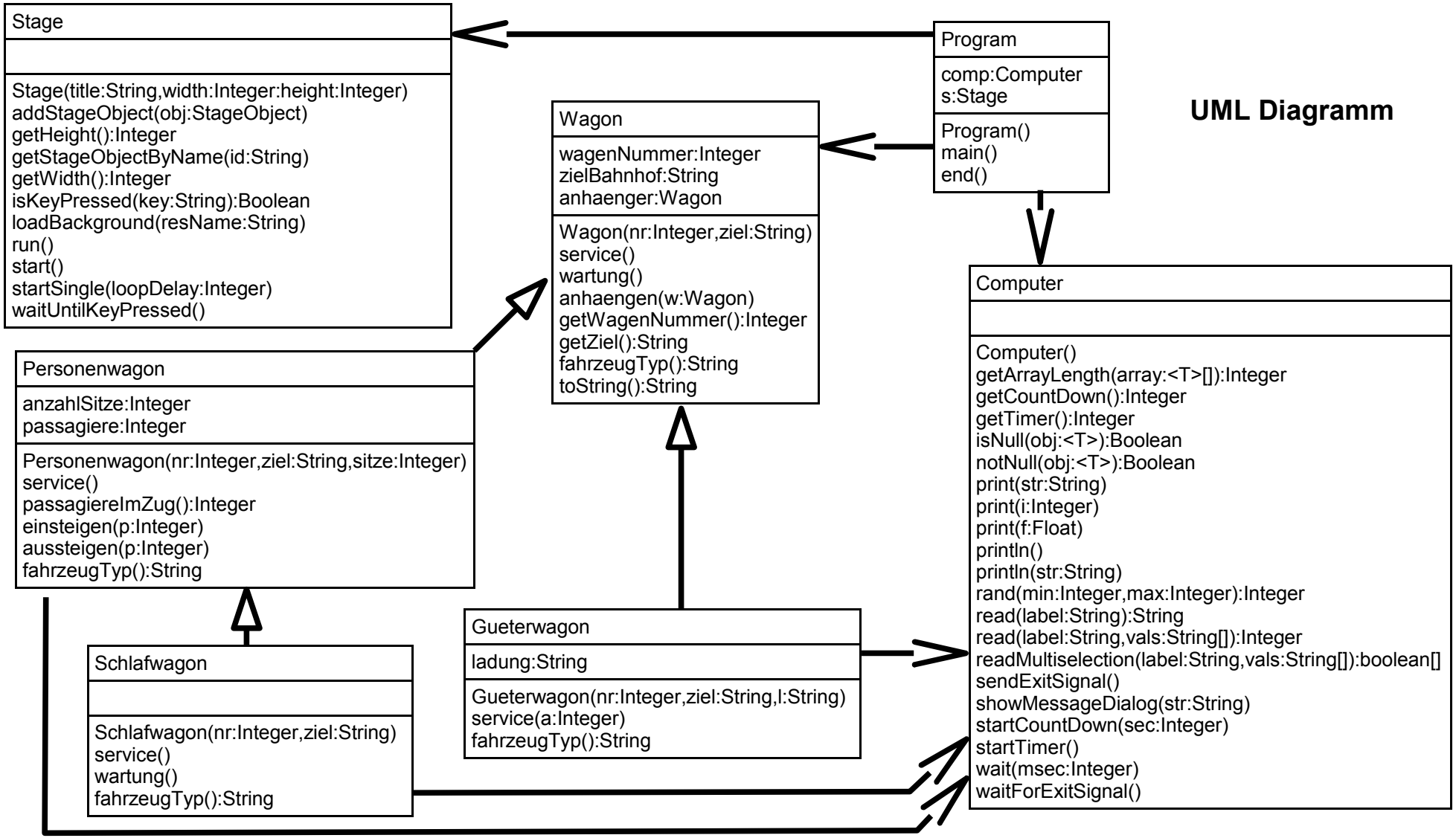
Vererbung in der objektorientieren Programmierung



© 2015 Henrik Horstmann

this

```
public class Wagon
{
   // ...

   public Wagon (Integer nr,String ziel)
   {
      String message;
      Computer comp;

      wagenNummer = nr;
      zielBahnhof = ziel;
      message = this.fahrzeugTyp();
      message.append(" erzeugt.");
      comp = new Computer();
      comp.println(message);
   }

   public void service ()
   {
      Computer comp;

      comp = new Computer();
      comp.println("Service Wagon");
   }

   // ...
}

public class Schlafwagon extends Personenwagon
{
   // ...

   public Schlafwagon (Integer nr,String ziel)
   {
      super(nr,ziel,0);
   }

   public void service ()
   {
      Computer comp;

      comp = new Computer();
      comp.println("Service Schlafwagon");
   }

   // ...
}
```

Aufgabe:

Erweitern Sie den Konstruktor von Wagon so, dass immer wenn ein *Wagon*, *Personenwagon*, *Schlafwagon* oder *Gueterwagon* mit `new` erzeugt wird, die jeweils passende *service* Methode aufgerufen wird.

 © 2010 Henrik Horstmann

this versus super

```
public class Schlafwagon extends Personenwagon
{
   // ...

   public Schlafwagon (Integer nr,String ziel)
   {
      super(nr,ziel,0);
   }

   // ...
}
```

Aufgabe

1. Beim erzeugen eines *Schlafwagons* soll die dazugehörige Methode *wartung* aufgerufen werden.

2. Beim erzeugen eines *Schlafwagons* soll die Methode *wartung* der Klasse *Wagon* aufgerufen werden.

3. Beim erzeugen eines *Schlafwagons* soll ein weiteres mal die zugehörige *service* Methode ausgeführt werden.

4. Beim erzeugen eines *Schlafwagons* soll die *service* Methode der Klasse *Personenwagon* ausgeführt werden.

5. Beim erzeugen eines *Schlafwagons* soll die *service* Methode der Klasse *Wagon* ausgeführt werden.

 © 2010 Henrik Horstmann

Überschreiben

```
public class Wagon
{
   // ...
   public void wartung ()
   {
      Computer comp;

      comp = new Computer();
      comp.println("Wartung Wagon");
   }
   // ...
}

public class Personenwagon extends Wagon
{
   // ...
   public Personenwagon (Integer nr,String ziel,Integer sitze)
   {
      super(nr,ziel);
      anzahlSitze = sitze;
      passagiere = new Integer();
      this.wartung();
   }
   // ...
}
```

Aufgabe

1. Welche Ausgabe wird durch die Codezeile `this.wartung();` im Konstruktor von *Personenwagon* am Bildschirm angezeigt, wenn folgender Code Ausgeführt wird.

```
Personenwagon sw;
sw = new Personenwagon (312,"Tübingen",32);
```

2. Welche Ausgabe wird am Bildschirm angezeigt, wenn im Konstruktor von Personenwagon `this.wartung()` durch `super.wartung()` ersetzt wird?

Die Klasse Personenwagon überschreibt die Methode *wartung* wie folgt:

```
   public void wartung ()
   {
      Computer comp;

      comp = new Computer();
      comp.println("Wartung Personenwagon");
   }
```

3. Welche Ausgabe produziert dann der Aufruf `this.wartung()` im Konstruktor von *Personenwagon*?

4. Welche Ausgabe produziert dann der Aufruf `super.wartung()` im Konstruktor von *Personenwagon*?

 © 2010 Henrik Horstmann

Zuweisungen

```
Wagon w1;
Personenwagon w2;
Schlafwagon w3;
Gueterwagon w4;
```

Aufgaben:

1. Ist die folgende Codezeile korrekt? Falls ja, was für ein Objekt befindet sich in der Variablen w2?
```
w2 = new Wagon(312,"Tübingen");
```

2. Ist die folgende Codezeile korrekt? Falls ja, was für ein Objekt befindet sich in der Variablen w1?
```
w1 = new Personenwagon(312,"Tübingen",32);
```

3. Sind die folgenden Codezeilen korrekt?
```
w1 = new Gueterwagon(312,"Tübingen","Bananen");
w1.service();
```

4. Sind die folgenden Codezeilen korrekt?
```
w1 = new Gueterwagon(312,"Tübingen","Bananen");
w1.service(4);
```

5. Sind die folgenden Codezeilen korrekt?
```
w4 = new Gueterwagon(312,"Tübingen","Bananen");
w4.service();
```

6. Sind die folgenden Codezeilen korrekt?
```
w4 = new Wagon(312,"Tübingen");
w4.service(4);
```

 © 2010 Henrik Horstmann