2.5 Die Mehrfachauswahl

Bevor wir die KREUZUNG zusammensetzen können, müssen wir noch die AMPEL optisch drehen können.
Wir benutzen hierfür ein neues Attribut:

```
char ausrichtung; //char steht für einen einzelnen Buchstaben
```

Wir müssen nun das Attribut im Konstruktor mit einem sinnvollen Startwert belegen. Beim zuweisen eines Wertes an einen „Character“ `char` (engl. Zeichen) muss man einfache Anführungsstriche benutzen:

```
public AMPEL()
{
     ausrichtung = 'S';  //Ausrichtung nach Süden (unten)
}
```

Wenn wir nun die Ausrichtung setzen wollen brauchen wir eine neue Methode, die dem Nutzer eine Eingabe des neuen Wertes ermöglicht. Außerdem sollten die LAMPEN der Ampel richtig zueinander gedreht werden.

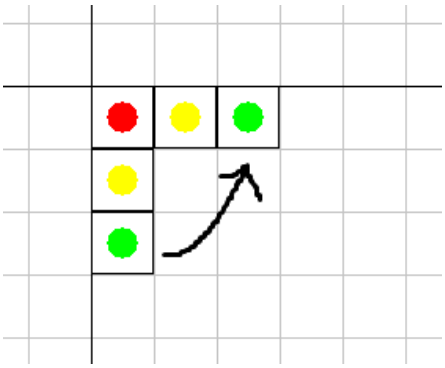
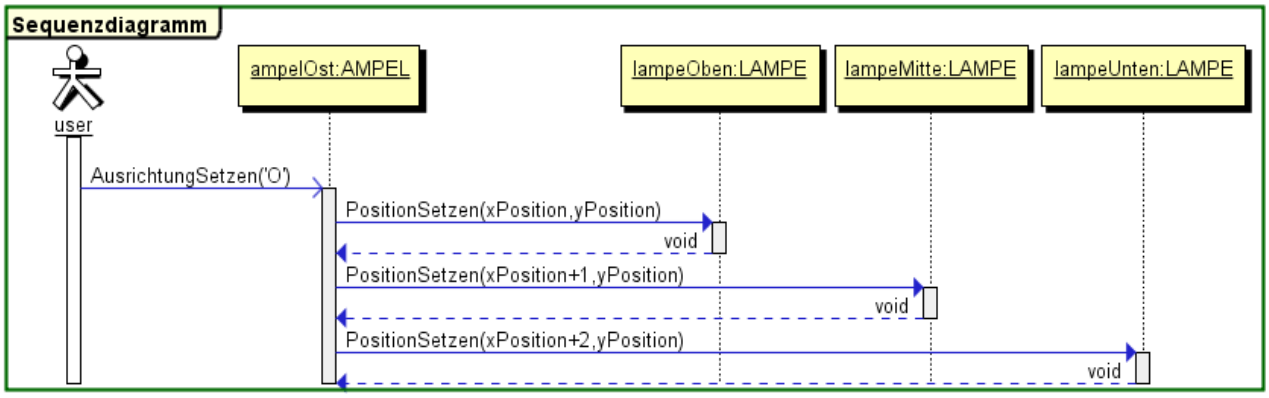


Abb 1: Drehen der AMPEL

Dabei muss die AMPEL in die jeweilige Himmelsrichtung 'N', 'S', 'W', 'O' schauen.
Schnell wird klar, dass die Startposition bisher 'S' war.

Wir überlegen uns, wie die Positionen der LAMPEn sich zueinander verändern muss, wenn die AMPEL um die obere LAMPE gedreht wird.

Betrachten wir das Sequenzdiagramm beispielhaft für 'O':



Drehungen in andere Richtungen gehen analog.

Java Code:

```
lampeOben.PositionSetzen(xPosition,yPosition);
lampeMitte.PositionSetzen(xPosition+1,yPosition);
lampeUnten.PositionSetzen(xPosition+2,yPosition);
```

Jetzt müssen wir beachten, dass die Drehung vom User abhängt. Daher müssten wir eigentlich umsetzen.

```
WENN 'O'
DANN „Drehe nach Osten“
SONST
        WENN 'W'
        DANN „Drehe nach Westen“
        SONST
        usw……..
```

Leichter wäre es, wenn wir statt SONST einfach die nächste Richtung angeben könnten.

```
ÜBERPRÜFE ausrichtung
FALL 'O'
        Drehe nach Osten
ENDE

FALL 'W'
        Drehe nach Westen
ENDE
…
```

Dies geht mit der **Mehrfachauswahl**:

void AusrichtungSetzen(char neueAusrichtung)

neueAusrichtung: 'O'	'S'	'W'	'N'	default

lampeOben.PositionSetzen(xPosition,yPosition)
lampeMitte.PositionSetzen(xPosition+1,yPosition);
lampeUnten.PositionSetzen(xPosition+2,yPosition);
	usw..	usw..	usw..	So reagiert das Programm wenn etwas anderes eingegeben wurde

Der Java Code orientiert sich hierbei wieder stark am Englischen.
ÜBERPRÜFE wird zu `switch` (engl. Schalter), FALL wird zu `case` (engl. Fall) und ENDE zu `break` (engl. Pause). Am Ende kann man einen `default` Fall für alle weiteren Angaben einfügen.
Am `break` springt das Programm immer raus aus der Mehrfachauswahl.
Damit ist der Ablauf:
Überprüfe die „`neueAusrichtung`“, springe zum dazugehörigen Fall, führe allen Code aus, der bis zum nächsten `break` kommt, springe vom `break` bis zum Ende der Mehrfachauswahl.

In der Mehrfachauswahl können als Fallunterscheidungen nur char (Buchstaben) und int (Zahl) benutzt werden!

Code:

```
void AusrichtungSetzen(char neueAusrichtung)
{
     ausrichtung = neueAusrichtung;

     switch(neueAusrichtung)
     {
          case 'O':
               lampeOben.PositionSetzen(xPosition,yPosition);
               lampeMitte.PositionSetzen(xPosition+1,yPosition);
               lampeUnten.PositionSetzen(xPosition+2,yPosition);
          break;
          case 'S':
               …
          break;
               ……
          default:
               //Was passiert wenn User sich vertippt oder so
     }

}
```

Aufgaben:

1) Ergänze deine AMPEL um das Attribut ausrichtung und weise ihm im Konstruktor einen sinnvollen Startwert zu.

2) Überlege dir, welche Positionen die LAMPEN in den vier Fälle 'N','S','W' und 'O' einnehmen müssen. Nutze eine Skizze in deinem Heft.

3) Setze die Methode AusrichtungSetzen(char neueAusrichtung) in der AMPEL um.

4) Gehe in dein Projekt mit HAUS und BAUM zurück. Könntest du die Mehrfachauswahl nutzen, um deine if-else Kaskaden leichter zu gestalten?

5) Erweitere das Verkehrsprojekt um die Klasse FUSSGAENGERAMPEL und lass auch diese ihre Ausrichtung wechseln.