```
  1 //===========================================================================
  2 //        * Letsch Informatik *       www.LetsInfo.ch       CH-8636 Wald
  3 //          Beratung, Ausbildung und Realisation in Software-Engineering
  4 //===========================================================================
  5 // Project   : Master of Advanced Studies in Software-Engineering 2010
  6 // Modul     : OO-Einführung mit UML und Java
  7 //             Teil: UML -> Java
  8 // Title     : Übung 5: Klassen-Hierarchie 'Fahrzeuge'
  9 // Author    : Thomas Letsch
 10 // Tab-Width : 2
 11 /*///=========================================================================
 12
 13 * Description:
 14 Klassen-Hierarchie mit "Fahrzeug" als Basis-Klasse.
 15 Klasse "Strassenfahrzeug" und "Wasserfahrzeug" davon abgeleitet.
 16 Strassenfahrzeug zusätzlich mit zurückgelegten Kilometer.
 17 Wasserfahrzeug zusätzlich mit zurückgelegten Seemeilen.
 18
 19 * History    :
 20 02.05.1999: Initial-Version
 21 07.05.2000: WIN32
 22 10.05.2001: WIN32-spezielle Anpassungen gelöscht.
 23 14.05.2001: Fahrzeug::fahren() mit korrektem Trace.
 24 18.04.2002: Von C++ auf Java portiert.
 25 07.05.2003: Neue File-Aufteilung betref Eclipse.
 26 06.05.2004: Kosmetik.
 27 21.08.2005: Fahrzeug::printInfos(); public -> protected.
 28 07.05.2007: Referenz-Typ nur noch 'Fahrzeug'.
 29 CVS: $Revision: 1.10 $  $Date: 2008/05/05 12:30:56 $
 30 /*///=========================================================================
 31 //       1         2         3         4         5         6         7         8
 32 //345678901234567890123456789012345678901234567890123456789012345678901234567890
 33 //===========================================================================
 34
 35
 36 public class FahrzeugeTest {
 37
 38   public static void main(String[] args) {
 39
 40     System.out.println("\n=== fz = new Fahrzeug(0, 0): ================");
 41     Fahrzeug fz = new Fahrzeug(0, 0);
 42     System.out.println("=== fz.fahren(3, 4): ========================");
 43     fz.fahren(3, 4);
 44     fz.printlnInfos("fz");
 45     System.out.println("\n=== fz.fahren(-1, 1): ========================");
 46     fz.fahren(-1, 1);
 47     fz.printlnInfos("fz");
 48
 49     System.out.println("\n=== sfz = new Strassenfahrzeug(0, 0, 0): ===");
 50     Fahrzeug sfz = new Strassenfahrzeug(0, 0, 0);
 51     System.out.println("=== sfz.fahren(3, 4): ======================");
 52     sfz.fahren(3, 4);
 53     sfz.printlnInfos("sfz");
 54     System.out.println("\n=== sfz.fahren(-1, 1): ======================");
 55     sfz.fahren(-1, 1);
 56     sfz.printlnInfos("sfz");
 57
 58     System.out.println("\n=== wfz = new Wasserfahrzeug(0, 0, 0): ===");
 59     Fahrzeug wfz = new Wasserfahrzeug(0, 0, 0);
 60     System.out.println("=== wfz.fahren(3, 4): ======================");
 61     wfz.fahren(3, 4);
 62     wfz.printlnInfos("wfz");
 63
 64   }
 65
 66 }
 67
 68
 69 /* Session-Log:
 70
 71 === fz = new Fahrzeug(0, 0): ================
 72 === fz.fahren(3, 4): ========================
 73 Fahrzeug.fahren(): 5.0
 74 fz: Koord = 3.0/4.0
 75
 76 === fz.fahren(-1, 1): ========================
 77 Fahrzeug.fahren(): 5.0
 78 fz: Koord = -1.0/1.0
 79
 80 === sfz = new Strassenfahrzeug(0, 0, 0): ===
 81 === sfz.fahren(3, 4): ======================
 82 Fahrzeug.fahren(): 5.0
 83 Strassenfahrzeug.fahren(): total KM: 5.0
 84 sfz: Koord = 3.0/4.0    KM = 5.0
 85
 86 === sfz.fahren(-1, 1): ======================
 87 Fahrzeug.fahren(): 5.0
 88 Strassenfahrzeug.fahren(): total KM: 10.0
 89 sfz: Koord = -1.0/1.0   KM = 10.0
 90
 91 === wfz = new Wasserfahrzeug(0, 0, 0): ===
 92 === wfz.fahren(3, 4): ======================
 93 Fahrzeug.fahren(): 5.0
 94 Wasserfahrzeug.fahren(): total SM: 2.6997840172786174
 95 wfz: Koord = 3.0/4.0    SM = 2.6997840172786174
 96
 97 */
```

```
 1 // MAS-SE-10: Übung 5: Klassen-Hierarchie 'Fahrzeuge'
 2
 3 class Fahrzeug {
 4
 5   private double mPosX;
 6   private double mPosY;
 7
 8
 9   public Fahrzeug(double pXinit, double pYinit) {
10     mPosX = pXinit;
11     mPosY = pYinit;
12   }
13
14
15   public double fahren(double pX, double pY) {
16     System.out.print("Fahrzeug.fahren(): ");
17     double xDiff = mPosX - pX;
18     double yDiff = mPosY - pY;
19     double km = Math.sqrt(xDiff * xDiff + yDiff * yDiff);
20     mPosX = pX;
21     mPosY = pY;
22     System.out.println(km);
23     return km;
24   }
25
26
27   protected void printInfos(String pPrefix) {
28     System.out.print(pPrefix + ": Koord = " + mPosX + "/" + mPosY);
29   }
30
31
32   public void printlnInfos(String pPrefix) {
33     printInfos(pPrefix);
34     System.out.print("\n");
35   }
36
37
38   public double getPosX() {
39     return mPosX;
40   }
41
42
43   public double getPosY() {
44     return mPosY;
45   }
46
47 }
```



```
 1 // MAS-SE-10: Übung 5: Klassen-Hierarchie 'Fahrzeuge'
 2
 3 class Strassenfahrzeug extends Fahrzeug {
 4
 5   private double mKM; // gefahrene Kilometer
 6
 7
 8   public Strassenfahrzeug(double pXinit, double pYinit, double pKMinit) {
 9     super(pXinit, pYinit); // Aufruf des Konstruktors der Basis-Klasse
10     mKM = pKMinit;
11   }
12
13
14   public double fahren(double pX, double pY) {
15     double km = super.fahren(pX, pY);
16     mKM = mKM + km;
17     System.out.println("Strassenfahrzeug.fahren(): total KM: " + mKM);
18     return km;
19   }
20
21
22   public void printlnInfos(String pPrefix) {
23     printInfos(pPrefix);
24     System.out.println("\tKM = " + mKM);
25   }
26
27 }
```

```
 1 // MAS-SE-10: Übung 5: Klassen-Hierarchie 'Fahrzeuge'
 2
 3 class Wasserfahrzeug extends Fahrzeug {
 4
 5   private double mSM; // gefahrene Seemeilen
 6
 7
 8   public Wasserfahrzeug(double pXinit, double pYinit, double pSMinit) {
 9     super(pXinit, pYinit); // Aufruf des Konstruktors der Basis-Klasse
10     mSM = pSMinit;
11   }
12
13
14   public double fahren(double pX, double pY) {
15     double km = super.fahren(pX, pY);
16     mSM = mSM + (km / 1.852);
17     System.out.println("Wasserfahrzeug.fahren(): total SM: " + mSM);
18     return km;
19   }
20
21
22   public void printlnInfos(String pPrefix) {
23     printInfos(pPrefix);
24     System.out.println("\tSM = " + mSM);
25   }
26
27 }
```