Aufgabenblatt 4

Themen

- Polymorphie
- Dokumentation

Antestat

- Laden Sie Ihre Modellierung vor Beginn des Praktikums in Ilias hoch. Achten Sie auf eine korrekte Umsetzung der UML-Notation.
- Ihre Modelle werden geprüft und Sie erhalten eine Rückmeldung.

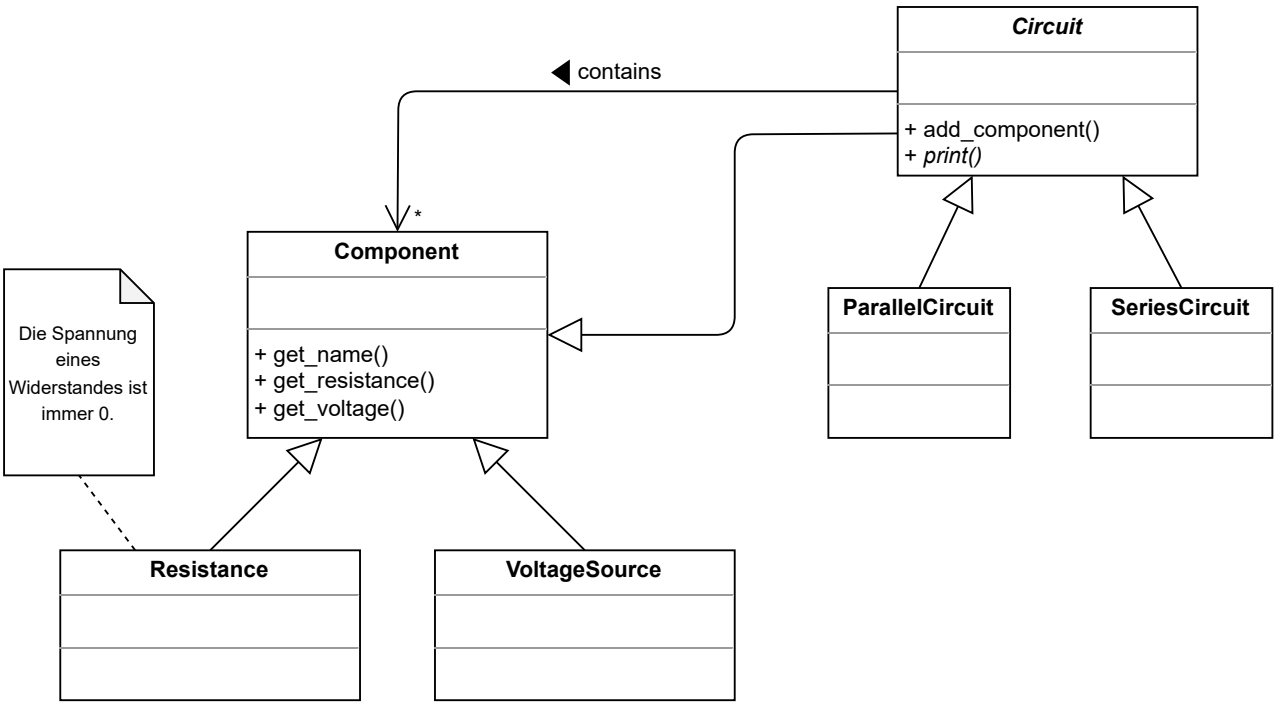

Abbildung 1: Klassendiagramm zur Realisierung einer komplexen Schaltung.

Aufgabe 1 - Modellierung: Komplexe Schaltung

Die Modellierung von Aufgabenblatt 3 erlaubt keine Verschachtelung von Schaltkreisen. Um dies zu ermöglichen, wird die Klasse `Circuit` verändert: Diese erbt nun von `Component`.

a) Erweitern bzw. verändern Sie Ihr Modell von Aufgabenblatt 3 gemäß Abbildung 1.

b) Bisher wurde nur der Widerstandswert von `Component` betrachtet. Das Modell soll nun um *Spannung* und *Stromstärke* erweitert werden. Erweitern Sie Ihr Modell um geeignete Felder und Methoden.

> **Hinweis:** In unserem Modell sind die Werte für Widerstand, Spannung und Stromstärke *berechnete Größen*. Das bedeutet, dass sie nicht direkt gespeichert, sondern aus anderen Werten abgeleitet werden. Solche berechneten Werte besitzen daher ausschließlich einen *getter*, aber keinen *setter*.
> Es gibt zwei unterschiedliche Ansätze, wie und wann diese Werte berechnet werden können: **Variante 1:** Der Wert wird bei jeder Änderung relevanter Eingangsgrößen einmal berechnet und in einer *Member-Variable* gespeichert. Der *getter* gibt anschließend diesen gespeicherten Wert zurück. **Variante 2:** Der Wert wird bei jeder Abfrage im *getter* (z. B. `get_resistance()`) neu berechnet. Eine separate Speicherung in einer *Member-Variable* ist hier nicht notwendig.
> Variante 2 folgt stärker den Prinzipien der objektorientierten Programmierung und ist in den meisten Fällen vorzuziehen. (Warum?)

Aufgabe 2 - Implementierung & Berechnung von Ersatzspannung und Ersatzstrom

a) Implementieren Sie Ihr Modell.

- Verwenden Sie für die Berechnung folgende (idealisierte) Gesetzmäßigkeiten:
  - Reihenschaltung: $U_{ersatz} = U_1 + U_2 + \cdots + U_n$
  - Parallelschaltung: $U_{ersatz} = \max(U_1, U_2, \ldots, U_n)$[1]

  und den Ersatzstrom I_{ersatz}:
  - Reihenschaltung: $I_{ersatz} = I_1 = I_2 = \cdots = I_n$
  - Parallelschaltung: $I_{ersatz} = I_1 + I_2 + \cdots + I_n$
- Passen Sie die Ausgaben an, sodass alle Informationen textuell auf der Konsole ausgegeben werden. Die Struktur des Schaltkreises soll dabei deutlich werden, sowie alle Ersatzwiderstände, -spannungen und -ströme.
- (optional) (anspruchsvoll) Erstellen Sie eine grafische Ausgabe auf der Konsole.
  Tipp: Beschränken Sie sich dabei auf einfache Schaltungen, z. B. maximal 2 Sub-Schaltkreise.

b) Implementieren Sie folgende Testfälle:

1. Testschaltung 1
2. Testschaltung 2
3. eine selbst erstellte Schaltung mit mindestens 1 Reihen- und 2 Parallel-Schaltungen und 1 Spannungsquelle. Erstellen Sie eine Skizze inklusive aller (berechneten) Werte dieser Schaltung, sodass die Ausgabe Ihres Programms nachvollziehbar ist.
4. (optional) selbst gewählte Testfälle, die die Grenzen und Besonderheiten des Programms aufzeigen. Zum Beispiel mehrere Spannungsquellen, „sinnlose“ Schaltungen, Schaltungen mit Extremwerten, ...

[1] Achtung: Vereinfachung! Wir erheben in diesem Praktikum keinen Anspruch auf physikalische Korrektheit.

Aufgabe 3 - Dokumentation

Das Tool Doxygen[2] erlaubt es, aus formatierten Kommentaren im Quelltext automatisiert eine Dokumentation zu erzeugen.

Fügen Sie den Klassendefinitionen (`.hpp`) eine aussagekräftige Dokumention hinzu. Beschreiben Sie die Aufgaben der Klassen und ihrer Methoden und Konstruktoren und geben Sie Hinweise zu deren korrekten Parametrisierung. Konvertieren Sie anschließend diese Dokumentation mit Hilfe von Doxygen nach HTML.

Abtestat

- Demonstrieren Sie Ihr Programm anhand der Testfälle.
- Erläutern Sie anhand Ihres Codes und der Dokumentation die Begriffe *Vererbung*, *überschreiben*, *abstrakt*, *Polymorphie*.

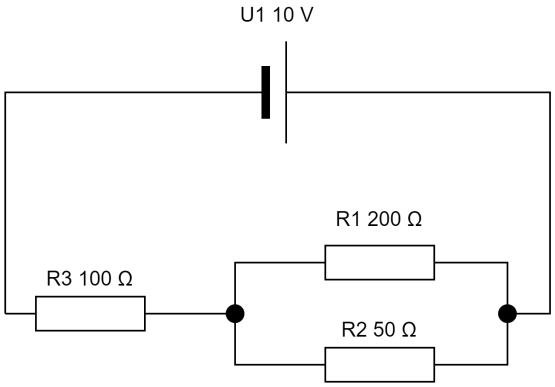

Abbildung 2: Testschaltung 1

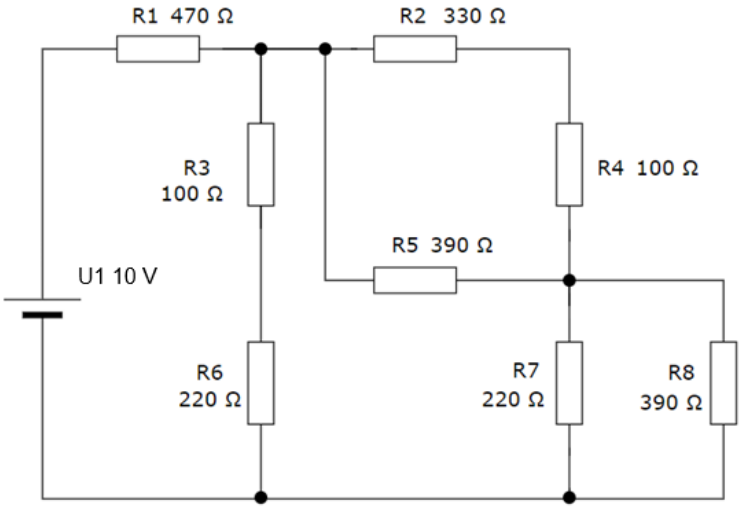

Abbildung 3: Testschaltung 2

[2] `https://www.doxygen.nl`