Heinrich Heine Universität Düsseldorf

**Programmierung – WS 2025/26**
Dr. Markus Brenneis
Abgabefrist für Feedback:
02.02.26, 10 Uhr

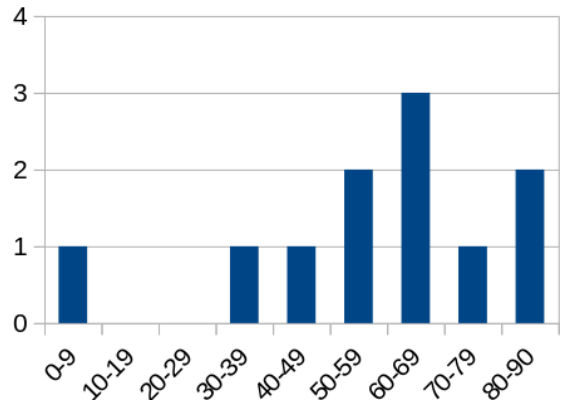
# 14. Übungsblatt (Lösungsvorschlag[1])

Es gibt kein Übungsblatt 15. Stattdessen werden wir in den nächsten Tagen im Download-Bereich für die Übungsblätter eine Probeklausur hochladen, die Sie zuhause bearbeiten können.

## Aufgabe 1: Histogramm nach VL 24

Ein Histogramm ist ein Diagramm, aus dem man die Häufigkeitsverteilung von in Klassen (*Bins*) eingeteilten Daten ablesen kann. Zum Beispiel könnte es bei einer Klausur mit max. 90 Punkten folgende Ergebnisse gegeben haben: 31, 58, 64, 90, 0, 49, 51, 79, 67, 63, 87

Um eine bessere Übersicht zu bekommen, wie die Punkte verteilt sind, teilen wir sie in die 9 Bins der Breite 10 ein: $[0;10)$, $[10;20)$, …, $[70;80)$, $[80;90]$. Auf diese Bins entfallen jeweils 1, 0, 0, 1, 1, 2, 3, 1 bzw. 2 Punktzahlen. Grafisch darstellt ergäbe sich folgendes Diagramm:




Wir wollen nun ein Programm `Histogram` schreiben, das die Höhe der Balken, also die Anzahl der Elemente in einem Bin, berechnet. Als **Argumente** von der Konsole nimmt das Programm drei Integer-Werte `min`, `max` und `n` entgegen. Der Bereich zwischen `min` und `max` soll dann in `n` gleich große Bins aufgeteilt werden (wobei der letzte Bereich um 1 größer ist). Im Beispiel oben sind min=0, max=90 und n=9.

Ihr Programm soll Zahlen einlesen, die auf die Bins verteilt werden. Das Einlesen dieser Zahlen kann auf zwei unterschiedliche Arten erfolgen:

[1] Bei den meisten Programmieraufgaben gibt es mehr als einen funktierenden Lösungsweg. Diskutieren Sie gerne untereinander Ihre Lösungsansätze und lerne Sie damit verschiedene Lösungsstrategien und verschiedene Anwendungsmöglichkeiten der Java-Funktionalitäten kennen. Ihre Abgabe müssen Sie aber final selbst formulieren/eintippen.

- Wenn ein viertes Argument angegeben ist, wird dieses als Dateiname interpretiert.
- Sind nur drei Argumente gegeben, werden Daten über die Standardeingabe erwartet.

In beiden Fällen erwartet das Programm eine Eingabe-Zeile, in der eine beliebige Anzahl ganzer Zahlen mit Leerzeichen getrennt stehen.

Ihr Programm soll in einer Zeile auf der Standardausgabe ausgeben, wie viele Zahlen in welchen Bereich fallen. Mit den Zahlen aus obigem Beispiel soll die Programmausgabe wie folgt aussehen:

```
% cat punkte.txt
31 58 64 90 0 49 51 79 67 63 87
% java Histogram 0 90 9 punkte.txt
1 0 0 1 1 2 3 1 2
% java Histogram 0 90 9 < punkte.txt
1 0 0 1 1 2 3 1 2
```

*Hinweis: Die Ausgabe darf mit einem Leerzeichen enden.*

Hierbei ist zu beachten:

- In folgenden Fällen soll eine Fehlermeldung ausgegeben werden, die `ERROR` enthält, und das Programm beendet werden:
  - weniger als drei Argumente übergeben
  - wenn ein Dateiname angegeben wurde: die Datei ist nicht lesbar
  - `max - min` nicht restlos durch `n` teilbar
  - `n` kleiner 1
  - `min` größer als `max`
  - eine Zahl, die kleiner als `min` oder größer als `max` ist, wird eingelesen
- Ein Bin schließt seine untere Grenze mit ein, jedoch nicht seine obere.
- Der letzte Bin schließt sowohl seine untere, als auch seine obere Grenze mit ein.
- `min` und `max` dürfen auch negativ sein.
- Neben `main` vorhandene Methoden sind privat.

## Aufgabe 2: Streams nach VL 27

Mit Streams haben Sie eine neue Möglichkeit gelernt, um Collections zu verarbeiten. Erfahrene Java-Entwickler:innen nutzen Streams in modernem Java-Code, um Daten zu filtern, transformieren, auszugeben usw. Stream-Ausdrücke sind für diese Aufgaben also häufig die idiomatische Lösung.

Bearbeiten Sie die Selbsttests zum Thema Lambda-Ausdrücke und Streams auf der Kursseite, um für die Java-Praxis und die Klausur vorbereitet zu sein.

## Aufgabe 3: Classpath nach VL 26

*Für diese Aufgabe ist es* nicht *notwendig, die Implementierung der Klassen nachzuvollziehen.*

Wir wollen wiederholen, wie der Classpath zusammengesetzt werden kann und wie uns Build-Systeme dabei die Arbeit erleichtern.

1. Arbeiten Sie mit Codevorgabe 1.
   (a) Compilieren Sie die Klasse in der Datei `src/main/java/planer/WgPlan.java`, sodass die zugehörige class-Datei entsteht. Zum Compilieren und Ausführen wird die Datei `jars/pdfbox-2.0.33.jar` benötigt.
   (b) Führen Sie die `main`-Methode der Klasse mit den Argumenten `out.pdf 9 kristin 70 christian 30` aus. Zum Ausführen werden zusätzlich die jar-Dateien für `commons-logging` und `fontbox` benötigt.
   Wenn der Aufruf korrekt ist, entsteht eine Datei `out.pdf`.
   (c) Compilieren Sie die Test-Klasse in der Datei `src/test/java/planer/SchedulerTest.java`. Zum Compilieren und Ausführen werden `src/main/java/planer/Scheduler.java` sowie die jar-Dateien für `junit-jupiter-api`, `assertj-core` und `apiguardian-api` benötigt.
   (d) Führen Sie die Tests mithilfe von `java -cp ?? org.junit.platform.console.ConsoleLauncher execute --scan-class-path` aus; setzen Sie dafür den Classpath richtig. Die Klasse `org.junit.platform.console.ConsoleLauncher` befindet sich in der jar-Datei `junit-platform-console-standalone`. Außerdem werden die Compilate der java-Dateien und `assertj-core` benötigt.
   Wenn alles richtig ist, sollte unter anderem `8 tests found` ausgegeben werden.
2. Arbeiten Sie mit Codevorgabe 2.
   (a) Führen Sie `./gradlew run --args "out.pdf 9 kristin 70 christian 30"` aus. Wenn alles richtig ist, entsteht eine Datei `out.pdf`.
   (b) Führen Sie `./gradlew test` aus. Wenn alles richtig ist, sollte `BUILD SUCCESSFUL` ausgegeben werden.
   (c) Woher weiß der Befehl `./gradlew`, welche zusätzlichen jar-Dateien zum Compilieren und Ausführen von Produktiv- und Testcode benötigt werden?

## Aufgabe 4: Programmier-Werkzeuge & Vorbereitung aufs Propra 1 nach VL 28

In den letzten Vorlesungen haben Sie einige Werkzeuge kennengelernt, die uns den Programmieralltag erleichtern. In den Übungsstunden 14 und 15 werden Sie darüber reden, welches Werkzeug es für welchen Zweck gibt und die Verwendung nochmal praktisch üben.

Wenn Sie das Programmierpraktikum 1 im nächsten Semester belegen, ist es von Vorteil, wenn Sie vorm Start des Praktikums auf Ihrem Rechner eine funktionierende IDE und git-Installation eingerichtet haben und wissen, wie man ein gradle-Projekt in die IDE importiert und ausführt. Üben Sie das jetzt schon mit dem Material aus den Vorlesungen 26–28 und stellen Sie Fragen im Forum, Tutorium oder in den Übungen.

Im Programmierpraktikum werden wir auf den Inhalten der Programmierung aufbauen. Auf der Kursseite haben wir für Sie Verweise auf Inhalte der Programmierung zusammengestellt, die für das Programmierpraktikum besonders wichtig sind.