Vererbung

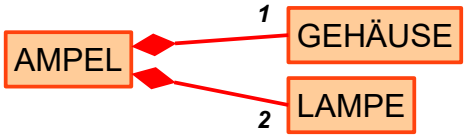
Was tust du, wenn du einmal eine fremde Klasse vorfindest, die viele Attribute und Methoden anbietet, die du für dein Projekt brauchst, aber leider nicht alles, was du brauchst?

- Du könntest ***in der fremden Klasse Veränderungen vornehmen***. Aber das ***ist eine ganz schlechte Idee!*** Sehr schnell hast du etwas verändert, so dass die Klasse sich gar nicht mehr übersetzen lässt und dann ist guter Rat teuer …

- Du solltest in so einem Fall ***besser eine eigene Klasse schreiben und damit die fremde vorgegebene Klasse erweitern***. Man sagt auch, deine Klasse wird von der vorgegebenen Klasse ***erben***. Damit stehen dir in deiner Klasse alle Konstruktoren und Methoden der vorgegebenen Klasse zur Verfügung ohne dass dieser Code in deiner Klasse erscheint. Du schreibst in deiner Klasse nur die Ergänzungen oder Veränderungen. Das ist viel übersichtlicher und „ungefährlicher"!

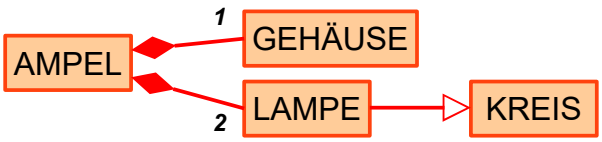
Erweitern

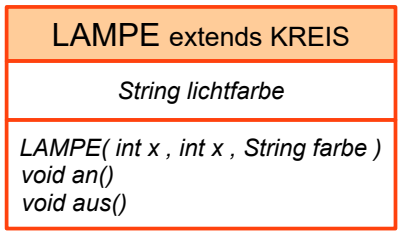
Wir wollen für eine ***Fußgänger-Ampel*** zwei runde Lampen programmieren die sich in einem rechteckigen Gehäuse befinden sollen. Unsere Ampel wird sich also aus drei Referenz-Attributen zusammensetzen: 1 Gehäuse und 2 Lampen.



- Das ***Gehäuse*** braucht eine bestimmte Breite, Höhe und Farbe. Außerdem muss es an einen bestimmten Ort gezeichnet werden können. All das kann die Klasse RECHTECK. Also wird das Gehäuse durch ein Referenz-Attribut der Klasse RECHTECK realisiert.

- Eine Lampe braucht eine bestimmte Farbe und einen bestimmten Radius. Außerdem muss sie an eine bestimmte Stelle gezeichnet werden können. Bis hierhin kommt die Klasse KREIS selbst in Frage. Aber wir wollen unsere Lampen ohne viel Code-Aufwand ganz einfach aus- und wieder anschalten. Die Klasse KREIS hat also vieles was wir brauchen aber leider nicht alles. Deshalb schreiben wir selbst eine Klasse LAMPE, die von der Klasse KREIS erben soll. In der Klasse LAMPE schreiben wir nur einen brauchbaren Konstruktor und die Methoden `an()` und `aus()` … fertig!





Code-Beispiel:

```
public class LAMPE extends KREIS                    ← erbt von Klasse Kreis
{
   private String lichtfarbe;                       ← neues Attribut

   public LAMPE( int x , int y , String farbe )
   {
      super(40);                                    ← geerbten Konstruktor KREIS(int radius) mit Wert 40 aufrufen
      super.setzeMittelpunkt( x , y );              ← geerbte Methoden aufrufen
      super.setzeFarbe( farbe );
      this.lichtfarbe = farbe;                      ← neues Attribut initialisieren
   }

   public void aus()
   {
      super.setzeFarbe( "schwarz" );                ← geerbte Methode aufrufen
   }

   public void an()
   {
      super.setzeFarbe( lichtfarbe );               ← geerbte Methode aufrufen
   }
}

public class AMPEL
{
   private RECHTECK gehaeuse;
   private LAMPE lampe_oben;                        ← Referenz-Attribute vom Datentyp
   private LAMPE lampe_unten;                         der neuen Klasse deklarieren
   private String zustand;

   public AMPEL()
   {
      this.gehaeuse = new RECHTECK( 100 , 200 );
      this.gehaeuse.setzeMittelpunkt( 50 , 100 );
      this.lampe_oben = new LAMPE( 50 , 50 , "rot");
      this.lampe_unten = new LAMPE( 50 , 150 , "gruen" );   ← wie gewohnt
      this.lampe_unten.aus();                                 weiter arbeiten
      this.zustand = "rot";
   }
   public void rot() {. . .}          public void gruen() {. . .}
}
```

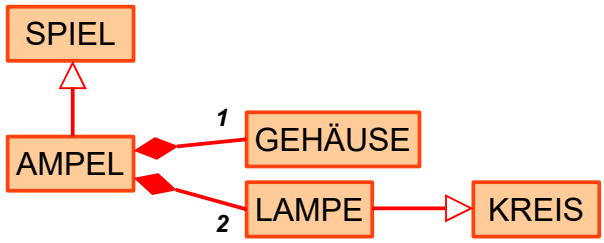
Erweitern und verändern

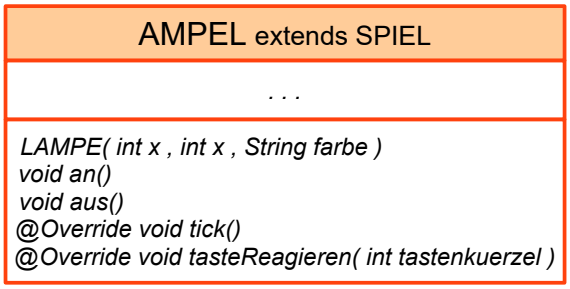
Stell dir vor, die Klasse AMPEL verfügt mittlerweile über die Methoden `rot()` und `gruen()`. Du möchtest, dass deine Ampel nun automatisch nach einer gewissen Zeit von rot nach grün und wieder auf rot … umschaltet. Außerdem möchtest du diese Automatik über die Tastatur aus- und einschalten können. Für automatische Ereignisse und Reaktion auf Tastatur-Eingaben gibt es die Klasse SPIEL.

Bitte arbeite nun zuerst das Dokument „Die Klasse SPIEL“ durch bevor du hier weiter machst!

- Sieh dir die Klassenkarte der Klasse SPIEL genau an.
  Du erbst nicht nur die Methoden
  `tick()` , `tickerStoppen()`, `tickerIntervallSetzen(int millisekunden)`, `tickerNeuStarten(int millisekunden)`, `tasteReagieren(int tastenkuerzel)`
  sondern auch eine Methdode `warte(int millisekunden)`.

- Lasse deine Klasse AMPEL nachträglich von der Klasse SPIEL erben und dir stehen all diese Methoden zur Verfügung.
  Rufe im Konstruktor von AMPEL vor allen anderen Befehlen als erstes den geerbten Konstruktor von SPIEL auf und erzeuge damit ein Fenster der Breite 100 Pixel und der Höhe 200 Pixel.
  Rufe als zweiten Befehl die geerbte Methode `tickerStoppen()` auf.





- Überschreibe die Methoden `tick()` und `tasteReagieren(int tastenkuerzel)` so dass sie deine Aufgaben erfüllen … fertig!

Code-Beispiel:

```
public class AMPEL extends SPIEL
{
   . . .
   public AMPEL()
   {
      super( 100 , 200 , false , false , false );
      super.tickerStoppen();
      this.gehaeuse = new . . .
      . . .
   }
   . . .
@Override
public void tick()
{
   if ( this.zustand == "rot" )
   {
      this.gruen();
   }
   else if ( this.zustand == "gruen" )
   {
      this.rot();
   }
}
   @Override
   public void tasteReagieren( int tastenkuerzel )
   {
      if ( tastenkuerzel == 0 )          // Taste A
      {
         this.tickerNeuStarten( 1000 );
      }
      else if ( tastenkuerzel == 18 )    // Taste S
      {
         this.tickerStoppen();
      }
   }
}
```