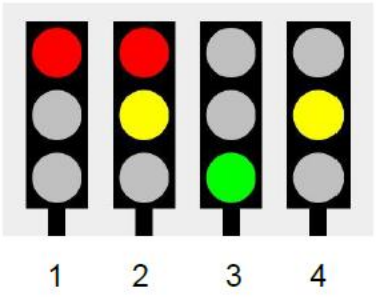
Java13 − Ampel modellieren und implementieren (2/2)

Methoden „`umschalten()`“ und „`ampelzyklus()`“ − Bedingte Anweisung

Wir wollen eine (einzige) Methode „`umschalten()`“ programmieren, bei deren Aufruf die Ampel in den jeweils nächsten Zustand übergeht, unabhängig davon, in welchem Zustand sich die Ampel aktuell befindet.

Bedingte Anweisung „mit mehreren else“ für „`umschalten()`“




```
public void umschalten() {
    if (zustand == 1) {
        rotGelbSetzen();
        zustand = 2;
    } else if (zustand == 2) {
        …
        …
    } …
        …
        …
    } else {                    // else if (zustand == 4) {
        …
        …
    }
}
```

AUFGABE: Methode „`umschalten()`“ programmieren

Öffne Dein Ampel-Projekt und implementiere die Methode „`public void umschalten()`“!

Teste, ob mit jedem Aufruf von „`umschalten()`“ jeweils der richtige nächste Zustand erreicht wird!

AUFGABE: Methode „`ampelzyklus()`“ programmieren

Programmiere eine Methode „`public void ampelzyklus()`“, bei deren Aufruf ein vollständiger Zyklus der vier Ampelphasen abläuft.

Damit man jede Ampelphase ausreichend lange sieht, muss man zwischen den einzelnen Umschalt-Vorgängen eine „Wartezeit“ einbauen. Das gelingt mit „`StaticTools.warte(1000);`“.
Der Parameter (hier: 1000) steht für die Anzahl an Millisekunden, die gewartet werden soll, bis der nächste Befehl ausgeführt wird.

Teste, ob Deine Methode „`ampelzyklus()`“ tatsächlich „tut, was sie soll“!

AUFGABE: Zugriffsmodifikatoren von „`rotSetzen()`“ usw. (wieder) ändern

Ändere den Zugriffsmodifikator der vier Methoden zum Setzen der Farben (wieder) auf „`private`“!

Probiere aus, ob die neuen Methoden „`umschalten()`“ und „`ampelzyklus()`“ noch „funktionieren“!

Was hat sich durch die Änderung der Zugriffsmodifikatoren geändert?

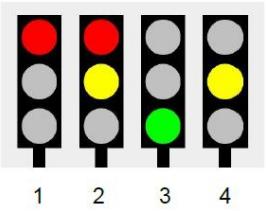
Fortsetzung: Nächste Seite!

Methoden „`umschalten10x()`“ und „`umschaltenNx(int anzahl)`“ – for-Schleife

Damit die Ampelanlage einige Male hintereinander umschaltet, könnte man die Methode „`umschalten()`“ oder auch die die Methode „`ampelzyklus()`“ mehrmals aufrufen. Das geht aber komfortabler:

for-Schleife für „`umschalten10x()`“

```
public void umschalten10x() {
    for (int i = 0; i < 10; i++) {
        umschalten();
        StaticTools.warte(2000);  // 2 Sekunden warten
    }
}
```



In der **`for`**-Schleife wird mit der Variablen **`i`** gezählt, wie oft der Schleifenrumpf ausgeführt werden soll.
Die Sichtbarkeit der Variablen **`i`** beschränkt sich auf den Rumpf der **`for`**-Schleife.
Außerhalb der **`for`**-Schleife kann auf diese Zählvariable nicht zugegriffen werden.
Aus diesem Grund heißt sie **lokale Variable**.

Ablauf der for-Schleife:

1. Zuerst wird die Anweisung `int i = 0` ausgeführt.
   Die lokale Variable **`i`** wird deklariert und mit dem Wert 0 initialisiert.
2. Nun wird geprüft, ob die Bedingung `i < 10` wahr ist.
3. Wenn sie wahr ist, wird der Rumpf der **`for`**-Schleife ausgeführt. Andernfalls wird das Programm mit den nach der **`for`**-Schleife ggf. stehenden Anweisungen fortgesetzt.
4. Nun wird die Anweisung `i++` ausgeführt.
   Der Wert der lokalen Variablen **`i`** wird dadurch um 1 erhöht (inkrementiert).
   Am Ende des Durchgangs wird die **`for`**-Schleife an der Stelle 2. wieder fortgeführt.

AUFGABE: Methode „`umschalten10x()`“ programmieren

Öffne Dein Ampel-Projekt und implementiere die Methode „`public void umschalten10x()`“!

Teste, ob Deine Methode „`umschalten10x()`“ tatsächlich „tut, was sie soll“!

AUFGABE: Methode „`umschaltenNx(int anzahl)`“ programmieren

Bei der Methode „`umschalten10x()`“ ist die Anzahl der Wiederholungen (hier: 10) fest vorgegeben.

Programmiere nun eine Methode „`umschaltenNx(int anzahl)`“, bei deren Aufruf die gewünschte Anzahl der Wiederholungen abgefragt wird. Der beim Methoden-Aufruf übergebene Parameter `int anzahl` kann als lokale Variable direkt in der **`for`**-Schleife verwendet werden, d. h. … `i < anzahl;` …

Teste, ob Deine Methode „`umschaltenNx()`“ tatsächlich „tut, was sie soll“!

ZUSATZ-AUFGABE: Methode „`umschaltenNxReal(int anzahl)`“ programmieren

Bei den bisherigen Methoden war die „Wartezeit“ zwischen den Umschalt-Vorgängen immer einheitlich 2 Sekunden. Das entspricht natürlich nicht der Wirklichkeit.

Programmiere eine Methode „`umschaltenNxReal(int anzahl)`“, bei deren Aufruf die beiden Gelb-Phasen jeweils 1 Sekunde lang sind und die Rot-Phase und die Grün-Phase jeweils 3 Sekunden lang.

Verwende dazu (innerhalb der **`for`**-Schleife) eine bedingte Anweisung, d. h. eine **`if`**-Anweisung.
Als Bedingung dieser **`if`**-Anweisung eignet sich:

```
if (zustand == ? || zustand == ?) { … } else { … }
```

Die beiden senkrechten Striche „`||`“ stehen für die logische Verknüfung „**oder**“. („?“ geeignet ersetzen!)

Teste, ob Deine Methode „`umschaltenNxReal(int anzahl)`“ tatsächlich „tut, was sie soll“!