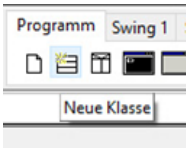
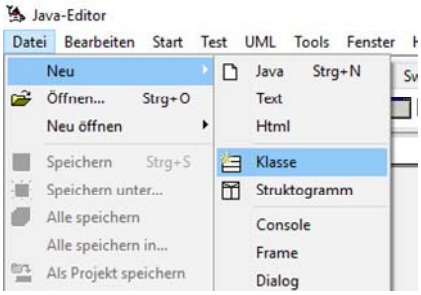
Java-OOP-Grundlagen

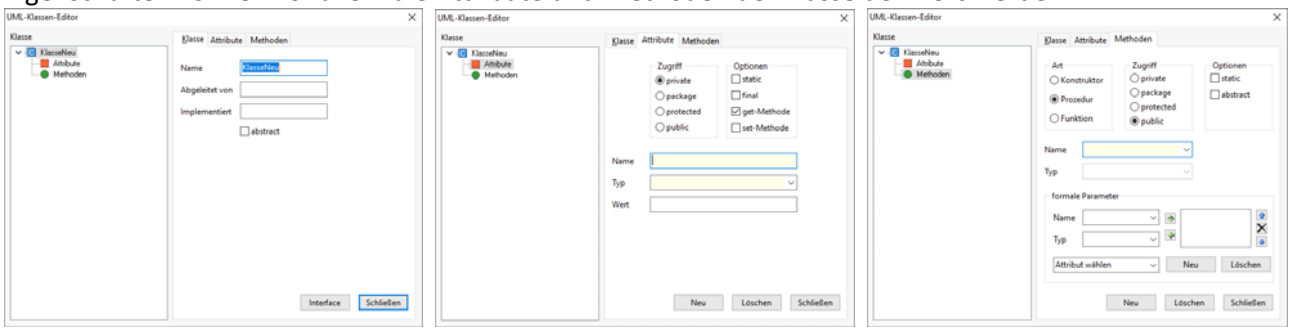
Aus dem PT-Unterricht kennen wir bereits das UML-Diagramm für eine Klasse, bzw. mehrere Klassen und deren Beziehungen zueinander. Da das UML-Diagramm Teil der Planungsphase ist, lässt es sich direkt nutzen, um den grundlegenden Programmcode zu erstellen.

Hierzu im Java-Editor eine neue Klasse erstellen, entweder über das Menü (Datei → Neu → Klasse) oder über die Symbolleiste.

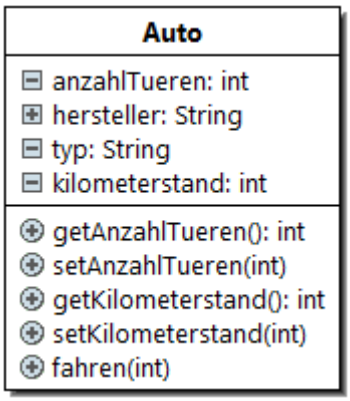




Es öffnet sich direkt ein Editor, um die Klasse zu definieren. Neben dem Namen und grundlegenden Eigenschaften können vor allem die Attribute und Methoden der Klasse definiert werden.

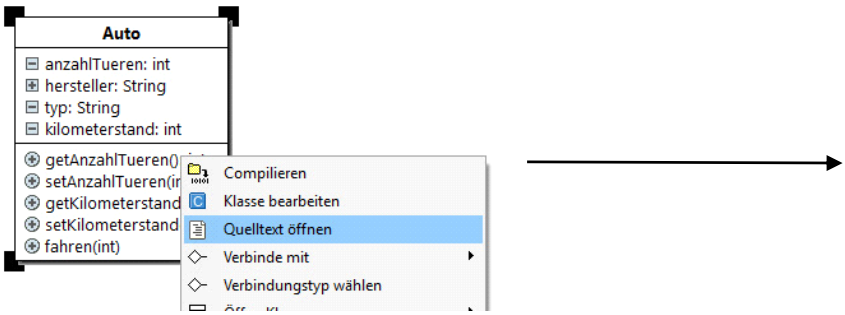


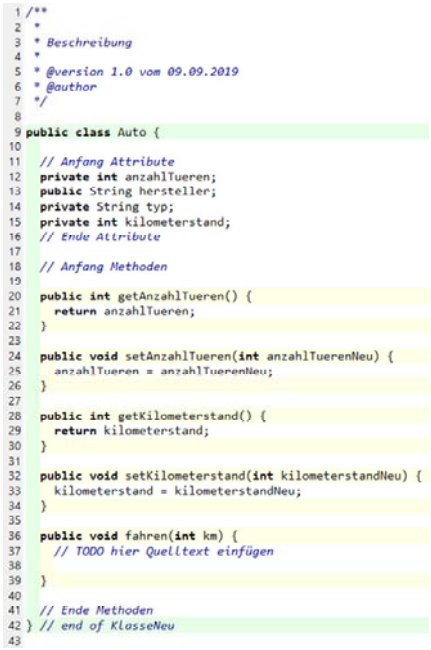
Es können alle wesentlichen Aspekte, die die Klasse betreffen, definiert und konfiguriert werden. Mach dich mit dem Editor vertraut, indem du folgende Klasse erstellst:



Die get… und set…-Methoden können automatisch beim Definieren eines Attributs mit erstellt werden. Sie ergeben – dem Prinzip der Kapselung folgende – auf jeden Fall Sinn.

Ist die Klasse fertig definiert, so kann der zugehörige Quelltext direkt erstellt werden, indem mit der rechten Maustaste auf die Klasse geklickt und „Quelltext öffnen“ ausgewählt wird.





```
/**
 *
 * Beschreibung
 *
 * @version 1.0 vom 09.09.2019
 * @author
 */

public class Auto {

  // Anfang Attribute
  private int anzahlTueren;
  public String hersteller;
  private String typ;
  private int kilometerstand;
  // Ende Attribute

  // Anfang Methoden

  public int getAnzahlTueren() {
    return anzahlTueren;
  }

  public void setAnzahlTueren(int anzahlTuerenNeu) {
    anzahlTueren = anzahlTuerenNeu;
  }

  public int getKilometerstand() {
    return kilometerstand;
  }

  public void setKilometerstand(int kilometerstandNeu) {
    kilometerstand = kilometerstandNeu;
  }

  public void fahren(int km) {
    // TODO hier Quelltext einfügen

  }

  // Ende Methoden
} // end of KlasseNeu
```

Die get…- und set….-Methoden sind bereits definiert, zu füllen wäre noch die Methode `fahren`, wobei wir hier vom JavaEditor explizit drauf hingewiesen werden (TODO…). Der JavaEditor nimmt dem Entwickler einiges an Arbeit und ev. Fehlerquellen ab. Wichtig ist, dass die Klasse vorher vollständig und korrekt definiert wurde, wobei natürlich Änderungen und Ergänzungen möglich sind.

Aufgabe

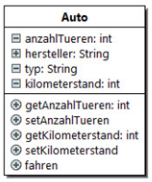
a) Vergleiche UML-Diagramm und Quellcode und schaue vor allem, wie die Symbole, Parameter und Rückgabewerte umgesetzt wurden. Mache kurze Notizen.
b) Worin unterscheidet sich diese Klasse zu den bisher erstellten „Klassen“?
c) Was wäre nun zu tun, um die Klasse `Auto` tatsächlich verwenden zu können?

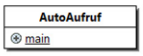
public static void main…

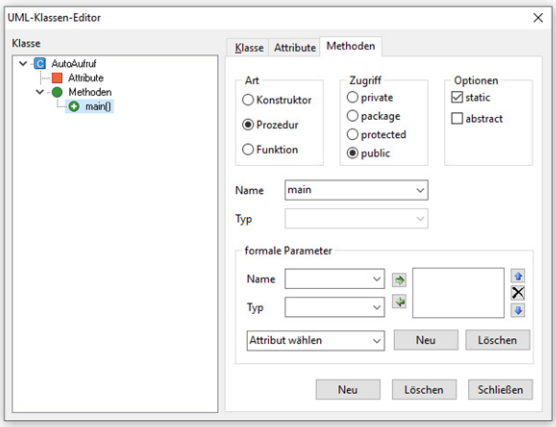
Tatsächlich zeigt nun die Klasse `Auto` das, was wir immer als „Bauanleitung“ bezeichnet haben. Es fehlt jedoch die Möglichkeit, das Auto zu „starten“. Der Modifikator `static` dient bei Methoden dazu, diese aufrufen und nutzen zu können, ohne dass tatsächlich eine Instanz, also ein Objekt, gebildet wurde. Wir haben im vergangenen Schuljahr damit bereits Erfahrungen gesammelt. Daher kann die Klasse `Auto` nicht direkt aufgerufen werden, sondern wir benötigen nun wieder eine Applikation, die diese Klasse nutzt.

Aufgabe

Erstelle eine neue Java-Klasse `AutoAufruf` in dem vorhandenen Klassendiagramm und definiere dort geeignete Methode `main`, mit der ein Aufruf möglich wird.
Es fehlt noch das `String[] args`, ergänze es selbständig.







Einbinden der Klasse `Auto`

Wir haben bereits häufig Klassen eingebunden. Wichtig ist nun, dass die Dateien Auto.java und AutoAufruf.java im gleichen Ordner liegen.
Durch Ergänzen einer geeigneten Zuweisung in der Main-Methode, wird die neue Klasse eingebunden und instanziiert.

```
1  public class AutoAufruf {
2
3    // Anfang Attribute
4    // Ende Attribute
5
6    public static void main(String[] args) {
7      Auto car = new Auto();
8
9    }
10
11   // Anfang Methoden
12   // Ende Methoden
13 } // end of AutoAufruf
14
```

Aufgabe

a) Das Auto `car` soll ein 5-türiger VW Golf mit einem Kilometerstand von 32.768km sein. Nimm die entsprechende Zuweisung vor und lass dir die Werte anschließend ausgeben.
b) Die Methode `fahren` in der Klasse Auto ist noch leer. Sie soll eine Kilometer-Angabe, die gefahren wurde, entgegennehmen und den Kilometerstand anpassen. Ergänze die Funktion dieser Methode.
c) Die Klasse `Auto` soll um die Attribute tankinhalt (60l) und durchschnittlicher Verbrauch je 100km (6,4l/100km) ergänzt werden. Nimm die Anpassung in der Klasse `Auto` vor und weise `car` die entsprechenden Werte zu.
d) Ändere die Methode `fahren` so ab, dass der Tankinhalt entsprechend der gefahrenen Strecke abnimmt. Sind für die aktuelle Füllung zu viele Kilometer gefahren worden, so soll eine entsprechende Meldung zurückgegeben werden, verbunden mit der Angabe, wie weit maximal hätte gefahren werden können.
e) Ergänze eine Methode `tanken`, die den Tankinhalt wieder füllt.
f) Ergänze das Attribut maximalerTankinhalt (70l) und passe die Methode `tanken` so an, dass nicht zu voll getankt werden kann. Ergänze eine entsprechende Meldung, sollte zu viel getankt werden.

$$t = k \cdot \frac{v}{100} \qquad K = \frac{100}{v} \cdot t$$

Aufgabe 1

a) Erstelle eine Klasse `Rechteck`, die die Attribute `seiteA` und `seiteB`, sowie die Methode `berechneFlaeche` enthält. Eine Applikation soll diese Klasse nutzen und die Seitenlängen, sowie die Fläche des Rechtecks mit Seitenlängen a = 5cm und b = 8cm berechnen und ausgeben.

b) Erstelle eine Klasse `Kreis`, die den `radius` als Attribut und die Methoden `berechneFlaeche` und `berechneUmfang` enthält. Erstelle auch hier die Ausgabe für den Kreis mit r = 5cm.

$u = 2\pi R$ $F = 2\pi R^2$

Aufgabe 2

Erstelle eine Klasse `Artikel` mit den Attributen `artikelNr`, `artikelBez`, `ekPreis` und `bestand`. Ergänze die Methode `artikelWert`, die aus ekPreis und bestand den aktuellen Lagerwert des Artikels berechnet.

Erstelle eine passende Applikation und instanziiere fünf Artikel, die rechts dargestellt werden. Gib den Lagerwert je Artikel und den gesamten Lagerwert aus.

artikelNr	artikelBez	ekPreis	bestand
815	Reifen	4,35	8
1435	Klingel	6,94	12
37	Rücklicht	12,52	24
294	Vorderlicht	17,28	18
536	Sattel	29,94	15