

Arbeitsblatt 07 (24.11.2015)

Astah – UML-Modellierungswerkzeug

In dieser Übung lernen Sie das UML-Modellierungswerkzeug Astah kennen:

- ✓ Lernen Sie das Erstellen von Diagrammen am Beispiel eines Klassendiagramms kennen.
- ✓ Lernen Sie das Exportieren der erstellten Diagramme.

Es gibt mehrere Möglichkeiten Astah zu starten:

1. Im Terminal mit folgendem Befehl: `astah&` (gilt für den CIP-Pool in INF350).
2. Im Terminal in den Astah-Ordner navigieren und folgenden Befehl eingeben:
`java -jar astah-community.jar`
3. Doppelklick auf ausführbare Anwendung `astah-community.jar` im Astah-Ordner.

Ihre Aufgabe ist es ein Klassendiagramm aus einem gegebenen Quellcode zu erstellen.

Aufgabe 1: Klassen


1. Neue Datei anlegen und Klassendiagramm erstellen

- Menu → File → New ...
- Menu → Diagramm → Class Diagram

2. Klassen identifizieren

- Identifizieren Sie im gegebenen Quellcode alle Klassen (siehe letzte Seite)

3. Klassen zeichnen

- Zeichnen Sie die von Ihnen identifizierten Klassen mit der Funktion „Class“ in das Diagramm ein: 

Aufgabe 2: Attribute und Operationen

Fügen Sie zu den von Ihnen gezeichneten Klassen jeweils deren Attribute und Operationen hinzu. Markieren Sie zunächst die Klasse, bei der Sie Attribute und Operationen hinzufügen wollen. In der linken Detailspalte erscheint die Übersicht über die Klasse. Im Tab „Attribute“ können Sie Namen, Typ und Sichtbarkeit festlegen. Im Tab „Operation“ können Sie Namen, Rückgabewert und Sichtbarkeit festlegen. Beide Einstellungen erreichen Sie auch über die Shortcuts „CTRL + F“ für Attribute und „CTRL + M“ für Operationen.

Attribute				
Name	Type	Type...	Visibility	Initial...
wheel...	int		private	
speed	int		private	

Operation			
Name	Return V...	Type Mo...	Visibility
getWheel...	int		public
getSpeed	int		public

Aufgabe 3: Beziehungen zwischen den Klassen

1. Identifizieren Sie die zwischen den Klassen vorliegenden Beziehungen. Beziehungstypen sind z.B. Assoziation, Aggregation, Komposition, Generalisierung etc.
2. Fügen Sie die identifizierten Beziehungen über das Menu hinzu.



Aufgabe 4: Multiplizitäten

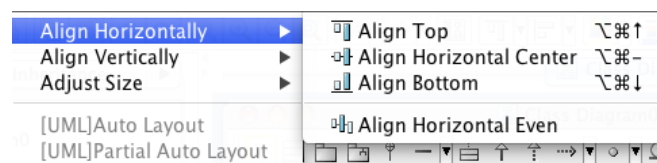
1. Identifizieren Sie die zwischen den Klassen vorliegenden Multiplizitäten.
2. Markieren Sie eine Beziehung.
3. Stellen Sie in der Detail-Spalte in den Tabs "Association End A" und "Association End B" die jeweilige Multiplizität ein.

Association End A	
Target	Car
Type Modifier	
Name	
Navigation	unspecified navig...
Aggregation	aggregate
Initial Value	
Visibility	private
Static	false
Leaf	false
Multiplicity	1
Derived	false

Association End B	
Target	Passenger
Type Modifier	
Name	
Navigation	navigable
Aggregation	none
Initial Value	
Visibility	private
Static	false
Leaf	false
Multiplicity	*
Derived	false

Aufgabe 5: Layout

Wenn Sie in Astah eine Klasse markieren und diese per Drag&Drop im Diagramm hin- und herziehen, werden Ihnen automatisch grüne Hilfslinien angeboten, die bei der Ausrichtung Ihrer Klassen helfen. Alternativ können Sie eine oder mehrere Klassen markieren und über das Menu → Alignment jeweils horizontal oder vertikal automatisch ausrichten lassen.



Aufgabe 6: Speichern und Exportieren

1. Speichern Sie das erstellte Diagramm über das Menu → File → Save As ...
2. Das von Ihnen erstellte Diagramm kann über das Menu → Tool in verschiedene Formate exportiert werden. Exportieren Sie Ihr Diagramm in das Format PNG.



Quellcode:

<pre> 1 package de.hd.uni.isw.vehicles; 2 3 import java.util.ArrayList; 4 5 abstract class Vehicle { 6 private int wheelCount; 7 private int speed; 8 9 public Vehicle(int pWheels, int pSpeed) { 10 this.wheelCount = pWheels; 11 this.speed = pSpeed; 12 } 13 14 public int getWheelCount() { 15 return wheelCount; 16 } 17 18 public int getSpeed() { 19 return speed; 20 } 21 } 22 23 class Passenger { 24 private String name; 25 26 public Passenger(String pName) { 27 this.name = pName; 28 } 29 30 public String getName() { 31 return name; 32 } 33 34 public void setName(String pName) { 35 this.name = pName; 36 } 37 } 38 39 class Engine { 40 private boolean isRunning = false; 41 42 public void startEngine() { 43 isRunning = true; 44 } 45 46 public void stopEngine() { 47 isRunning = false; 48 } 49 50 public boolean isEngineRunning() { 51 return isRunning; 52 } 53 } 54 55 56 57 </pre>	<pre> 58 class Car extends Vehicle { 59 private Engine engine; 60 private ArrayList<Passenger> passengers; 61 private int passengerCapacity; 62 63 public Car(int pWheels, int pSpeed, 64 int pCapacity) { 65 super(pWheels, pSpeed); 66 engine = new Engine(); 67 passengers = new 68 ArrayList<Passenger>(); 69 passengerCapacity = pCapacity; 70 } 71 public int getPassengerCapacity() { 72 return passengerCapacity; 73 } 74 75 public boolean addPassenger(Passenger p) { 76 if (passengers.size() < 77 passengerCapacity) { 78 passengers.add(p); 79 return true; 80 } else { 81 return false; 82 } 83 } 84 85 public boolean removePassenger(86 Passenger p) { 87 return passengers.remove(p); 88 } 89 90 public void startCar() { 91 engine.startEngine(); 92 } 93 94 public void stopCar() { 95 engine.stopEngine(); 96 } 97 } 98 99 class Bicycle extends Vehicle { 100 private Passenger p = null; 101 102 public Bicycle(int pWheels, int pSpeed) { 103 super(pWheels, pSpeed); 104 } 105 106 public void setPassenger(Passenger p) { 107 this.p = p; 108 } 109 110 public Passenger getPassenger() { 111 return p; 112 } 113 } 114 </pre>
<pre> 115 public class ExampleVehicles { 116 public static void main(String[] args) { 117 Car car = new Car(4, 250, 5); 118 Bicycle bike = new Bicycle(2, 40); 119 Passenger leonard = new Passenger("Leonard"); 120 Passenger sheldon = new Passenger("Sheldon"); 121 Passenger penny = new Passenger("Penny"); 122 car.addPassenger(leonard); 123 car.addPassenger(sheldon); 124 bike.setPassenger(penny); 125 } 126 } </pre>	