

# Table of Contents

Laboreinheit: Algorithmen und Datenstrukturen – Aufgabenblatt 1 .....	1
Aufgaben .....	1
Deadline und Punkte .....	11
Feedback .....	11
Literaturverzeichnis .....	12
Appendix A: Abgabebedingungen .....	13
Appendix B: Bewertung und Verteilung der Klausurpunkte (KP) .....	14

## Laboreinheit: Algorithmen und Datenstrukturen – Aufgabenblatt 1

Tom Kamberg-Buhrtz <[buhrtz@htw-berlin.de](mailto:buhrtz@htw-berlin.de)>



**Die Abgabe dieses Aufgabenblattes ist Voraussetzung für die Abgabe von weiteren Aufgabenblättern.** In diesem Aufgabenblatt werden Sie sich mit verschiedenen Disziplinen beschäftigen, welche Sie zum Teil im Selbststudium meistern müssen.

### Aufgaben

1. **Installieren** Sie auf Ihrem Arbeitsrechner das **Java Development Kit (JDK)** und **Gradle**. Die Installationsanleitung für Gradle finden Sie unter folgendem Link: <https://gradle.org/install>. Nach der Installation sollte Ihnen der Befehl `gradle --version` auf der **Command-line** zur Verfügung stehen.



Am Besten nach der Installation ein neues Konsolenfenster öffnen, damit die neuen Suchpfade nach dem Setzen geladen werden!



Bei diesem Beispiel wurde das **Java Development Kit (JDK) in der Version 16.x.y** und **Gradle in der Version 7.2** verwendet. Die Kompatibilität zwischen der JAVA- und Gradle-Version muss von Ihnen sichergestellt werden (siehe <https://docs.gradle.org/current/userguide/compatibility.html>).

(2 Pkt.)

2. Erstellen Sie bitte einen neuen Projektordner und führen Sie innerhalb des Ordners den Befehl `gradle init` per Command-line Interface aus, um eine neue Java Application mit den folgenden Optionen zu erstellen.

```
$ mkdir -p ~/Desktop/Exercise-1/MyApp
$ cd ~/Desktop/Exercise-1/MyApp
$ gradle init
```

Select type of project to generate:

- 1: basic
- 2: application
- 3: library
- 4: Gradle plugin

Enter selection (default: basic) [1..4] 2

Select implementation language:

- 1: C++
- 2: Groovy
- 3: Java
- 4: Kotlin
- 5: Scala
- 6: Swift

Enter selection (default: Java) [1..6] 3

Split functionality across multiple subprojects?:

- 1: no - only one application project
- 2: yes - application and library projects

Enter selection (default: no - only one application project) [1..2] 1

Select build script DSL:

- 1: Groovy
- 2: Kotlin

Enter selection (default: Groovy) [1..2] 1

Select test framework:

- 1: JUnit 4
- 2: TestNG
- 3: Spock
- 4: JUnit Jupiter

Enter selection (default: JUnit Jupiter) [1..4] 1

Project name (default: MyApp):

Source package (default: MyApp):

> Task :init

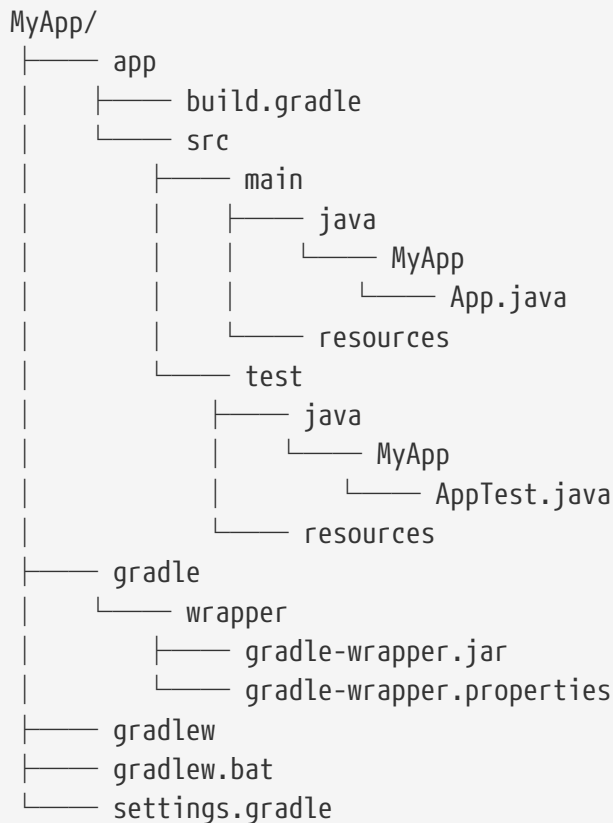
Get more help with your project:

[https://docs.gradle.org/7.2/samples/sample\\_building\\_java\\_applications.html](https://docs.gradle.org/7.2/samples/sample_building_java_applications.html)

BUILD SUCCESSFUL in 16s

2 actionable tasks: 2 executed

Nach der erfolgreichen Initialisierung sollte die folgende Projektstruktur vorhanden sein:



Wechseln Sie per **Command-line** in den Projektordner:

```
$ cd ~/Desktop/Exercise-1/MyApp
```

Testen Sie die folgenden Gradle-Wrapper-Befehle auf der Root-Ebene:

\$ ./gradlew clean	# Löscht automatisch generierte Dateien.
\$ ./gradlew build	# Programm kompilieren.
\$ ./gradlew test	# Alle Unittests ausführen.
\$ ./gradlew javadoc	# API Dokumentation erstellen.
\$ ./gradlew run -q --console=plain	# Programm per Console ausführen.
\$ ./gradlew run	# Programm per Console ausführen.



Alle Abgaben müssen die zuvor genannten Gradle-Wrapper-Befehle erfolgreich ausführen können, sonst wird **keine Bewertung des Aufgabenblattes** vorgenommen!



Auf Unix-Systemen kann das Gradle-Wrapper-Skript mit **./gradlew** ausgeführt werden. Bei Microsoft Windows ist es abhängig von der verwendeten Konsole, ob der Befehl **gradlew.bat** oder **./gradlew** eingesetzt werden muss. **Wichtig:** Sie müssen sich für die Ausführung der Befehle immer auf der Root-Ebene befinden, auf der die beiden Skripte liegen!

(5 Pkt.)

3. Fügen Sie bitte die Plugins für IntelliJ, Eclipse, JaCoCo und PDM in die `build.gradle` ein:

```
plugins {  
    id 'application'      ①  
    id 'eclipse'          ②  
    id 'idea'             ③  
    id 'pmd'              ④  
    id 'jacoco'           ⑤  
    // ...  
}
```

① [https://docs.gradle.org/current/userguide/application\\_plugin.html](https://docs.gradle.org/current/userguide/application_plugin.html)

② [https://docs.gradle.org/current/userguide/eclipse\\_plugin.html](https://docs.gradle.org/current/userguide/eclipse_plugin.html)

③ [https://docs.gradle.org/current/userguide/idea\\_plugin.html](https://docs.gradle.org/current/userguide/idea_plugin.html)

④ [https://docs.gradle.org/current/userguide/pmd\\_plugin.html](https://docs.gradle.org/current/userguide/pmd_plugin.html)

⑤ [https://docs.gradle.org/current/userguide/jacoco\\_plugin.html](https://docs.gradle.org/current/userguide/jacoco_plugin.html)

Testen Sie die neuen Gradle-Wrapper-Befehle mit:

```
./gradlew eclipse      # Generates all Eclipse files.  
./gradlew idea         # Generates IDEA project files (IML, IPR, IWS)  
./gradlew check        # Runs all checks
```



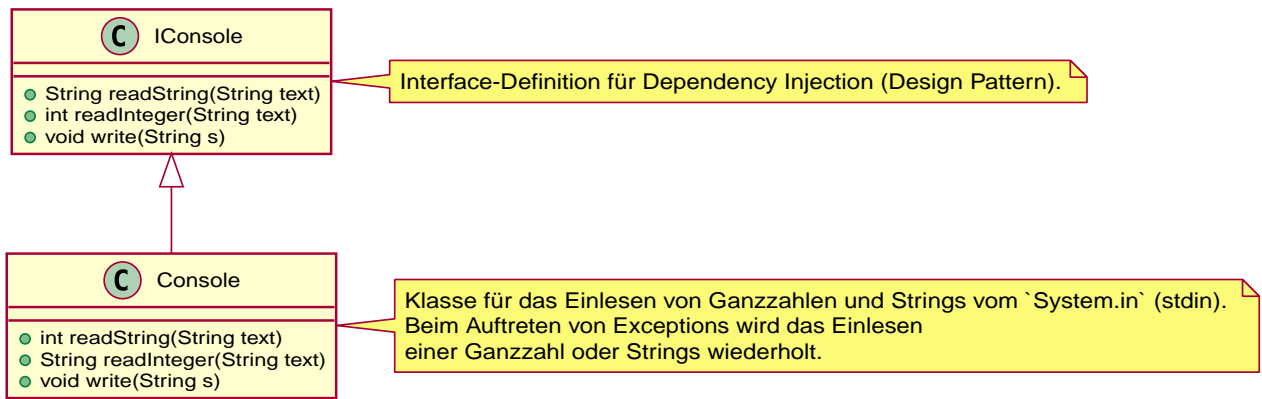
Nach der **erfolgreichen Ausführung dieser Befehle** können Sie die Projekte in der **Eclipse IDE** oder **IntelliJ IDEA** öffnen. Die Projektdateien werden automatisch über die Gradle-Wrapper-Befehle generiert. Mit den Befehlen `./gradlew cleanEclipse` oder `./gradlew cleanIdea` können die Projektdateien wieder entfernt werden. Das jeweilige IDE-Projekt sollte dafür geschlossen sein!



Sie können sich die Gradle-Wrapper-Tasks mit dem Befehl: `./gradlew task` oder `gradlew task --all` als Liste anzeigen lassen. Die Auflösung der Abhängigkeiten können Sie sich mit dem Befehl: `./gradlew build -i` anschauen. Weitere Command-line Interface-Optionen können Sie sich mit dem Befehl `./gradlew --help` anzeigen lassen.

(4 Pkt.)

4. Legen Sie bitte ein neues Interface **IConsole** und eine Klasse **Console** an.



a. Implementieren Sie eine Methode zum Einlesen von Ganzzahlen (int), zum Beispiel:

```
Enter number for x: 43<enter>
Enter number for y: 13<enter>
```

Testen Sie die Eingabe von folgenden Werten: **Sheldon, 1** und **Milch macht munter**.



Wird eine ungültige Zeichenkette eingegeben, muss die Eingabe solange wiederholt werden, bis eine gültige Eingabe erfolgt ist oder das CLI-Programm durch den Benutzer abgebrochen wird.

b. Implementieren Sie eine Methode zum Einlesen von Zeichenketten (String), zum Beispiel:

```
Enter: Max Mustermann<enter> // z.B.
```

Testen Sie die Eingabe von Sonderzeichen und Steuerzeichen, wie z.B. `%&/\r?"\n\t`, **1**, **Sheldon Cooper** oder **Ted**. Trimmen Sie Steuerzeichen und Leerzeichen vor und nach einer Zeichenkette weg.

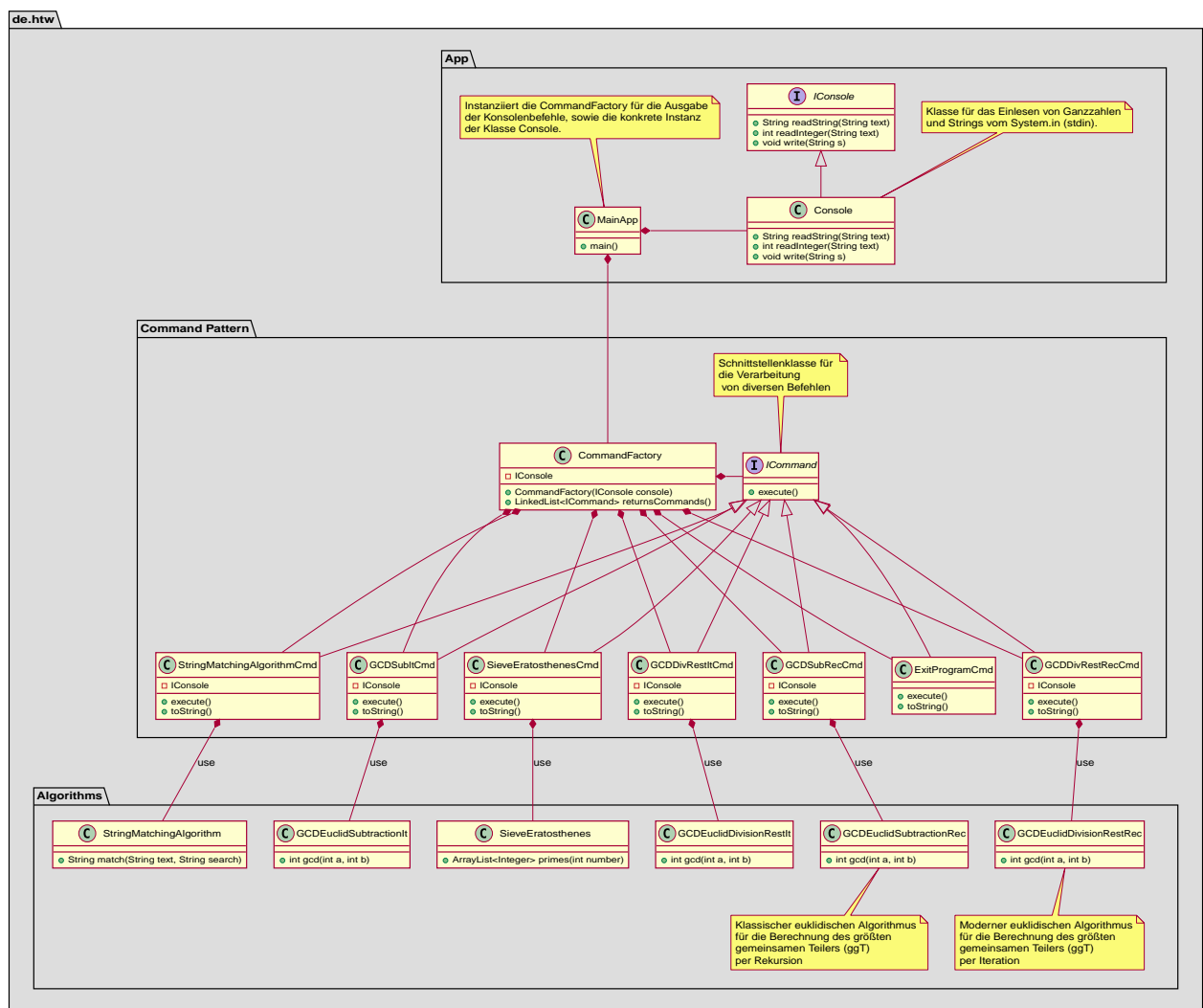
(8 Pkt.)

5. Implementieren Sie bitte ein Konsolenmenü mit folgendem Aussehen:

1. Greatest Common Divisor (GCD) - (Recursive) Euclid's algorithm subtraction.
2. Greatest Common Divisor (GCD) - (Iterative) Euclid's algorithm subtraction.
3. Greatest Common Divisor (GCD) - (Recursive) Euclid's algorithm division rest.
4. Greatest Common Divisor (GCD) - (Iterative) Euclid's algorithm division rest.
5. Sieve of Eratosthenes.
6. Search for a specific string in Linus Torvald's joke (String-Matching-Algorithm).
0. Exit.









Please enter a number for an option:

Beim Auswählen einer Option soll der entsprechende Algorithmus ausgeführt werden. Wählt der Benutzer z. B. die Option **5**, dann wird der Algorithmus für das **Sieb des Eratosthenes** aufgerufen. Für die Strukturierung der Klassen und Abarbeitung der folgenden Aufgaben soll Ihnen das folgende Bild nach der **UML-Notation** helfen:



Dieses Beispiel erhebt keinen Anspruch auf Vollständigkeit.

Table 1. Bedeutung der Symbole aus der UML-Notation.

Character	Icon für Feld	Icon für Methode	Sichtbarkeit
-			private
#			protected
~			package private
+			public

[https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)

<http://openbook.rheinwerk-verlag.de/javainasel9/>

[http://openbook.rheinwerk-verlag.de/javainasel9/javainasel\\_05\\_008.htm](http://openbook.rheinwerk-verlag.de/javainasel9/javainasel_05_008.htm)

[https://sourcemaking.com/design\\_patterns](https://sourcemaking.com/design_patterns)

<https://plantuml.com/class-diagram>

(12 Pkt.)

6. Implementieren Sie den **klassischen euklidischen Algorithmus** für die Berechnung des größten gemeinsamen Teilers (ggTs) **ohne Rekursion**. Der Algorithmus soll über das Command-line Interface (CLI) auswählbar und parametrierbar sein.

```

1 EuclidSubtractionIterative(a,b)
2   wenn a = 0
3     dann return b
4   sonst solange b != 0
5     wenn a > b
6       dann a <-- a - b
7     sonst b <-- b - a
8   return a

```

Schreiben Sie ein Unittest mit dem JUnit Test-Framework, welcher nur die Methode für die Berechnung des ggTs testet.

(5 Pkt.)

7. Implementieren Sie den **klassischen euklidischen Algorithmus** für die Berechnung des größten gemeinsamen Teilers (ggTs) **mit Rekursion**. Der Algorithmus soll über die Command-line Interface auswählbar und parametrierbar sein.

```

1 EuclidSubtractionRecursive(a,b)
2   wenn b = 0
3       dann return a
4   sonst wenn a = 0
5       return b
6   sonst wenn a > b
7       dann return EuclidSubtractionRecursive(a - b, b)
8   sonst return EuclidSubtractionRecursive( a, b - a)

```

Schreiben Sie ein Unittest mit dem JUnit Test-Framework, welcher nur die Methode für die Berechnung des ggTs testet.

(5 Pkt.)

8. Implementieren Sie den **modernen euklidischen Algorithmus** für die Berechnung des größten gemeinsamen Teilers (ggTs) **ohne Rekursion**. Der Algorithmus soll über die Command-line Interface auswählbar und parametrierbar sein.

```

1 EuclidDivisionRestIterative(a,b)
2   solange b != 0
3       h <-- a mod b
4       a <-- b
5       b <-- h
6   return a

```

Schreiben Sie Unittests mit dem JUnit Test-Framework, welcher nur die Methode für die Berechnung des ggTs testet.

(5 Pkt.)

9. Implementieren Sie den **modernen euklidischen Algorithmus** für die Berechnung des größten gemeinsamen Teilers (ggTs) **mit Rekursion**. Der Algorithmus soll über die Command-line Interface auswählbar und parametrierbar sein.

```

1 EuclidDivisionRestRecursive(a,b)
2   wenn b = 0
3       dann return a
4   sonst return Euclid(b, a mod b)

```

Schreiben Sie Unittests mit dem JUnit Test-Framework, welcher nur die Methode für die Berechnung des ggTs testet.

(5 Pkt.)

10. Implementieren Sie das Sieb des Eratosthenes. Der Algorithmus soll über die Command-line Interface auswählbar und parametrierbar sein.



- Schreibe alle Zahlen auf (ab 2 bis Maximalwert N),
- Begreife alle diese Zahlen als potentielle Primzahlen,
- Die kleinste unmarkierte Zahl ist immer eine Primzahl,
- Wähle die kleinste unmarkierte Zahl und markiere alle Vielfachen als zusammengesetzt,
- Wähle die nächste unmarkierte Zahl und markiere alle Vielfachen als zusammengesetzt,
- usw.;

Schreiben Sie Unittests mit dem JUnit Test-Framework, welcher nur die Methode für die Berechnung der Primzahlen testet.



Bei N = 11 muss als Ergebnis: 2, 3, 5, 7 und 11 als Ergebnis herauskommen.

<https://www.mathe-lexikon.at/arithmetik/natuerliche-zahlen/teilbarkeit/primzahlen/sieb-des-eratosthenes.html>

(5 Pkt.)

11. Implementieren Sie einen String-Matching-Algorithmus (z. B. den naiven Algorithmus). Dieser soll den Witz von Linus Torvalds: **I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'**; nach einem über die Kommandozeile eingelesenen String (z. B. **Git** oder **Bill Gates**) durchsuchen.

(10 Pkt.)

12. Testen Sie das Konsolenprogramm auf der Kommandozeile. Damit Sie keine Probleme mit dem **System.in** (stdin) bekommen, muss in der **build.gradle** der folgende Abschnitt eingefügt werden:

```
run {
    standardInput = System.in
}
```



Vergessen Sie den oben genannten Abschnitt in der **build.gradle** nicht, sonst kommt es zu einer Endlos-Schleife in den weiteren Abgaben! Sie können diesen Sachverhalt mit **./gradlew run --console=plain -q** testen.

Rufen Sie die Java-Application mit dem Befehl **./gradlew run --console=plain** oder **./gradlew run --console=plain -q** auf, um unnötigen Informationen zu unterdrücken.



Gradle-Beispiel: <https://github.com/CSV-Tom/gradle-example>

(3 Pkt.)

13. Nennen und beschreiben Sie mindestens fünf Eigenschaften von Algorithmen.

(5 Pkt.)

# Deadline und Punkte

Table 2. Maximale Punkte und Deadline

<b>Deadline:</b>	<b>21.11.2021 - 23:55</b>
<b>Maximale KP:</b>	<b>15 KP (Klausurpunkt)</b>
<b>Maximale Punktzahl:</b>	<b>74 Punkte == 100 % == 15 KP</b>



Wird die volle Punktzahl beim Aufgabenblatt erreicht, dann haben Sie **100 %** erfüllt und somit **15 KP \* 1.0 = 15 KP**. Haben Sie nur **50 %** erfüllt, dann berechnen sich die Klausurpunkte wie folgt: **15 KP \* 0.5 = 7.5 KP** usw.

## Feedback

Senden Sie mir gerne konstruktives Feedback zu den Aufgabenblättern und Dokumenten zu.



Verbesserungsvorschläge für dieses Aufgabenblatt können Sie gerne per Git-Patch oder schriftlich per E-Mail an mich senden. Git-Patches geben Ihnen die Möglichkeit, zusätzliche Punkte für ein Aufgabenblatt zu erhalten. Dafür müssen die Git-Patches atomar sein und im ZIP-Archive abgelegt, oder per E-Mail an mich gesendet werden.

**Viel Spaß und Erfolg bei der Bearbeitung der Aufgabenblätter!**

*Erstellungsdatum und -zeit: 2021-10-12 - 22:06:31 +0200*

# Literaturverzeichnis

- [1] Thomas Ottmann and Peter Widmayer, Algorithmen und Datenstrukturen (German Edition), Spektrum Akademischer Verlag}, 2002, ISBN: 3827410290.
- [2] Sebastian Dworatschek, Grundlagen der Datenverarbeitung (de Gruyter Lehrbuch) (German Edition), de Gruyter, 1989, ISBN: 3110120259.
- [3] Robert Sedgewick, Algorithmen in C++ (German Edition), Pearson Studium, 2002, ISBN: 3827370264.
- [4] Ullenboom, Christian (Autor), Java ist auch eine Insel – Das Standardwerk für Programmierer. Über 1.000 Seiten Java-Wissen. Mit vielen Beispielen und Übungen, aktuell zu Java 14, 2020, ISBN: 9783836277372. <https://www.buchhandel.de/buch/Java-ist-auch-eine-Insel-9783836277372>
- [Java10] [Openbook: Java ist auch eine Insel](#)
- [Java11] [Java ist auch eine Insel - Kapitel Vererbung](#)
- [Java12] [Java ist auch eine Insel - Kapitel Vererbung](#)
- [Java13] [JAVA Interfaces](#)
- [Java14] [JUnit-Testframework](#)
- [Java15] [Java ist auch eine Insel - Generics<T>](#)
- [Gradle20] [Gradle](#)
- [Gradle21] [Gradle Installation](#)
- [Gradle22] [Gradle Tutorials and Guides](#)
- [Git-SCM] [Git](#)
- [Git-EN-v2-Book] [Pro Git Book - Englisch \(PDF etc.\)](#)
- [Git-DE-v2-Book] [Pro Git Book - Deutsch](#)
- [HTW70] [HTW Berlin - Projekteserver](#)
- [PlantUML-ClassDiagram] [Plant-UML-Klassendiagramme](#)
- [Asciidoctor] [Asciidoctor](#)
- [QuickAsciidoctor] [Asciidoctor - Quick Reference](#)
- [HTWMoodle] [HTW Berlin - Moodle](#)
- [DeepLTranslator] [AI Assistance for Language](#)
- [Eclipse Plugin for Gradle] [Buildship Plugin for Gradle](#)
- [GraphViz Pocket Reference] [GraphViz Pocket Reference](#)

# Appendix A: Abgabebedingungen

## Voraussetzung, damit eine Bewertung vorgenommen wird!

Bei jeder Abgabe muss der Gradle-Wrapper `./gradlew` im **Projektrootverzeichnis** vorhanden sein und **funktionieren**, sonst wird **keine Bewertung der Aufgabe vorgenommen**. Es dürfen nur relative Pfade verwendet werden. **Die folgenden Tasks müssen per Command-line im Projektrootverzeichnis ausführbar sein:**

<code>./gradlew clean</code>	# Löscht automatisch generierte Dateien.
<code>./gradlew build</code>	# Programm kompilieren.
<code>./gradlew test</code>	# Alle Unittests ausführen.
<code>./gradlew javadoc</code>	# API Dokumentation erstellen.
<code>./gradlew run -q --console=plain</code>	# Programm per Console ausführen.
<code>./gradlew run</code>	# Programm per Console ausführen.



Bitte in der `build.gradle` den Parameter `System.in` setzen, damit die Eingabe von Parametern `./gradlew run -q --console=plain` auf der Konsole per Gradle funktioniert.



Halten Sie bitte die Reihenfolge der Optionen und die Zuordnung der Nummern ein, damit automatische Skripte für die Kontrolle der Abgaben eingesetzt werden können.



Verwalten Sie den Source Code über die Versionsverwaltung Git als Übung für zukünftige Projekte in der Angewandten Informatik (AI).

# Appendix B: Bewertung und Verteilung der Klausurpunkte (KP)

## Wie wird die Bewertung in diesem Semester vorgenommen?

Es müssen folgende Aufgabenblätter in diesem Semester abgegeben werden. Sie können insgesamt 50 Klausurpunkte (KP) bei mir in der Laboreinheit erhalten. Sie müssen mindestens 25 Klausurpunkte (KP) bei mir erreichen, damit Sie an der Prüfung bei Prof. Sieck teilnehmen können.

### Aufgabenblatt 1

15 Klausurpunkte - Themen: Einführung (Gradle, allgemeine Algorithmen, Anforderung an die Abgaben)

### Aufgabenblatt 2

25 Klausurpunkte - Themen: Einfach und doppelt verkettete Liste, Sortierverfahren und Komplexitätsberechnung

### Aufgabenblatt 3

15 Klausurpunkte - Themen: Adjazenzliste, Adjazenzmatrix, Graphen und ADTs

### Aufgabenblatt 4

Entfällt

### Aufgabenblatt 5

10 Klausurpunkte - Themen: Lineares Sondieren, Quadratisches Sondieren und Double Hashing

### Aufgabenblatt 6

30 Klausurpunkte - Themen: Verlustfreie Kompressionsalgorithmen

Die Aufgabenblätter haben ein eigenes Punktesystem. Es sind keine Klausurpunkte (KP). Wenn Sie 100 % des Aufgabenblatts erfüllt haben, dann bekommen Sie die maximal möglichen Klausurpunkte pro Aufgabenblatt. Die maximal möglichen Klausurpunkte stehen auf dem jeweiligen Aufgabenblatt am Ende. Hier ein kurzes Rechenbeispiel der maximal möglichen Klausurpunkte pro Aufgabenblatt:

- Wenn Sie z.B. beim Aufgabenblatt 1 die volle Punktzahl (100%) erreichen, dann bekommen Sie die vollen 15 Klausurpunkte.
- Wenn Sie z.B. beim Aufgabenblatt 2 die Hälfte der angegebene Punktzahl (50%) erreichen, dann bekommen Sie 12,5 Klausurpunkte von 25 maximal möglichen Klausurpunkten.
- Wenn Sie z.B. beim Aufgabenblatt 3 die volle Punktzahl erreichen, dann bekommen Sie die 15 Klausurpunkte.

In Summe können Sie aus der Laboreinheit und dem seminaristischen Unterricht 100

Klausurpunkte erreichen. Bei mir können Sie genau die Hälfte der Klausurpunkte (also 50 KP) erreichen. Die restlichen 50 Klausurpunkte bekommen Sie von Prof. Sieck in einer mündlichen oder schriftlichen Prüfung. Wenn Sie alle Aufgabenblätter in der Laboreinheit mit 100 % absolvieren, dann haben Sie die maximal möglichen 50 Klausurpunkte (KP) bei mir aus der Laboreinheit erreicht.