TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)

`http://db.in.tum.de/teaching/ss22/ei2/`

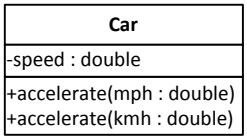
Blatt Nr. 3

Dieses Blatt wird am Montag, den 16. Mai 2022 besprochen.

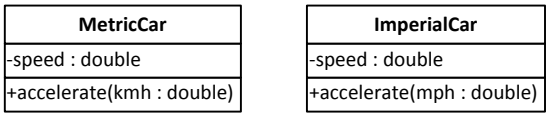
Aufgabe 1: Overloading

Welche der folgenden Methoden-Überladungen sind erlaubt und welche nicht? Überprüfen Sie Ihre Antworten indem Sie die Beispiele in Java programmieren. Eine unterstrichener Methodenname bedeutet dass es sich um eine Klassenmethode handelt.

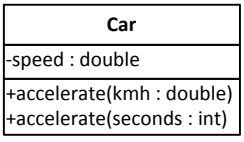
a)



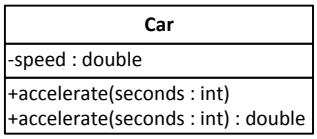
b)



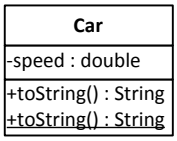
c)



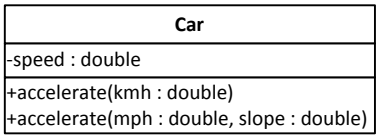
d)



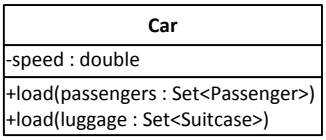
e)



f)



g)



Aufgabe 2: Dynamisches Binden - Teil 1

Angenommen, es gibt die folgenden Klassenbeziehungen:

```
abstract class A {}
class B extends A {}
class C extends B {}
class D extends B {}
class E extends D {}
```

		dynamischer Typ				
		A	B	C	D	E
statischer Typ	A					
	B					
	C					
	D					
	E					

Geben Sie in der Tabelle an, welche Objekte welchen dynamischen Typs einer Variablen mit einem statischen Typen zugeordent werden können.

Aufgabe 3: Dynamisches Binden - Teil 2

Überlegen Sie sich welche Methoden aufgerufen werden, wenn man die Klasse `DynamicDispatch` ausführt. Überprüfen Sie anschließend ihre Vermutung, indem Sie das Programm tatsächlich ausführen.

```
class DynamicDispatch {
  public static void main(String[] args) {
    A a = new A();
    B b = new B();
    C c = new C();
    D d = new D();

    A[] array = {a, b, c, d};
    for (A element : array) {
      System.out.println("x():");
      element.x();
      System.out.println("\ny():");
      element.y();
      System.out.println("\nz():");
      element.z();
      System.out.println("\n=====\n");
    }
  }
}

class A {
  public void x() {
```

```
23      System.out.println("-> A x()");
24      z();
25    }
26
27    public void y() {
28      System.out.println("-> A y()");
29      this.z();
30    }
31
32    public void z() {
33      System.out.println("-> A z()");
34    }
35  }
36
37  class B extends A {
38    public void y() {
39      System.out.println("-> B y()");
40      x();
41    }
42
43    public void z() {
44      System.out.println("-> B z()");
45    }
46  }
47
48  class C extends B {
49    public void x() {
50      System.out.println("-> C x()");
51      z();
52    }
53  }
54
55  class D extends A {
56    public void x() {
57      System.out.println("-> D x()");
58      super.x();
59    }
60
61    public void z() {
62      System.out.println("-> D z()");
63    }
64  }
```