


ARBEITSAUFTRAG	
Fach: SEW2	
Thema: JAVA – Vererbung, static, UML Klassendiagramm	

Achtung: alle 4 Aufgaben sind getrennt voneinander abzugeben.

Aufgabe 1:

Gegeben sind folgende Klassen.

Fahrzeug als Superklasse, **Motorrad** und **PKW** als Subklassen von Fahrzeug.

Die Klasse **Fahrzeug**:

.....

```
-typ:String
-anzahlRaeder:int
+Fahrzeug(anzahlRaeder:int)
+Fahrzeug(typ:String, anzahlRaeder:int)
+getTyp():String
+getAnzahlRaeder():int
```

Die Klasse **Motorrad**:

```
+Motorrad()
```

Die Klasse **PKW**:

```
-anzahlTueren:int
+PKW(anzahlRaeder:int, anzahlTueren:int)
+getAnzahlTueren() :int
```

Implementieren Sie die Klassen so, dass das folgende Programm korrekt abläuft:

Schreiben Sie dafür eine ausführbare Klasse die den Namen FahrzeugTest hat.


```
fahrzeug fz = new Fahrzeug(10);
System.out.println("Das Fahrzeug ist ein "+fz.getTyp()+" mit "+ fz.getAnzahlRaeder()+" Raedern.");
PKW pkw = new PKW(4,6);
System.out.println("Das Fahrzeug ist ein "+pkw.getTyp()+" mit "+ pkw.getAnzahlRaeder() +" Raedern
und "+pkw.getAnzahlTueren()+" Tueren.");
Motorrad mo = new Motorrad();
System.out.println("Das Fahrzeug ist ein "+mo.getTyp()+" mit "+ mo.getAnzahlRaeder() +" Raedern.");
```

Das Programm soll auf dem Bildschirm diese Ausgabe erzeugen:

Das Fahrzeug ist ein allgemeines Fahrzeug mit 10 Rädern.

Das Fahrzeug ist ein PKW mit 4 Rädern und 6 Türen.

Das Fahrzeug ist ein Motorrad mit 2 Rädern.

ARBEITSAUFTRAG	
Fach: SEW2	
Thema: JAVA – Vererbung, static, UML Klassendiagramm	

Aufgabe 2:

Fügen Sie der Klasse Fahrzeug die Methode

String gibAlsString() hinzu. Sie liefert eine String Repräsentation des aktuellen Objekts.

Diese Methode wird in der Klasse PKW überschreiben, aber so, dass sie das Ergebnis der Methode der Oberklasse nutzt und entsprechend ergänzt. Die Klasse Motorrad überschreibt die Methode nicht, sie kommt mit der Implementierung in der Oberklasse aus.

Ändern Sie das Programm folgendermaßen ab:

```
Fahrzeug[] fahrzeuge = new Fahrzeug[] {new Fahrzeug(10),
new PKW(4,6),
new Motorrad()};
for (int i=0;i<fahrzeuge.length;i++)
{
    System.out.println("Das Fahrzeug ist ein "+fahrzeuge[i].gibAlsString());
}
```

Aufgabe 3:

Jede Klasse erbt von der Klasse Object die Methode


String toString()

Sie ist dafür gedacht, dem Aufrufer eine String Repräsentation des betreffenden Objekts zu liefern. Die Methode toString() wird dann implizit aufgerufen, wenn ein Objekt im Kontext von Strings genutzt wird. Die Standardimplementierung von toString() ist allerdings nicht zufriedenstellend, so dass in jeder Klasse, die eine String Repräsentation für ihre Objekte benötigt, die Methode toString() überschreiben muss.

Führen Sie in den Klassen, die eine Methode gibAlsString() haben, die Methode toString() ein, indem Sie die Methode gibAlsString() umbenennen in toString().

Bei der Methode toString() kommt einerseits der Polymorphismus zu tragen, auf der anderen Seite aber auch die Sonderrolle dieser Methode im Zusammenhang mit der String-Aufbereitung.

```
Fahrzeug[] fahrzeuge = new Fahrzeug[] {new Fahrzeug(10),
new PKW(4,6),
new Motorrad()};
System.out.println("(1)");
for (int i=0;i<fahrzeuge.length;i++)
{
    String s1 = "Das Fahrzeug ist ein ";
    String s2 = fahrzeuge[i].toString();
    System.out.println(s1 + s2);
}
System.out.println("(2)");
for (int i=0;i<fahrzeuge.length;i++)
{
    String s = "Das Fahrzeug ist ein "+fahrzeuge[i].gibAlsString();
    System.out.println(s);
}
```

ARBEITSAUFTRAG	
Fach: SEW2	
Thema: JAVA – Vererbung, static, UML Klassendiagramm	

Aufgabe 4:

Bislang haben wir den Polymorphismus auf der Ebene Fahrzeug ausprobiert. Da alle Klassen als oberste Superklasse die Klasse Object haben, spielt der Polymorphismus auf der Ebene Object eine große Rolle.

Ändern Sie das Programm folgendermaßen ab:

```
Object[] fahrzeuge = new Object[] {new Fahrzeug(10),
new PKW(4,6),
new Motorrad()};
for (int i=0;i<fahrzeuge.length;i++)
{
String s = "Das Fahrzeug ist ein "+fahrzeuge[i];
System.out.println(s);
}
```