Übungsblatt 1

Abgabe: nein

Aufgabe 1 Einmal mit LaTeX, bitte

Erweitert die Datei *loesung01.tex* so, dass LaTeX eine PDF-Ausgabedatei erzeugt, die der zweiten Seite dieses Übungsblatts gleicht. Einzig die Felder <Tutor:in> und <Bearbeiter:in> sollen durch die für euch passenden Daten ersetzt werden.

Folgende Konstrukte sollen eingesetzt werden:

- Hauptüberschriften
- Normaler Text
- Einbinden von externen Quelltextabschnitten. Verweist hierzu auf die beigelegte Datei *Primes.java*[1].
- Fetter Text
- Hervorgehobener Text. Dieser erscheint mit der verwendeten Vorlage kursiv.
- Querverweise
- Formelsatz
- Literaturverweis und Literaturliste mit BibTeX. Bindet hierzu die beigelegte Literaturdatei *Referenzen.bib* ein und verwendet den Literaturstil *gerplain*.
- Fließende Umgebung mit Beschriftung
- Tabellensatz

Tipp. Beachtet, dass der Quelltext für Querverweise mehrfach übersetzt werden muss. Für das Literaturverzeichnis muss zusätzlich auch noch BibTeX ausgeführt werden (*pdflatex*, *bibtex*, *pdflatex*, *pdflatex*). Verwendet ihr *TeXstudio*, werden diese Schritte automatisch ausgeführt.

[1] Wer möchte, kann das enthaltene Programm tatsächlich in BlueJ ausprobieren. Dazu die Datei *package.bluej* mit eben diesem öffnen. Über das Kontextmenü von *Primes* könnt ihr *void primes()* ausführen. Der gebogene Pfeil rechts unten im BlueJ-Fenster beendet das Programm wieder. Nichts davon ist aber für das Bearbeiten dieses Übungsblatts notwendig.

Praktische Informatik I | WiSe 2025/26

Tutor:in: <Tutor:in> | Bearbeiter:in: <Bearbeiter:in>

Übungsblatt 1

Lösungsvorschlag

Aufgabe 1 Primzahlen berechnen

Primzahlen kann man einfach berechnen, indem man eine Folge aller Zahlen ab 2 erzeugt,

```
10      static void primes()
11      {
12          IntStream.iterate(2, i -> i + 1)
```

nur Zahlen behält, die nicht durch kleinere Zahlen teilbar sind,

```
13                  .filter(i -> IntStream.range(2, i).noneMatch(j -> i % j == 0))
```

und die verbleibenden Zahlen ausgibt.

```
14                  .forEach(System.out::println);
15      }
```

Test. Nach einem Aufruf von *Primes.primes()* wird jeweils eine Zahl pro Zeile ausgegeben. Die Zahlenfolge *2, 3, 5, 7, 11, 13, 17* usw. sieht korrekt nach Primzahlen aus.

Aufgabe 2 Verbesserungen

Auch wenn das Programm aus Aufgabe 1 schnell ist, merkt man, dass es für größere Primzahlen immer langsamer wird. Folgende Verbesserungen sind denkbar:

- Man könnte die 2 separat ausgeben und danach nur noch ungerade Zahlen erzeugen.
- Man muss eigentlich nur testen, ob eine Zahl i durch keine Zahl $j \in [2 \dots \lfloor\sqrt{i}\rfloor]$ teilbar ist.
- Bei bekannter Obergrenze kann man das Sieb des Eratosthenes [1] verwenden (s. Tab. 1).

	2	**3**	4
5	6	**7**	8
9	10	**11**	12
13	14	15	16

Tabelle 1: Sieb des Eratosthenes. Die fett dargestellten Zahlen wurden nicht weggestrichen.

Literatur

[1] Möhring, Rolf H. und Martin Oellrich: *Das Sieb des Eratosthenes: Wie schnell kann man eine Primzahlentabelle berechnen?* In: *Taschenbuch der Algorithmen*, Seiten 127–138. 2008.