6. Praktische Übung in "Programmieren": RLC-Schaltung

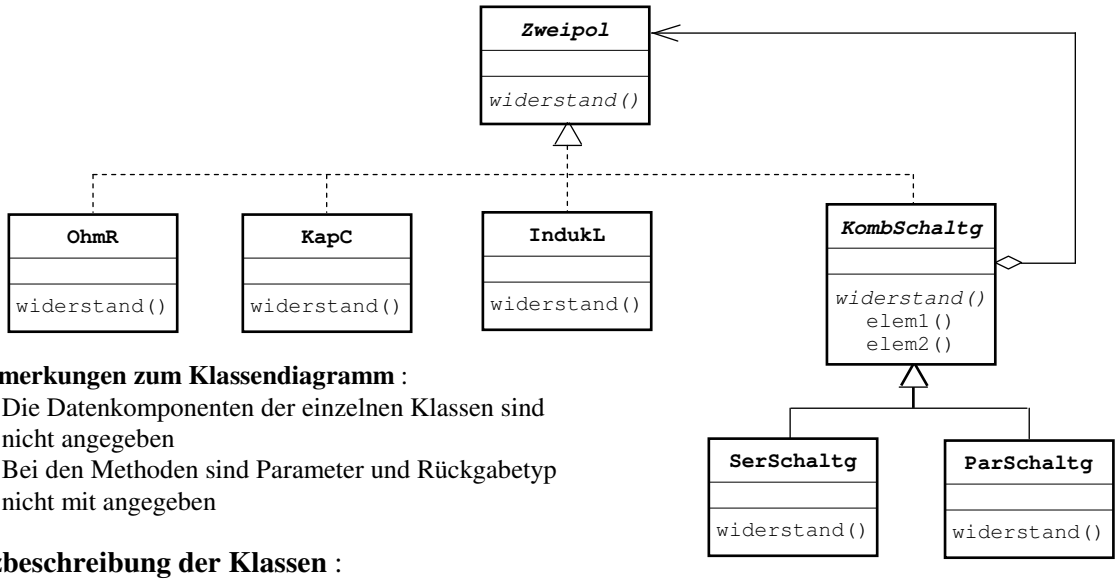
Es soll ein **Java-Projekt** realisiert werden, mit dem Berechnungen an **beliebigen linearen RLC-Schaltungen** durchgeführt werden können. Im vorliegenden Fall sollen die Berechnungen auf die Ermittlung des – komplexen – (Eingangs-)**Widerstands** beschränkt sein Der Projektname ist frei wählbar.
RLC-Schaltungen bestehen aus **Ohmschen Widerständen**, **Induktivitäten** und **Kapaziäten**, die in **Parallel-** und **Serien-Schaltungen** miteinander verbunden sind.

- **Interface `Zweipol`**
  Programmtechnisch lassen sich sowohl die verschiedenen Einzelelemente als auch ihre parallele und serielle Zusammenschaltung durch **geeignete Klassen** beschreiben, die alle das **Interface `Zweipol`** implementieren. Dieses Interface deklariert für die hier vorgesehene Anwendung lediglich die **Methode**

  ```
  Complex widerstand(double freq);
  ```

  In den das Interface implementierenden Klassen ermittelt diese Methode den i.a. von der Frequenz abhängigen (Parameter `freq`) komplexen Widerstand des jeweiligen Bauelements bzw der Zusammenschaltung.
  Die Klasse `Complex` ist die in der 5. Übung erstellte Klasse zur Darstellung komplexer Zahlen.

  Das folgende Klassendiagramm beschreibt den Zusammenhang zwischen dem Interface `Zweipol` und den implementierenden zur Schaltungsbeschreibung eingesetzten Klassen :



Anmerkungen zum Klassendiagramm :
1. Die Datenkomponenten der einzelnen Klassen sind nicht angegeben
2. Bei den Methoden sind Parameter und Rückgabetyp nicht mit angegeben

- **Kurzbeschreibung der Klassen** :
  - **`OhmR`** : Modellierung eines Ohmschen Widerstandes, Widerstandswert ist Konstruktorparameter
  - **`KapC`** : Modellierung einer Kapazität, Kapazitätswert ist Konstruktorparameter
  - **`IndukL`** : Modellierung einer Induktivität, Induktivitätswert ist Konstruktorparameter
  - **`KombSchaltg`** : Abstrakte Klasse, keine Implementierung der Methode `widerstand()`
    Datenkomponenten : 2 private Referenzen auf `Zweipol`−Objekte (Konstruktor-Parameter), der Konstruktor ist `protected`.
    Implementierte Memberfunktionen : `elem1()` und `elem2()`, sie geben die gespeicherte Referenz auf das erste bzw zweite `Zweipol`−Objekt zurück
  - **`SerSchaltg`** : Modellierung einer Serienschaltung aus zwei Elementen (die ihrerseits auch eine Serien- bzw Parallel-Schaltung sein können)
    Die Elemente werden dem Konstruktor als Parameter übergeben (in Form von `Zweipol`−Referenzen). Dieser ruft mit ihnen den Basisklassen-Konstruktor auf.
  - **`ParSchaltg`** : Modellierung einer Parallelschaltung aus zwei Elementen, übrige Beschreibung analog zur Klasse `SerSchaltung`.

a) **Realisieren** Sie das **Interface `Zweipol`** sowie alle o.a. **Klassen**.
Alle Klassen sowie das Interface sollen sich in einem Package mit dem Namen **`prakt.v6`** befinden.

weiter Seite 2

Jede nur aus den obigen Elementen (R, L, C, Serienschaltung, Parallelschaltung) bestehende **Schaltung** kann programmintern als **`Zweipol-Objekt`** dargestellt und referiert werden.

- Zur externen – für die **Programmeingabe** eingesetzten – **Schaltungsbeschreibung** soll eine spezielle "Beschreibungssprache" dienen, die durch die folgende **Syntax** definiert ist (Backus-Naur-Form) :

  schaltung **::=** *schaltelement* **E**
  schaltelement **::=** *einzelelement* | *serschaltung* | *parschaltung*
  einzelelement **::=** **R** *wert* | **L** *wert* | **C** *wert*
  serschaltung **::=** **S** *schaltelement* {*schaltelement*} **E**
  parschaltung **::=** **P** *schaltelement* {*schaltelement*} **E**
  wert **::=** Gleitpunktzahl

  Beispiele :
  ```
  R 333.33 E
  S R 100 P L 4.5e-3 C 6.2e-6 E E E
  P S R 250 C 4.7e-9 E S R 1000 L 9.3e-3 E R 570 E E
  ```

- **Klasse `SchaltgBuilder`**
  Diese zur Verfügung gestellte Klasse (enthalten in der Datei **`SchaltgBuilder.jar`**) dient zum **Einlesen** einer in obiger Form vorliegenden **Schaltungsbeschreibung** und **Erzeugung** eines entsprechenden **`Zweipol-Objekts`**. Sie befindet sich ebenfalls im Package **`prakt.v6.`**
  Die Klasse setzt zum Einlesen ein **`Scanner-Objekt`** ein, das ihrem **Konstruktor** zu übergeben ist.
  Das Einlesen einer Schaltung und den Aufbau des `Zweipol`-Objekts realisiert die Methode :

  `public Zweipol read()`

  Sie gibt eine Referenz auf das **erzeugte `Zweipol-Objekt`** als **Funktionswert** zurück bzw **`null`** bei **Eingabeende**.

  Hinweis : Die ebenfalls zur Verfügung gestellte Datei **`SchaltgBuilder.java`** dient nur zur Info über die Klassenimplementierung. Sie darf im Projekt **nicht** direkt eingesetzt werden (sondern nur die `.jar`- Datei)

- **Klasse `SchaltgTest`**
  Diese – **unvollständig** zur Verfügung gestellte – Klasse (Datei **`SchaltgTest.java`**) dient zum **Testen** des erzeugten Projekts. Sie soll sich ebenfalls im Package **`prakt.v6`** befinden. Es **fehlen** die **notwendigen Package- und Import-Deklarationen** sowie die **Definition der `main()`-Methode**.

  Die bereits implementierte **private statische Methode**

  `private static Scanner makeScanner(String path)`

  erzeugt ein **Scanner-Objekt**, dessen Eingabequelle
  - die Datei ist, deren Zugriffspfad als Parameter übergeben wurde – falls sie existiert
  - die Standard-Eingabe ist, falls die Datei nicht existiert oder als Parameter die `null`-Referenz übergeben wurde.

  Die noch **fehlende `main()`-**Methode soll
  - einen gegebenenfalls übergebenen Programmparameter als Dateizugriffspfad interpretieren
  - die Methode `makeScanner()` aufrufen,
  - ein `SchaltgBuilder`-Objekt erzeugen
  - und in einer Schleife die Eingabe einer Schaltung anfordern, diese einlesen,
    die Eingabe einer Frequenz anfordern, diese einlesen,
    den komplexen Wiederstand der Schaltung ermitteln und in die Standardausgabe in Real/Imaginär- sowie Polardarstellung ausgeben.

    Ende der Schleife, wenn keine Schaltung mehr eingelesen werden kann (Eingabeende)

b) **Vervollständigen** Sie die Klasse **`SchaltgTest`** durch die notwendigen **Package- und Import-Deklarationen** sowie die **Definition** der Methode **`main().`**

c) **Testen** Sie das Projekt durch **Schaltungs- u. Frequenzeingaben** von der **Standardeingabe** sowie von einer **Datei**. Die zur Verfügung gestellte Datei **`schaltg.txt`** enthält **mehrere Beispiel-Schaltungen** im Wechsel mit der jeweils zu untersuchenden Frequenz.