Aufgabenblatt 2

Themen

- Modellierung von Klassen
- `enum`-Klassen
- Testen

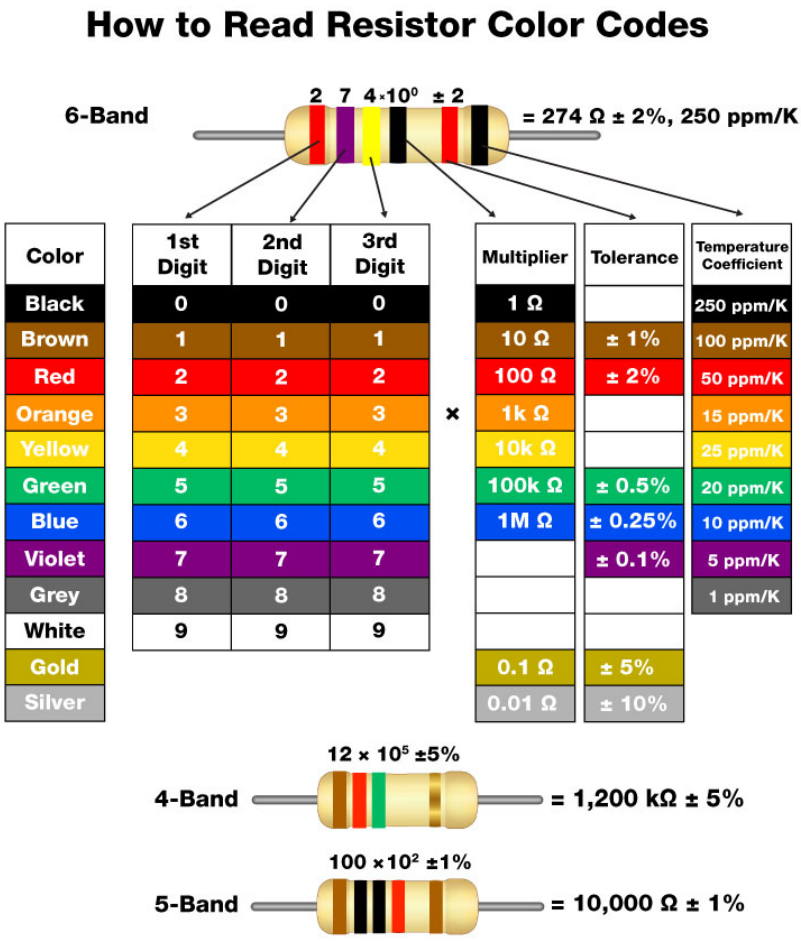

Abbildung 1: Farbencodes Bandwiderstand, Quelle:
`https://www.arrow.de/research-and-events/articles/resistor-color-code`

Antestat

1. **Dokumentation der Modellierung**
   - Erstellen Sie eine UML-Klassendiagram nach den folgenden vorgaben:
     - Entwickeln Sie eine Klasse zur Abbildung eines Bandwiderstands (siehe Abbildung 1).
     - Die Klasse soll Widerstände mit 4, 5 und 6 Bändern untersützen.
     - Klären Sie die notwendige Anzahl an Konstruktoren.
     - Nutzen Sie `enum`-Klassen zur Darstellung der Bänder:
       * Wie viele `enum`-Klassen sind erforderlich?
       * Wo sollten sie implementiert werden?
     - Die Bänder müssen zur Laufzeit **unveränderbar** sein.
     - Fügen Sie ein unveränderbares Attribut `Name` sowie die Methode `ausgabe()` hinzu.
     - Stellen Sie sicher, dass jede Instanz von `BandWiderstand` eine **gültige Kombination** von Farbringen besitzt.

   Hinweis: Der Widerstandswert selbst soll nicht berechnet werden.

2. **Hochladen des Ergebnis**
   - Das UML-Diagram muss vor Beginn des Praktikums in Ilias hochgeladen werden.
   - Die Abgabe wird überprüft, und Sie erhalten eine Rückmeldung.
   - Frühe Einreichung wird empfohlen, um etwaige Mängel rechtzeitig zu korrigieren.

Aufgaben

Trennen Sie bei folgenden Implementierung die Deklaration von der Definition, indem, neben einer Source-Datei, auch eine passende Header-Datei für die `Decoder`-Klasse erstellen.

1. **Implementierung der Klasse** `BandWiderstand`

   Achten Sie bei der Implementierung auf die Trennung von Deklaration und Definition. Verwenden Sie für die Methode `ausgabe()` die Funktion `std::print` (C++ Version 23).

2. **Erstellung der Klasse** `Decoder`
   - Entwickeln Sie eine Klasse `Decoder`, der einen `BandWiderstand` in einen `Widerstand` umwandelt. Nutzen Sie hierfür die Klasse `Widerstand` aus Aufgabenblatt 1.
   - Die Klasse `Decoder` soll eine Instanz der Klasse `BandWiderstand` übernehmen und daraus eine Instanz der Klasse `Widerstand` erzeugen.

3. **Erstellung von Testfällen**
   - Entwickeln Sie umfassende Testfälle zur Evaluation der Klassen.

4. **(Optional) Implementierung des ==-Operators**
   - Fügen Sie den `Equals`-Operator == sinnvoll in `BandWiderstand` hinzu.
   - Entwickeln Sie Testfälle zur Demonstration des Operators.

5. **(Optional) Verbesserung der Klasse** `Decoder`
   - Diskutieren Sie die Vor- und Nachteile der Übergabe eines `BandWiderstand`-Objektes *by reference* vs. *by value*.

- Erläutern Sie die Vorteile einer `static`-Deklaration der Methode zur Umwandlung von `BandWiderstand` zu `Widerstand`.

Abtestat

1. **Demonstration des Modells anhand der Testfälle** (Aufgabe 4).
2. **Abgleich von Implementierung und Modellierung (vgl. Antestat)**
   - Die Implementierung muss der Modellierung entsprechen.
   - Abweichungen sind zu benennen und zu begründen.
3. **Erläuterung von Implementierungsdetails**
   - Beantwortung relevanter Fragestellungen aus Aufgabe 1 und Aufgabe 2 anhand Ihrer erstellten Testfälle.

Hinweis: Hilfestellung bzw. Feedback zu Ihrem Modell können Sie im Tutorium erhalten.