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

Philipps Universität Marburg

Übungen zur Vorlesung
Objektorientierte Programmierung: Wintersemester 2021/2022

Nr. 6, Abgabe bis 06.12.2021

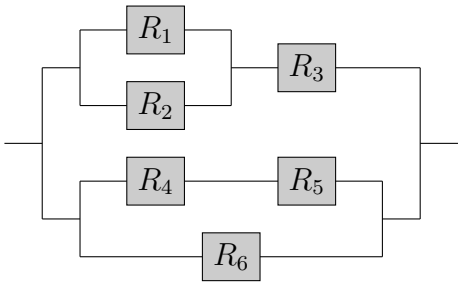
Aufgabe 6.1: Tron: Uprising — 3 Punkte

Der Widerstand R eines Schaltkreises besteht aus zwei Widerständen, R_1 und R_2. Ein Schaltkreis kann auf zwei unterschiedliche Arten geschaltet werden und kann wie folgt berechnet werden:

$$R = R_1 + R_2 \text{ Reihenschaltung (Seriell)}$$
$$R = \frac{R_1 * R_2}{R_1 + R_2} \text{ Parallelschaltung}$$

a) Implementieren Sie eine Methode `double seriesCircuit(double rOne, double rTwo)`, welche den Widerstand einer übergebenen Reihenschaltung berechnet.
b) Implementieren Sie eine Methode `double parallelCircuit(double rOne, double rTwo)`, welche den Widerstand einer übergebenen Parallelschaltung berechnet.
c) Testen Sie Ihren Code, indem Sie den Widerstand des untenstehenden Schaltkreises berechnen. Verwenden Sie dabei die folgenden Widerstände:

$$R_1 = 60\Omega,\ R_2 = 40\Omega,\ R_3 = 50\Omega,\ R_4 = 50\Omega,\ R_5 = 70\Omega,\ R_6 = 80\Omega$$



Aufgabe 6.2: Optimus Prime | 5 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.

   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.

Aufgabe 6.3: El Psy Congroo | 4 Punkte

Der verrückte Wissenschaftler Hououin Kyouma benötigt für seine Forschung eine Methode zum Berechnen der k-ten Wurzel. Im Allgemeinen kann die k-te Wurzel einer Zahl durch das Verfahren von Heron wie folgt beschrieben werden:

$$\sqrt[k]{a} = \begin{cases} x_0 = a \\ x_{n+1} = \frac{1}{k} * ((k-1) * x_n + \frac{a}{x_n^{k-1}}) \end{cases}$$

Das Verfahren bricht dann ab, wenn $|x_n - x_{n+1}| < d$, wobei `d` für Delta steht. In diesem Fall bedeutet es also, dass es sich um eine Schranke handelt, unter die der Abstand zwischen zwei Berechnungsschritten fallen muss, damit der Algorithmus fertig sein kann.

a) Schreiben Sie eine Methode `double krt(double a, double k, double d)`, die das Verfahren von Heron implementiert. Diese Methode soll lediglich eine rekursive Hilfsmethode `double krtH(double a, double k, double d, double x_n)` mit geeigneten Werten aufrufen.

   Hinweis: Sie sollten in `x_n` immer den aktuell berechnete Wert speichern. Fangen Sie negative Eingaben bereits in *krt* ab und geben 0 als Ergebnis zurück.

b) Implementieren Sie die Methode `double krtH(double a, double k, double d, double x_n)`, sodass sie Herons Verfahren, wie oben beschrieben, folgt.

c) Testen Sie *krt* mit mindestens drei verschiedenen, positiven Werten für *a*, *k* und *d*.