variablen sollen ihre Funktion rechts in der selben Zeile in einem Kommentar beschrieben haben. Zum Beispiel: int level; //indentation level

Variablen Namen gehören kleingeschrieben, jedes einzelne Wort nach dem ersten Wort soll großgeschrieben werden. Zum Beispiel: int level, int levelSize, int levelSizeTemp;

2)

```
if (condition) {

   statements;

}
```

ODER

```
if (condition) {

   statements;

} else {

   statements;

}
```

ODER

```
if (condition) {

   statements;

} else if (condition) {

   statements;

} else {

   statements;

}
```

Man soll die {} klammern nie weglassen weil sonst kommt man auf einem Error im compiling.

3)
Stichwörter, die mit einer Klammer folgen sollen mit Leerzeichen getrennt werden.
Eine liste von argumenten die mit kommas folgen, sollen immer mit Leerzeichen getrennt werden.
Alle Operatoren sollen von deren Operanden mit Leerzeichen getrennt werden.
Alle Äußerungen in einer for, foreach Schleife sollen mit einem Leerzeichen getrennt werden.

Alle Casts sollen mit einem Leerzeichen getrennt werden.

4)
Klassen: Jedes Wort soll mit Großbuchstaben anfangen, die Namen sollen simpel und beschreibend sein. Komplette/Lange Wörter sollen vermieden werden, Abkürzungen werden erwartet.

Methoden: Sollen eine Aktion beschreiben. Sollen Verben sein. Das erste Wort soll kleingeschrieben werden, später jedes einzelne Wort soll großgeschrieben werden.

Variablen: Das erste Wort soll kleingeschrieben werden, später jedes einzelne Wort soll großgeschrieben werden. Sollen nicht mit _ oder $ beginnen. Sollen kurz und beschreibend sein, 1 buchstaben Variablen sollen vermieden werden, außer wenn sie temporär sind. "i,j,k,m,n" für int's und "c,d,e" für char's oder string's.

Konstanten: Alle Buchstaben sollen großgeschrieben werden, wörter werden mit "_" getrennt.

5)
Implementation comments vom Code mit "/* (content) */" und Documentation comments mit "//".

Aufgabe 4)
1)
Variablen, Methoden/Functions, Bugs, Klassen(am Anfang der Datei), usw.

2)
@author tag ist nicht kritisch, soll den Author des Pakets/Codes nennen.
@version Die Version der Software zu dem dieser Code gehört.
@param wird vor dem Namen einer Variable geschrieben, danach folgt eine kurze beschreibung einer Variable.
@return beschreibt was für ein Datentyp und was für Daten eine Methode/Funktion zurückliefert. Wenn es sich um eine Void Methode handelt dann welche funktionalität diese liefert.

Aufgabe 5)
1)

```java
public class Streusel {
    Run | Debug
    public static void main(String[] args) {
        double maxX = 500;
        double maxY = 500;
        int count1 = 0;
        int count2 = 1;
        float inCircle = 0.0f;
        float outCircle = 0.0f;
        while (count1 < 3000000) {
            int[] temp = getRandXY(maxX, maxY);
            if ((Math.pow((temp[0] - (maxX / 2)), 2) + Math.pow((temp[1] - (maxY / 2)), 2)) < Math.pow(maxX / 2, 2)) {
                inCircle++;
            } else {
                outCircle++;
            }
            if (count1 == (300000 * count2)) {
                float result = (4.0f * (inCircle / outCircle)) / 100 + 3;
                System.out.println("CHECKPOINT " + count1 + " POINTS: " + inCircle + " In Circle, " + outCircle + " Out Circle.");
                System.out.println("Resulted Pi: " + result);
                count2++;
            }
            count1++;
        }
        float result = (4.0f * (inCircle / outCircle)) / 100 + 3;
        System.out.println("-----------------------------------------------------");
        System.out.println("FINISHED, RESULTS: " + inCircle + " In Circle, " + outCircle + " Out Circle.");
        System.out.println("Resulted Pi: " + result);
    }

    public static int[] getRandXY(double maxX, double maxY) {
        int[] coord = new int[] { (int) (Math.floor(Math.random() * (maxX - 0 + 1)) + 0),
                (int) (Math.floor(Math.random() * (maxY - 0 + 1)) + 0) };
        return coord;
    }
}
```

Ich habe hier mir immer X und Y koordinaten mit Math.random genommen, und mit Pythagoras habe ich herausgefunden ob diese Position sich außerhalb oder innerhalb des Kreises befindet.
Mein Quadrat war 500x500 positionen groß.
Diese Methode habe ich auf dem Internet gefunden, Stackoverflow:
https://stackoverflow.com/questions/481144/equation-for-testing-if-a-point-is-inside-a-circle
"(x-center_x)^2 + (y - center_y)^2 < radius^2"
Das ganze hat dann auch gut funktioniert :)

```
CHECKPOINT 300000 POINTS: 234584.0 In Circle, 65417.0 Out Circle.
Resulted Pi: 3.1434393
CHECKPOINT 600000 POINTS: 468831.0 In Circle, 131170.0 Out Circle.
Resulted Pi: 3.142969
CHECKPOINT 900000 POINTS: 703580.0 In Circle, 196421.0 Out Circle.
Resulted Pi: 3.14328
CHECKPOINT 1200000 POINTS: 938047.0 In Circle, 261954.0 Out Circle.
Resulted Pi: 3.1432385
CHECKPOINT 1500000 POINTS: 1172429.0 In Circle, 327572.0 Out Circle.
Resulted Pi: 3.143166
CHECKPOINT 1800000 POINTS: 1407033.0 In Circle, 392968.0 Out Circle.
Resulted Pi: 3.1432211
CHECKPOINT 2100000 POINTS: 1641848.0 In Circle, 458153.0 Out Circle.
Resulted Pi: 3.1433449
CHECKPOINT 2400000 POINTS: 1875961.0 In Circle, 524040.0 Out Circle.
Resulted Pi: 3.1431923
CHECKPOINT 2700000 POINTS: 2110329.0 In Circle, 589672.0 Out Circle.
Resulted Pi: 3.1431527
-----------------------------------------------------
FINISHED, RESULTS: 2344710.0 In Circle, 655290.0 Out Circle.
Resulted Pi: 3.143125
```

2)

```java
public class Wallis {
    Run | Debug
    public static void main(String[] args) {
        int count1 = 0;
        int count2 = 1;
        int countup = 1;
        int countdown = 2;
        float up = 2.00f;
        float down = 1.00f;
        float result = 2.00f/1.00f;
        while (count1 < 3000000) {
            if (countdown == 2) {
                countdown = 1;
                down = down + 2.00f;
            } else {
                countdown++;
            }
            if (countup == 2) {
                countup = 1;
                up = up + 2.00f;
            } else {
                countup++;
            }
            result = result * (up / down);
            if (count1 == (300000 * count2)) {
                System.out.println("CHECKPOINT " + 300000 * count2 + " POINTS");
                System.out.println("Resulted Pi: " + result * 2);
                count2++;
            }
            count1++;
        }
        System.out.println("FINISHED");
        System.out.println("Resulted Pi: " + result * 2);
    }
}
```

```
CHECKPOINT 300000 POINTS
Resulted Pi: 3.1425169
CHECKPOINT 600000 POINTS
Resulted Pi: 3.1438067
CHECKPOINT 900000 POINTS
Resulted Pi: 3.1435273
CHECKPOINT 1200000 POINTS
Resulted Pi: 3.1457736
CHECKPOINT 1500000 POINTS
Resulted Pi: 3.1453185
CHECKPOINT 1800000 POINTS
Resulted Pi: 3.1407118
CHECKPOINT 2100000 POINTS
Resulted Pi: 3.1352856
CHECKPOINT 2400000 POINTS
Resulted Pi: 3.142856
CHECKPOINT 2700000 POINTS
Resulted Pi: 3.142856
FINISHED
Resulted Pi: 3.142857
```

3)

```java
import java.util.Scanner;

public class WallisFaktoren {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Geben sie eine Genauigkeit an: ");
        float genau = input.nextFloat();

        int count = 0;
        int countup = 1;
        int countdown = 2;
        float up = 2.00f;
        float down = 1.00f;
        float result = 2.00f / 1.00f;
        while (Math.abs((result*2)-Math.PI) > genau) {
            if (countdown == 2) {
                countdown = 1;
                down = down + 2.00f;
            } else {
                countdown++;
            }
            if (countup == 2) {
                countup = 1;
                up = up + 2.00f;
            } else {
                countup++;
            }
            result = result * (up / down);
            count++;
        }
        System.out.println("Es werden " + count + " WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.");
        System.out.println("Math.Pi(" + Math.PI + ") - Result(" + result*2 + ") = " + ((result*2)-Math.PI));
        input.close();
    }
}
```

```
Geben sie eine Genauigkeit an:
0.3
Es werden 3 WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.
Math.Pi(3.141592653589793) - Result(2.8444448) = -0.29714790185029116
```

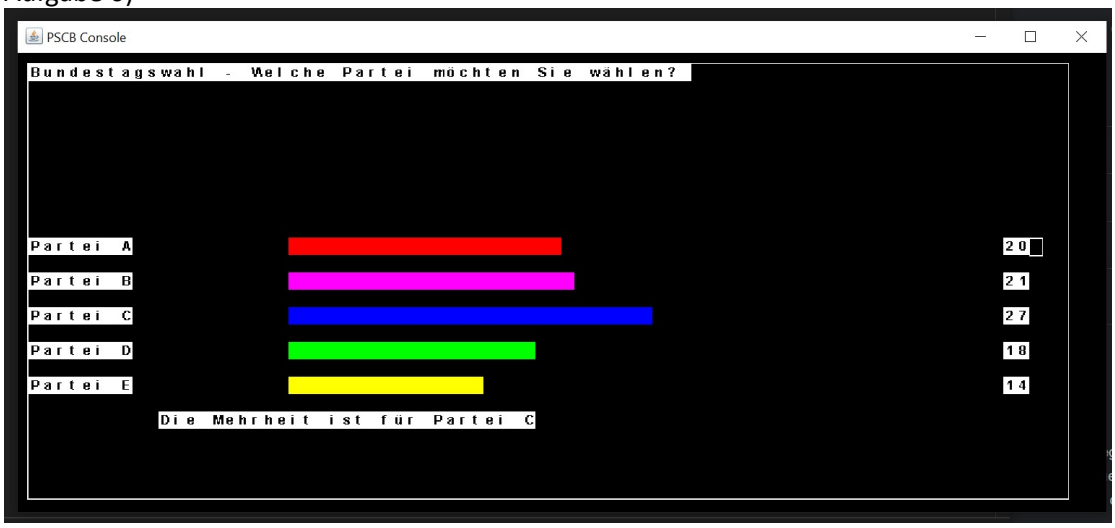
```
Geben sie eine Genauigkeit an:
0.03
Es werden 51 WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.
Math.Pi(3.141592653589793) - Result(3.112098) = -0.02949467499787417
```

```
Geben sie eine Genauigkeit an:
0.003
Es werden 521 WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.
Math.Pi(3.141592653589793) - Result(3.1385992) = -0.0029934962564190926
```

```
Geben sie eine Genauigkeit an:
0.0003
Es werden 4867 WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.
Math.Pi(3.141592653589793) - Result(3.1412928) = -2.9984314972963944E-4
```

```
Geben sie eine Genauigkeit an:
0.00003
Es werden 28761 WallisFaktoren benötigt um ihre Genauigkeit zu kriegen.
Math.Pi(3.141592653589793) - Result(3.1415627) = -2.9953318186670685E-5
```

Aufgabe 6)



Geschafft.