Prof. Dr. Christoph Bockisch
MSc. Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Übungen zur Vorlesung
Objektorientierte Programmierung: Wintersemester 2022/2023

Nr. 7, Abgabe bis 12.12.2022

Aufgabe 7.1: Optimus Prime — 10 Punkte

Professor Doktor Abdul Nachtigaller hat eine neuartige Idee zur Berechnung von Primzahlen. Er geht davon aus, dass man Primzahlen anhand von anderen Primzahlen bestimmen kann. Dafür schreibt er sich zunächst alle Zahlen von 0 bis zu einer gewünschten Zahl, n, auf. Da er bereits weiß, dass 0 und 1 keine Primzahlen sind, streicht der diese in seiner Liste durch. Als nächstes streicht er alle multiplen von 2, der ersten Primzahl, aus seiner Liste heraus. Nun nimmt er sich die nächste Zahl, die er noch nicht gestrichen hat, und streicht alle multiplen dieser Zahl aus seiner Liste heraus. Dies macht er für alle Zahlen, die kleiner als $\sqrt{n} + 2$ sind. Alle übrigen Zahlen müssen dann, nach seiner Theorie, Primzahlen sein.

Denken Sie daran: Wissen ist Nacht!

a) Implementieren Sie das Verfahren von Professor Nachtigaller. Schreiben Sie hierfür eine Methode `int[] primesUpTo(int n)`, die ein Array aller Primzahlen bis einschließlich n als Ergebnis liefert. — **8**

   Stellen Sie sicher, dass keine negativen Zahlen für n akzeptiert werden.

b) Testen Sie Ihre Methode mit den Primzahlen bis 100. Sie können `int[] primes = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}` zum Testen verwenden. — **2**

Aufgabe 7.2: Power Rangers: Turbo — 14 Punkte

Aus der Vorlesungen ist Ihnen sicher Fibonacci bekannt. Die Fibonacci-Folge ist ein bekanntes Beispiel für baumartige Rekursion. Die Berechnungsformel für Fibonacci (kurz fib) lautet wie folgt:

$$fib(n) = \begin{cases} 0 & , n = 0 \\ 1 & , n = 1 \\ fib(n-1) + fib(n-2) & , sonst \end{cases}$$

Implementieren Sie die folgenden Methoden in einer Klasse `Fibo`. Stellen Sie sicher, dass sämtliche Methoden aufgerufen werden können **OHNE** ein Objekt der Klasse zu erzeugen.

a) Implementieren Sie eine Methode `int fiboTree(int n)`, die den Fibonacci nach der oben genannten Formel rekursiv berechnet. **2**

b) Schreiben Sie nun eine Methode `int fiboFast(int n)`, die eine Hilfsmethode `fiboHelp(int n, int a, int b)` mit geeigneten Werten aufruft. **2**

   Die Parameter sollen dabei wie folgt belegt sein:

   - `n` Die Position in der Fibonacci Folge
   - `a` Den größeren Fibonacci (n-1)
   - `b` Den kleineren Fibonacci (n-2)

c) Implementieren Sie nun die rekursive Hilfsmethode `fiboHelp`. Stellen Sie sicher, dass sämtliche `return`-Statements lediglich `a`, `b` oder einen Aufruf von `fiboHelp` enthalten. **6**

   Achten Sie bei Ihrer Implementierung darauf, dass `fiboHelp` nicht außerhalb der Klasse aufrufbar ist.

d) Implementieren Sie eine Methode `measureAndTest(int n)`, die die Laufzeit beider Implementierungen misst und auf der Konsole ausgibt. Die Methode soll weiterhin die Ergebnisse der Methoden zwischenspeichern und vergleichen. Geben Sie anschließend auf der Konsole aus, ob beide Ergebnisse gleich waren. **2**

   Hinweis: Mit `long System.currentTimeMillis()` können Sie die aktuelle Systemzeit erhalten.

e) Verwenden Sie die Methode `measureAndTest`, um Ihre Implementierungen zu messen und zu testen. Implementieren Sie hierzu eine `main`-Methode, die `measureAndTest` mit **geeigneten** Werten aufruft. **2**

Hinweis: Sie können für diese Aufgabe automatisch generiertes Feedback unter `https://pro.quarterfall.com/do/cphwylfe` erhalten.