

Schau mal – ein Vokal

Die Hexenjagd 4

Florian Sihler ◦ KW 47



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int randoms[] = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
    }  
}
```

Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int randoms[] = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
    }  
}
```



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int randoms[] = new int[arrayLength];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
    }  
}
```

That does not
look right...



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int[] randoms = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
  
        // 🐧 Platzhalterpingu  
    }  
}
```

Better



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int[] randoms = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
  
        // 🐧 Platzhalterpingu  
    }  
}
```

Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int[] randoms = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
  
        // 🐧 Platzhalterpingu  
    }  
}
```

Ersetzen Sie 🐧 mit Code zur Berechnung der folgenden statistische Kenngrößen für randoms:
1) Minimum, 2) Maximum,
3) Mittelwert und 4) Median.
Geben Sie sie aus.



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int[] randoms = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
  
        // 🐧 Platzhalterpingu  
    }  
}
```

Ersetzen Sie 🐧 mit Code zur Berechnung der folgenden statistische Kenngrößen für randoms:
1) Minimum, 2) Maximum,
3) Mittelwert und 4) Median.
Geben Sie sie aus.

Arrays::sort sortiert ein Array!



Präsenzaufgabe

1

Maximal Medium – Charakterlimit erreicht

```
public class Statistics {  
    public static void main(String[] args) {  
        // Zwischen 10 und 20  
        int length = (int) (Math.random() * 11 + 10);  
        int[] randoms = new int[length];  
  
        for(int i = 0; i < length; i++) {  
            // Zwischen 1 und 100  
            randoms[i] = (int) (Math.random() * 100 + 1);  
        }  
  
        // 🐧 Platzhalterpingu  
    }  
}
```

Ersetzen Sie 🐧 mit Code zur Berechnung der folgenden statistische Kenngrößen für randoms:
1) Minimum, 2) Maximum,
3) Mittelwert und 4) Median.
Geben Sie sie aus.

java.util.Arrays

Arrays::sort sortiert ein Array!



Präsenzaufgabe - Lösung

Präsenzaufgabe - Lösung

- Der gesamte Code ist wieder hier: [Statistics.java](#)

Präsenzaufgabe - Lösung

- Der gesamte Code ist wieder hier: [Statistics.java](#)
- Sortieren macht das finden von Mini- und Maximum leicht:

Präsenzaufgabe - Lösung

- Der gesamte Code ist wieder hier: [Statistics.java](#)
- Sortieren macht das finden von Mini- und Maximum leicht:

```
Arrays.sort(randoms);
```

```
int min = randomness[0]; // first  
int max = randomness[length-1]; // last
```

Präsenzaufgabe - Lösung

Präsenzaufgabe - Lösung

- Es geht mir mittelmäßig. Ich komm schon drüber weg...

Präsenzaufgabe - Lösung

- Es geht mir mittelmäßig. Ich komm schon drüber weg...

```
double avg = 0.0;
```

Präsenzaufgabe - Lösung

- Es geht mir mittelmäßig. Ich komm schon drüber weg...

```
double avg = 0.0;  
for(int i = 0; i < length; i++) {  
  
}
```

Präsenzaufgabe - Lösung

- Es geht mir mittelmäßig. Ich komm schon drüber weg...

```
double avg = 0.0;
for(int i = 0; i < length; i++) {
    avg += randoms[i];
}
```

Präsenzaufgabe - Lösung

- Es geht mir mittelmäßig. Ich komm schon drüber weg...

```
double avg = 0.0;
for(int i = 0; i < length; i++) {
    avg += randoms[i];
}
avg /= length; // avg = avg / length;
```

Präsenzaufgabe - Lösung

Präsenzaufgabe - Lösung

- Der Median ist bei ungerade vielen Elementen das mittlere und sonst der Durchschnitt der mittleren Elemente:

Präsenzaufgabe - Lösung

- Der Median ist bei ungerade vielen Elementen das mittlere und sonst der Durchschnitt der mittleren Elemente:

```
double med = 0.0;
```

Präsenzaufgabe - Lösung

- Der Median ist bei ungerade vielen Elementen das mittlere und sonst der Durchschnitt der mittleren Elemente:

```
double med = 0.0;  
if(length % 2 == 1) {  
  
} else {  
  
  
}
```


Präsenzaufgabe - Lösung

- Der Median ist bei ungerade vielen Elementen das mittlere und sonst der Durchschnitt der mittleren Elemente:

```
double med = 0.0;
if(length % 2 == 1) {
    med = randoms[(length / 2)];
} else {
    med = (randoms[(length / 2)] + randoms[(length / 2) - 1]) / 2;
}
```

Präsenzaufgabe - Lösung

- Der Median ist bei ungerade vielen Elementen das mittlere und sonst der Durchschnitt der mittleren Elemente:

```
double med = 0.0;
if(length % 2 == 1) {
    med = randoms[(length / 2)];
} else {
    med = randoms[(length / 2) - 1] + randoms[(length / 2)];
    med /= 2; // med = med / 2;
}
```

Präsenzaufgabe - Lösung

Präsenzaufgabe - Lösung

- Und nun der shout-out an die Fans:

Präsenzaufgabe - Lösung

- Und nun der shout-out an die Fans:

```
System.out.println("Minimum:␣" + min);  
System.out.println("Maximum:␣" + max);  
System.out.println("Mittelwert:␣" + avg);  
System.out.println("Median:␣" + med);
```

Präsenzaufgabe - Lösung

- Und nun der shout-out an die Fans:

```
System.out.println("Minimum:␣" + min);  
System.out.println("Maximum:␣" + max);  
System.out.println("Mittelwert:␣" + avg);  
System.out.println("Median:␣" + med);
```

- `java` Statistics:

```
Minimum: 7  
Maximum: 89  
Mittelwert: 53.63636363636363  
Median: 51.0
```


They've come...

They've come...

The Abgabeformalitäten are here.

Abgabeformalitäten

Abgabeformalitäten

- Java-Programmieraufgaben:

Abgabeformalitäten

- Java-Programmieraufgaben:



Nichts handschriftliches oder Pseudocode.

Abgabeformalitäten

- Java-Programmieraufgaben:



Nichts handschriftliches oder Pseudocode.



Nur .java-Dateien

Abgabeformalitäten

- Java-Programmieraufgaben:





-  Nichts handschriftliches oder Pseudocode.

-  Nur .java-Dateien

-  (Teil-)Aufgabe in separaten .java-Dateien.





Abgabeformalitäten

■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Abgabeformalitäten





■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

} Sonst (über Blätter hinweg):

Abgabeformalitäten

■ Java-Programmieraufgaben:





-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):

1. Mal: -25% der Punkte der Teilaufgabe

Abgabeformalitäten

■ Java-Programmieraufgaben:





-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):

1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe

Abgabeformalitäten

■ Java-Programmieraufgaben:





-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):

1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe
3. Mal: -100% der Punkte der Teilaufgabe

Abgabeformalitäten

■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.





Sonst (über Blätter hinweg):

- Sie müssen  **immer**  lauffähig sein.

1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe
3. Mal: -100% der Punkte der Teilaufgabe

Abgabeformalitäten

■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):





■ Sie müssen **immer** lauffähig sein.

- Fehlerhaften Code auskommentieren

1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe
3. Mal: -100% der Punkte der Teilaufgabe

Abgabeformalitäten

■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):





1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe
3. Mal: -100% der Punkte der Teilaufgabe

■ Sie müssen **immer** lauffähig sein.

- Fehlerhaften Code auskommentieren
- Anmerken, was das Ziel war.

Abgabeformalitäten

■ Java-Programmieraufgaben:

-  Nichts handschriftliches oder Pseudocode.
-  Nur .java-Dateien
-  (Teil-)Aufgabe in separaten .java-Dateien.
-  Formatierung nicht wie bei Hempels unterm Sofa.

Sonst (über Blätter hinweg):

1. Mal: -25% der Punkte der Teilaufgabe
2. Mal: -50% der Punkte der Teilaufgabe
3. Mal: -100% der Punkte der Teilaufgabe

■ Sie müssen **immer** lauffähig sein.

- Fehlerhaften Code auskommentieren
- Anmerken, was das Ziel war.

Sonst *null Punkte*.

- Textaufgaben:

- Textaufgaben:
 - Handschriftlich ist Ok, sofern lesbar.

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (👍) aber **nicht verpflichtend**.

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (👍) aber **nicht verpflichtend**.
- Wir öffnen nur .java, .txt und .pdf Formate.

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (👍) aber **nicht verpflichtend**.
- Wir öffnen nur .java, .txt und .pdf Formate.

■ Allgemein

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (👍) aber **nicht verpflichtend**.
- Wir öffnen nur .java, .txt und .pdf Formate.

■ Allgemein

- Sprecht mit anderen und tauscht eure **Ideen** aus.

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (🐼) aber **nicht verpflichtend**.
- Wir öffnen nur .java, .txt und .pdf Formate.

■ Allgemein

- Sprecht mit anderen und tauscht eure **Ideen** aus.
- Aber bitte nicht eure Lösungen.

■ Textaufgaben:

- Handschriftlich ist Ok, sofern lesbar.
- \LaTeX ist tsöööön (👉) aber **nicht verpflichtend**.
- Wir öffnen nur .java, .txt und .pdf Formate.

■ Allgemein

- Sprecht mit anderen und tauscht eure **Ideen** aus.
- Aber bitte nicht eure Lösungen.
- Es soll übermotivierte Tutoren geben, die an Clone-Detection arbeiten... und die sich alle Abgaben und Bewertungen ansehen.

Übungsblatt 4 - Aufgabe 1

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

```
int a = 0, e = 0, i = 0, o = 0, u = 0;
```

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

```
int a = 0, e = 0, i = 0, o = 0, u = 0;
```

```
for(int j = 0; j < input.length(); j++) {
```

```
}
```

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

```
int a = 0, e = 0, i = 0, o = 0, u = 0;
```

```
for(int j = 0; j < input.length(); j++) {  
    switch(input.toLowerCase().charAt(j)) {
```

```
    }  
}
```

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

```
int a = 0, e = 0, i = 0, o = 0, u = 0;
```

```
for(int j = 0; j < input.length(); j++) {  
    switch(input.toLowerCase().charAt(j)) {  
        case 'a': a++; break;
```

```
    }  
}
```

Übungsblatt 4 - Aufgabe 1

- Banale Vokale `CountVowels.java`

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

Übungsblatt 4 - Aufgabe 1

Übungsblatt 4 - Aufgabe 1

- Und die Ausgabe:

Übungsblatt 4 - Aufgabe 1

- Und die Ausgabe:

```
System.out.println("a:␣" + a);  
System.out.println("e:␣" + e);  
System.out.println("i:␣" + i);  
System.out.println("o:␣" + o);  
System.out.println("u:␣" + u);
```

Übungsblatt 4 - Aufgabe 1

Übungsblatt 4 - Aufgabe 1

Und nun...

Und nun...
Entdecke den **Overengineer-Kavalier** in dir

Und nun...
Entdecke den Overengineer-Kavalier in dir

Schnabeltier


```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}

System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```



```
int a = 0, e = 0, i = 0, o = 0, u = 0;
```

```
for(int j = 0; j < input.length(); j++) {  
    switch(input.toLowerCase().charAt(j)) {  
        case 'a': a++; break;  
        case 'e': e++; break;  
        case 'i': i++; break;  
        case 'o': o++; break;  
        case 'u': u++; break;  
    }  
}
```

```
System.out.println("a:␣" + a);  
System.out.println("e:␣" + e);  
System.out.println("i:␣" + i);  
System.out.println("o:␣" + o);  
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];

for(int j = 0; j < arr.length; j++) {

}
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a: " + a);
System.out.println("e: " + e);
System.out.println("i: " + i);
System.out.println("o: " + o);
System.out.println("u: " + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];

for(int j = 0; j < arr.length; j++) {
    char c = arr[j];
}
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];

for(int j = 0; j < arr.length; j++) {
    char c = arr[j];
    if('a' <= c && c <= 'z')
}
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];

for(int j = 0; j < arr.length; j++) {
    char c = arr[j];
    if('a' <= c && c <= 'z')
        counters[c - 'a'] += 1;
}
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];
```

```
for(int j = 0; j < arr.length; j++) {
    char c = arr[j];
    if('a' <= c && c <= 'z')
        counters[c - 'a'] += 1;
}
```

```
for(char c : new char[]{'a', 'e', 'i', 'o', 'u'}) {
}
```

```
int a = 0, e = 0, i = 0, o = 0, u = 0;

for(int j = 0; j < input.length(); j++) {
    switch(input.toLowerCase().charAt(j)) {
        case 'a': a++; break;
        case 'e': e++; break;
        case 'i': i++; break;
        case 'o': o++; break;
        case 'u': u++; break;
    }
}
```

```
System.out.println("a:␣" + a);
System.out.println("e:␣" + e);
System.out.println("i:␣" + i);
System.out.println("o:␣" + o);
System.out.println("u:␣" + u);
```

```
char[] arr = input.toLowerCase().toCharArray();
int[] counters = new int[26];
```

```
for(int j = 0; j < arr.length; j++) {
    char c = arr[j];
    if('a' <= c && c <= 'z')
        counters[c - 'a'] += 1;
}
```

```
for(char c : new char[]{'a', 'e', 'i', 'o', 'u'}) {
    System.out.println(c+":␣" + counters[c - 'a']);
}
```


Übungsblatt 4 - Aufgabe 2

Übungsblatt 4 - Aufgabe 2

- Isch bin ene Primeel: [PrimeFactors.java](#).

Übungsblatt 4 - Aufgabe 2

- Isch bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:

Übungsblatt 4 - Aufgabe 2

- Isch bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:
`if (args.length != 1) System.exit(1);`

Übungsblatt 4 - Aufgabe 2

- Isch bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:

```
if (args.length != 1) System.exit(1);
```

```
int N = Integer.parseInt(args[0]);
```

Übungsblatt 4 - Aufgabe 2

- Ist bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:

```
if (args.length != 1) System.exit(1);
```

```
int N = Integer.parseInt(args[0]);
```

```
if (N < 1) {
```

```
} else if (N == 1) {
```

```
}
```

Übungsblatt 4 - Aufgabe 2

- Ist bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:

```
if (args.length != 1) System.exit(1);

int N = Integer.parseInt(args[0]);
if (N < 1) {
    System.exit(1);
} else if (N == 1) {

}
```

Übungsblatt 4 - Aufgabe 2

- Ist bin ene Primeel: [PrimeFactors.java](#).
- Starting with the guardians of the C:

```
if (args.length != 1) System.exit(1);

int N = Integer.parseInt(args[0]);
if (N < 1) {
    System.exit(1);
} else if (N == 1) {
    System.out.println("Die_Zahl_1_besitzt_keine_Primfaktorzerlegung");
    return;
}
```


Übungsblatt 4 - Aufgabe 2

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];
```

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];  
for (int i = 0; i <= N; i++)  
    marker[i] = false; // Sieb ist Sieb!
```

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];  
for (int i = 0; i <= N; i++)  
    marker[i] = false; // Sieb ist Sieb!  
  
for (int i = 2; i < Math.sqrt(N); i++) {  
  
}
```

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];  
for (int i = 0; i <= N; i++)  
    marker[i] = false; // Sieb ist Sieb!  
  
for (int i = 2; i < Math.sqrt(N); i++) {  
    if (!marker[i]) {  
  
    }  
}
```

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];
for (int i = 0; i <= N; i++)
    marker[i] = false; // Sieb ist Sieb!

for (int i = 2; i < Math.sqrt(N); i++) {
    if (!marker[i]) {
        for (int j = i * i; j <= N; j += i) {

        }
    }
}
```

Übungsblatt 4 - Aufgabe 2

- Wir verwenden die Hinweise und Sieben:

```
boolean marker[] = new boolean[N + 1];
for (int i = 0; i <= N; i++)
    marker[i] = false; // Sieb ist Sieb!

for (int i = 2; i < Math.sqrt(N); i++) {
    if (!marker[i]) {
        for (int j = i * i; j <= N; j += i) {
            marker[j] = true;
        }
    }
}
```


Übungsblatt 4 - Aufgabe 2

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlives verhackseln:

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlies verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet: ");
```

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

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlies verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet: ");  
while (N >= 2) {  
  
}  
System.out.println();
```

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet: ");  
while (N >= 2) {  
    for (int i = 2; i < marker.length; i++) {  
  
    }  
}  
System.out.println();
```

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
while (N >= 2) {  
    for (int i = 2; i < marker.length; i++) {  
        if (marker[i]) continue;  
    }  
    System.out.println();
```

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
while (N >= 2) {  
    for (int i = 2; i < marker.length; i++) {  
        if (marker[i]) continue;  
        if ((N % i) == 0) {  
  
        }  
    }  
}  
System.out.println();
```

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
while (N >= 2) {
    for (int i = 2; i < marker.length; i++) {
        if (marker[i]) continue;
        if ((N % i) == 0) {
            N /= i; // N = N / i;
        }
    }
}
System.out.println();
```


Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
while (N >= 2) {
    for (int i = 2; i < marker.length; i++) {
        if (marker[i]) continue;
        if ((N % i) == 0) {
            N /= i; // N = N / i;
            System.out.print((N >= 2) ? i + "*" : i);
        }
    }
}
System.out.println();
```

Übungsblatt 4 - Aufgabe 2

- Und nun mit den Primlles verhackseln:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
while (N >= 2) {
    for (int i = 2; i < marker.length; i++) {
        if (marker[i]) continue;
        if ((N % i) == 0) {
            N /= i; // N = N / i;
            System.out.print((N >= 2) ? i + "*" : i);
            break; // Notwendig?
        }
    }
}
System.out.println();
```


- Oder direkt wiederholt anwenden:

- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
```

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


- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
for (int i = 2; i < marker.length; i++) {  
    if (marker[i]) continue;  
  
    }  
System.out.println("");
```

- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
for (int i = 2; i < marker.length; i++) {  
    if (marker[i]) continue;  
    while ((N % i) == 0) {  
  
    }  
}  
System.out.println("");
```


- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
for (int i = 2; i < marker.length; i++) {  
    if (marker[i]) continue;  
    while ((N % i) == 0) {  
        N /= i;  
    }  
}  
System.out.println("");
```

- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
for (int i = 2; i < marker.length; i++) {  
    if (marker[i]) continue;  
    while ((N % i) == 0) {  
        N /= i;  
        System.out.print((N >= 2) ? i + "*" : i);  
    }  
}  
System.out.println("");
```

- Oder direkt wiederholt anwenden:

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
for (int i = 2; i < marker.length; i++) {
    if (marker[i]) continue;
    while ((N % i) == 0) {
        N /= i;
        System.out.print((N >= 2) ? i + "*" : i);
    }
}
System.out.println("");
```

- `java PrimeFactors 12301234`:

```
Die Primfaktorzerlegung von 12301234 lautet: 2 * 11 * 17 * 31 * 1061
```


- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet: ");
```

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

- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
int max = N;
```

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

- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");
int max = N;
for (int i = 2; i <= max; i++) {

}
System.out.println("");
```


- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
int max = N;  
for (int i = 2; i <= max; i++) {  
    while ((N % i) == 0) {  
  
    }  
}  
System.out.println("");
```

- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
int max = N;  
for (int i = 2; i <= max; i++) {  
    while ((N % i) == 0) {  
        N /= i;  
        System.out.print((N >= 2) ? i + "*" : i);  
    }  
}  
System.out.println("");
```

- Wir können das Sieb direkt integrieren... ([PrimeFactorsAlternate.java](#))

```
System.out.print("Die Primfaktorzerlegung von " + N + " lautet:");  
int max = N;  
for (int i = 2; i <= max; i++) {  
    while ((N % i) == 0) {  
        N /= i;  
        System.out.print((N >= 2) ? i + "*" : i);  
    }  
}  
System.out.println("");
```

- `java PrimeFactorsAlternate 12301234`:

```
Die Primfaktorzerlegung von 12301234 lautet: 2 * 11 * 17 * 31 * 1061
```