Projekt 5: Klassen können erben - Mit besonderen Fahrzeugen unterwegs

Bisher können in unserem Motoradmanager vier verschiedene Fahrzeuge behandelt werden: Alle haben ähnliche Eigenschaften, sie können fahren und betankt werden. Es gibt jedoch darunter Fahrzeuge, die beondere Eigenschaften haben:

Klasse Fahrzeug: (alle Motorräder) Roller, Harley,Honda,Yamaha: **Attribute:** tankGröße: Größe des Tankes spritProKm: Spritverbrauch pro Kilometer tankInhalt: aktueller Inhalt des Tankes kilometerStand: aktueller Kilometerstand des Motorrades	**Klasse AutobahnFahrzeug: (Motorräder, die auf der Autobahn fahren dürfen)** Harley, Honda, Yamaha **zusätzliche Attribute:** autobahn: Gibt an, ob das Motorrad gerade auf der Auto-bahn unterwegs ist oder nicht spritproKmAutobahnzusätzlich: zusätzlicher Sptitverbrauch auf der Autobahn	**Klasse BeiwagenFahrzeug: (Motorräder, die einen Beiwagen führen können)** Honda, Yamaha **zusätzliche Attribute:** beiwagen: Gibt an, ob das Motorrad gerade mit Beiwagen unterwegs ist spritproKmBeiwagenzusätzlich: zusätzlicher zusätzlicher Sptitverbrauch, wenn der beiwagen montiert ist

Unterklassen und Oberklassen

Während alle Motorräder der Familie die vier Attribute der Klasse Fahrzeuge besitzen, können nur die Motorräder Harley, Honda und Yamaha auf der Autobahn fahren. Sie gehören zur Klasse **AutobahnFahrzeug**. Diese Klasse ist eine **Tochterklasse** der Klasse Fahrzeug, dies bedeutet, dieses Fahrzeuge haben alle Attribute und Methoden aus der Klasse Fahrzeuge. Man sagt auch: die Tochterklasse AutobahnFahrzeug **erbt** die Eigenschaften der **Elternklasse** Fahrzeug. Die Motorräder der Tochterklasse Autobahnfahrzeug haben jedoch zwei zusätzliche Attribute, die benötigt werden, um das Fahren auf der Autobahn zu registrieren, bei dem der Spritverbrauch höher wird.

Die Klasse BeiwagenFahrzeug ist nochmals eine **Tochterklasse** der Klasse AutobahnFahrzeug. Honda und Yamaha können also auch Autobahn fahren, sie können jedoch zusätzlich einen Beiwagen montieren. Für diese Eigenschaft benötigt man wiederum zwei neue Attribute, die registrieren, wenn ein Beiwagen montiert ist, der den Spritverbrauch nochmals erhöht.

Wenn man in Java eine Tochterklasse anlegt, bedeutet das, dass die Tochterklasse bereits alles kann, was man in der **Elternklasse** programmiert hat. Man kann jedoch zusätzliche Methoden und Attribute festlegen, die den speziellen Eigenschaften aller Objekte entgegenkommen, die mit der Unterklasse verwaltet werden können. Und so sehen der Konstruktor der **Elternklasse** Fahrzeug und der **Tochterklasse** Autobahnfahrzeug aus:

```
public class Fahrzeug {
    protected double tankGröße;
    protected double spritProKm;
    protected double tankInhalt;
    protected int kilometerStand;
    protected boolean tatsächlichGetankt;
    protected boolean tatsächlichGefahren;

    public Fahrzeug(double pTankgröße, double pSpritProKm, int pKmStand, double pTankInhalt) {
        tankGröße = pTankgröße;
        spritProKm = pSpritProKm;
        kilometerStand = pKmStand;          tatsächlichGetankt = false;
        tankInhalt = pTankInhalt;           tatsächlichGefahren = false;
    }
```

Aus dem Access Modifier **private** wird **protected**: Variablenfelder, die als **protected** definiert sind, können von Tochterklassen gelesen werden

Die Elternklasse Fahrzeuge hat die Variablenfelder, die alle Fahrzeuge benötigen. Das Fahrzeug roller1 wird z. B. so angelegt:
roller1 = new Fahrzeug(20,0.03,100,10);

extends

Im Konstruktor der Klasse **Fahrzeug** werden die vier Zahlen zunächst den vier Parametern pTankGröße, pSpritProKm, pKmStand, pTankInhalt übergeben.

Anschließend werden die Werte aus den Parametern an die Variablenfelder tankGröße, spritProKm, kilometerStand und tankInhalt weitergereicht.

Das Wörtchen **extends** gibt an, dass **AutobahnFahrzeug** eine Tochter-klasse der Klasse **Fahrzeug** sein soll

```
public class AutobahnFahrzeug extends Fahrzeug{
    protected boolean autobahn;
    protected double spritProKmAutobahnZusätzlich;
    public AutobahnFahrzeug (double pTankGröße, double pSpritProKm, int pKmStand, double pTankInhalt,
                             boolean pAutobahn, double pSpritProKmAutobahnZusätzlich) {
        super (pTankGröße, pSpritProKm, pKmStand, pTankInhalt);
        autobahn = pAutobahn;
        spritProKmAutobahnZusätzlich = pSpritProKmAutobahnZusätzlich;
    }
```

Die Tochterklasse Autobahnfahrzeuge hat zwei zusätzliche Variablenfelder (Autobahn und pSpritProKmAutobahnZusätzlich). Das AutobahnFahrzeug harley wird z. B. so angelegt:
harley = new AutobahnFahrzeug(40,0.05,100,20,false,0.01);

super

Im Konstruktor werden die sechs Zahlen zunächst den sechs Parametern pTankGröße, pSpritProKm, pKmStand, pTankInhalt und **neu:** pAutobahn, pSpritProKmAutobahnZusätzlich übergeben.

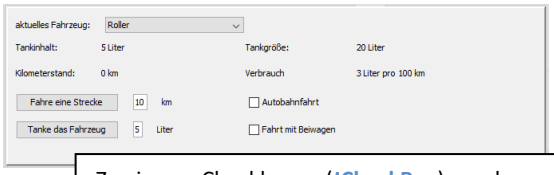
Die vier Parameter, die schon die Elternklasse Fahrzeug besitzt, werden durch den Befehl **super** an die Elternklasse Fahrzeug weitergereicht. Dadurch werden die Variabelnwerte innerhalb der Elternklasse Fahrzeuge an die vier Variablenfelder weitergegeben.

Anschließend werden die Werte aus den **beiden neuen Parametern** (pAutobahn und pSpritProKmAutobahnZusätzlich) an die Variablenfelder **autobahn** und **spritProKmAutobahnZusätzlich** weitergereicht. Das Schlüsselwort **this** steht für die aktuelle Klasse, d. h. die Klasse AutobahnFahrzeug, das Wort könnte man aber auch weglassen.

Aufgabe 1: Kopiere das Projekt **FahrzeugeZusatz** aus dem Schülerordner in deinen persönlichen Ordner. Öffne anschließend das Projekt über das Menü von Netbeans mit dem Befehl **File** / **Open Project**. Doppelklicke auf das Projekt FahrzeugeZusatz im Projekt-Fenster links, unter Source Package, my.FahrzeugeZusatz findest du die alten Klassen **Gui**, **Fahrzeug** und **Fahrer**, neu hinzugekommen sind die noch leeren Klassen **Autobahn-Fahrzeug** und **BeiwagenFahrzeug**. Definiere nun in der Klasse AutobahnFahrzeuge die Attribute und den Konstruktor so, wie auf S. 25 gezeigt.

Checkboxen

Aufgabe 2: a) Damit der Benutzer angeben kann, ob er Autobahn fahren möchte oder nicht, wird in der Klasse **Gui** im Designer ein Objekt der Klasse **JCheckBox** mit dem Namen **autobahnCheck** angelegt. Am Besten legst du in einem eine zweite **JCheckBox** mit Namen **beiwagenCheck** an, die später für die Benutzung des Beiwagens benötigt wird.



Zwei neue Checkboxen (**JCheckBox**) werden angelegt: **autobahnCheck** und **beiwagenCheck**

b) Es wird Zeit, die neuen Autobahnfahrzeuge anzulegen. Im Konstruktor von **Gui** werden harley, honda und yamaha als **AutobahnFahrzeug** deklariert. Die drei Autobahnfahrzeuge müssen aus der Deklaration von **Fahrzeuge** natürlich herausgenommen werden.

```
Fahrzeug roller1,zweirad;
AutobahnFahrzeug harley,honda,yamaha;
```

Die new-Befehle müssen geschrieben werden, verwende folgende Daten, die ersten vier Daten kannst du aus dem alten Programm übernehmen:

```
harley = new AutobahnFahrzeug(40,0.05,100,20,false,0.01);
```

	pTankgröße	pSpritproKm	pKmStand	pTankInhalt	pAutobahn	pSpritProKm-AutobahnZusätzlich
harley	40	0.05	100	20	false	0.01
honda	50	0.04	100	35	false	0.05
yamaha	50	0.05	100	50	false	0.03

Der Inhalt **false** bedeutet, dass die Motorräder standardmäßig zunächst nicht auf der Autobahn fahren.

c) Schreibe eine Event-Methode, die reagiert, wenn der Benutzer in der Checkbox für die Autobahnfahrt ein Kreuz setzt oder löscht. Wähle als Event dabei „Item“, „ItemStateChanged“ aus. Rechts siehst du die benötigte Event-Prozedur. Die vierte Zeile zeigt eine Besonderheit in Java: Das aktuell ausgewählte Motorrad hat den Namen zweiRad und ist ein Objekt der Klasse Fahrzeug. Wenn man wissen möchte, ob dieses zweiRad Autobahn fährt oder nicht, muss man an die Variable **autobahn** heran, die es aber nur in der Klasse AutobahnFahrzeug gibt:

```
private void autobahnCheckItemStateChanged(java.awt.ev
    AutobahnFahrzeug zweiradAutobahn;
    boolean b = autobahnCheck.isSelected();
    if (aktuellesFahrzeug > 0) {
        zweiradAutobahn = (AutobahnFahrzeug) zweirad;
        zweiradAutobahn.setAutobahn(b);
        verbrauchDarstellen();
    }
}
```

Ein Objekt der Klasse Fahrzeug wird zum Objekt der Klasse AutobahnFahrzeug

Die Methode isSelected() liefert als Ergebnis, ob in einer Checkbox das Kreuzchen gesetzt ist.

Typumwandlung (Cast)

Dazu kann man aus einem Objekt einer Elternklasse (Fahrzeug) ein Obejkt der Tochterklasse (Autobahn-Fahrzeug) machen. Dies geht mit dem Befehl:

zweiRadAutobahn = (AutobahnFahrzeug) zweirad

neues Objekt der Tochterklasse AutobahnFahrzeug

Unterklasse, in die das Objekt der Elternklasse umgewandelt werden soll

altes Objekt der Elternklasse Fahrzeuge

Ein Typenumwandlung (Cast) wird in die Klasse durchgeführt, die in Klammern angegeben ist

zweiradAutobahn wird dazu als lokale Variable als **AutobahnFahrzeug** in der ersten Zeile der Event-Prozedur deklariert. Diese Umwandlung funktioniert natürlich nur, wenn zweiRad tatsächlich auch ein Autobahnfahrzeug ist. Diese Bedingung wird im if-Befehl (**if aktuellesFahrzeug > 0**) realisiert: nur der Roller muss ausgeschlossen werden und dieser hat die Nummer 0.

d) Jetzt müssen einige neuen Methoden in der Klasse **AutobahnFahrzeug** geschrieben werden:

Die Methode **setAutobahn** soll die Variable autobahn setzen, die festhält, ob ein **AutobahnFahrzeug** gerade Autobahn fährt oder nicht. Zusätzlich benötigst du die get-Methode **public boolean getAutobahn()** , mit der man die Varibale **autobahn** auslesen kann.

```
public void setAutobahn(boolean pautobahn) {
  autobahn = pautobahn;
}
```

Die Methode **setAutobahn** aus der Klasse AutobahnFahrzeug

Und jetzt kommt das eigentliche Kernstück: Die Methode **fahre (int pStrecke)** muss so angepasst werden, dass der erhöhte Spritverbrauch berechnet wird, wenn das Fahrzeug auf der Autobahn fährt: Dazu wird zunächst der Spritverbrauch ohne Autobahnfahrt (spritverbrauch) berechnet. Fährt das Motorrad auf der Autobahn, muss der Spritverbrauch innerhalb eines if-Befehls erhöht werden. Wenn keine Autobahn benutzt wird, bleibt die Berechnung wie vorher. Zum Schluss müssen die Variablen **kilometerStand** und **tankInhalt** verändert werden. Ergänze die rechts dargestellte Methode entsprechend.

```
public void fahre(int pStrecke) {
   double spritVerbrauch = spritProKm * pStrecke;
   if (autobahn == true) {
     spritVerbrauch = spritVerbrauch + ...
```

Die Methode **fahre (pStrecke)** aus der Klasse AutobahnFahrzeug

Überschreiben und erben von Methoden

Zur Methode **fahre** muss man noch etwas grundsätzliches sagen: Diese Methode gibt es nun zweimal, einmal in der Klasse Fahrzeug und einmal in der Klasse Autobahnfahrzeug. Wird die Methode von einem Objekt, das als Fahrzeug deklariert ist, aufgerufen, wählt Java die Metode aus der Klasse Fahrzeuge. Wird die Methode von einem Objekt, das als AutobahnFahrzeug deklariert ist, aufgerufen, wählt Java automatisch die Metode aus der Klasse **AutobahnFahrzeuge**. Man sagt, dass die Methode fahre aus der Tochterklasse **AutobahnFahrzeug** die Methode aus der Klasse **Fahrzeuge** **überschreibt**.
Würde man die Methode fahre in der Tochterklasse **Autobahnfahrzeuge** nicht neu programmieren, würde einfach die Methode aus der Elternklasse **Fahrzeuge** ausgeführt, auch für AutobahnFahrzeuge. Man sagt dazu: die Klasse Autobahnfahrzeuge **erbt** eine Methode von der Elternklasse Fahrzeuge, d. h. man kann diese Methode auch in der Tochterklasse ausführen, obwohl man sie dort garnicht programmiert hat.

Als letztes muss noch die Methode **getSpritVerbrauch()** angepasst werden. Auch hier musst du jetzt unterscheiden, ob das Fahrzeug Autobahn fährt oder nicht. Teste dein Programm jetzt aus: Fahre mit der Yamaha einmal ohne und einmal mit Autobahn und untersuche, ob der Spritverbrauch richtig dargestellt wird.

```
public double getSpritproKm() {
   double wert = spritProKm;
   if (autobahn == true) {
     wert = wert + ...
   }
   return wert;
}
```

e) Einen Fehler müssen wir noch beseitigen: Wenn man ein anderes Fahrzeug auswählt, bleibt die Einstellung, ob Autobahnfahrt ausgewählt ist, einfach stehen. Das ist zum einen gefährlich, wenn der Roller gewählt wird und vorher Autobahnfahrt ausgewählt war. Außerdem sollte ein komfortables Programm immer den Zustand eines Fahrzeuges anzeigen, den man vorher für dieses Fahrzeug ausgewählt hatte. Das entsprechende Programmsegment, das rechts dargestellt wird, gehört ans Ende der Methode **fahrzeugAuswahl-ItemStateChanged** in der Klasse **Gui**. Der Befehl **autobahnCheck.setSelected(true)** setzt das Kreuz in der Checkbox, mit dem Wert **false** wird es entsprechend gelöscht.

```
verbrauchDarstellen();
if (aktuellesFahrzeug == 0) {autobahnCheck.setVisible(false);}
else {
  autobahnCheck.setVisible(true);
  zweiradAutobahn = (AutobahnFahrzeug) zweirad;
  autobahnCheck.setSelected(zweiradAutobahn.getAutobahn());
}
```

Text kann kopiert werden !

```
if (aktuellesFahrzeug == 0) {autobahnCheck.setVisible(false);}
else {
    autobahnCheck.setVisible(true);
    zweiradAutobahn = (AutobahnFahrzeug) zweirad;
    autobahnCheck.setSelected(zweiradAutobahn.getAutobahn());
  }
```

Ergänzung für die Methode **fahrzeugAuswahlItemStateChanged**:
Für den Roller wird die Auswahlmöglichkeit Autobahn unsichtbar gemacht, außerdem wird für Autobahnfahrzeuge der Zustand dargestellt, der zuletzt für dieses Fahrzeug gespeichert wurde.
Der Befehl **autobahnCheck.setSelected(true)** setzt das Kreuz in der Checkbox, mit dem Wert **false** wird es entsprechend gelöscht.

Ergänze am Ende des Konstruktors der Klasse **Gui** noch zwei Befehle, damit zu Beginn beim Roller die Checkboxen noch nicht dargestellt werden:

```
autobahnCheck.setVisible(false); beiwagenCheck.setVisible(false);
```

Aufgabe 3: Ergänze jetzt die Klasse **BeiwagenFahrzeug**.

	pAutobahn	pSpritProKm-AutobahnZusätzlich	pbeiwagen	pSpritproKmBeiwagenzusätzlich
honda	false	0.05	false	0.07
yamaha	false	0.02	false	0.04

Alle Schritte aus Aufgabe 1 und 2 werden erneut notwendig. **honda** und **yamaha** sind die beiden Objekte, die jetzt vom **AutobahnFahrzeug** zum **BeiwagenFahrzeug** werden. Im Aufgabenteil 2e) muss Java gesondert reagieren, wenn ein Beiwagenfahrzeug ausgewählt wird. Dies ist der Fall wenn die Variable **aktuellesFahrzeug** einen Wert größer als 1 einnimmt.

Zusatzaufgabe 1: Bisher wird mit Hilfe der Variablen **aktuellesFahrzeug** entschieden, ob ein Autobahnfahrzeug bzw, ein Beiwagenfahrzeug vorliegt. Dies kann man eleganter mit folgendem Befehl lösen:

If (zweirad instanceof Autobahnfahrzeug) {… }

instanceof liefert den boolschen Wert **true,** wenn **zweirad** ein **Autobahnfahrzeug** ist, sonst liefert es **false**. Bringe jetzt ein weiteres Motorrad ins Programm, das Autobahnfahrzeug, aber kein Beiwagenfahrzeug ist.

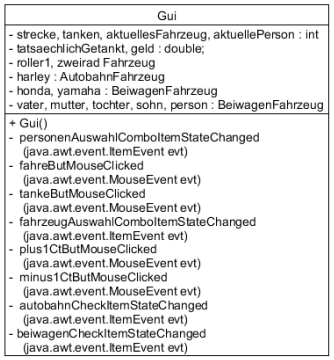
Zusatzaufgabe 2: Sobald ein AutobahnFahrzeug ausgewählt wird, soll in einem Label angegeben werden, wie viel Prozent der bisher gefahrenen Strecke auf der Autobahn stattfand. Überlege dazu, welche Attribute innerhalb der Klasse AutobahnFahrzeug neu eingeführt werden müssen. In diesem Fall ist es sinnvoll, die Variablenwerte im Konstruktor der Klasse Autobahnfahrzeug auf 0 zu setzen, eine Übergabe als Parameter ist nicht notwendig. Ein Sonderfall ist zu berücksichtigen: Ist das Fahrzeug bisher 0 km gefahren, muss man bei der Prozentrechnung aufpassen: duch 0 darf nicht dividiert werden. In diesem Fall soll einfach 0% angezeigt werden.

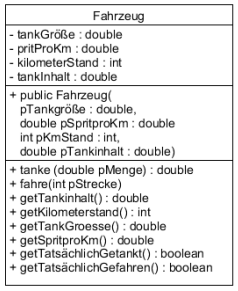
Ein Hinweis: In vielen Büchern findet man statt **Elternklasse** die Bezeichnung **Oberklasse**, statt **Tochterklasse** schreibt man **Unterklasse**.

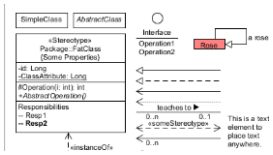
Aufgabe 4 - Wir schauen zurück:

Implemetationsdiagramme schreiben mit Umlet

Die nebenstehende Tabelle zeigt ein Implementationsdiagramm für alle Variablenfelder und Methoden, die in den beiden Klassen **Gui** und **Fahrzeug** verwendet werden. Starte das Programm Umlet vom Desktop und öffne das File „FahrzeugeZusatz UML“. Umlet dient zum schnellen erzeugen dieser Implemetationsdiagramme.
a) Schreibe das Implemetationsdiagramm für die fehlenden Klassen aus dem fertigen Projekt.



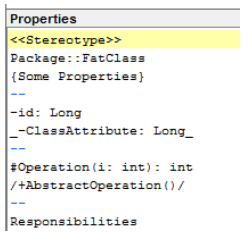




Stereotype ist das Objekt, mit dem Implementaions-diagramme gezeichnet werden

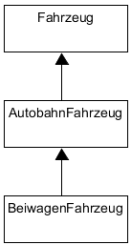
Klicke dazu im rechten Bereich auf das Objekt „Stereotype“ und ziehe es in den linken Bereich neben die beiden fertigen Diagramme.

Im unteren Properties-Fenster kannst du nun den Text gestalten. Die einzelnen Zeilen kannst du beliebig verändern. Der Eintrag „- -“ erzeugt die senkrecht verlaufenden Linien. Du sparst viel Arbeit, wenn du aus dem Java Programm die einzelnen Zeilen kopierst. **In den Tochterklassen werden nur die Attribute und Methoden aufgeführt, die neu geschrieben wurden.**



Im Properties-Fenster werden die einzelnen Zeilen eingegeben

b) Die Beziehungen zwischen den Klassen Fahrzeug, AutobahnFahrzeug und BeiwagenFahrzeug als Eltern- und Tochterklasse wird auch im Implementationsdiagrrmm deutlich gemacht: Ordne die drei Klassen entsprechend an und versehe das Diagramm mit den beiden Pfeilen.



Die Beziehungen zwischen den Klassen werden mit Pfeilen verdeutlicht. **Der Pfeil geht – anders als im Stammbaum - von der Tochterklasse zur Elternklasse.**