Technische Universität
Kaiserslautern

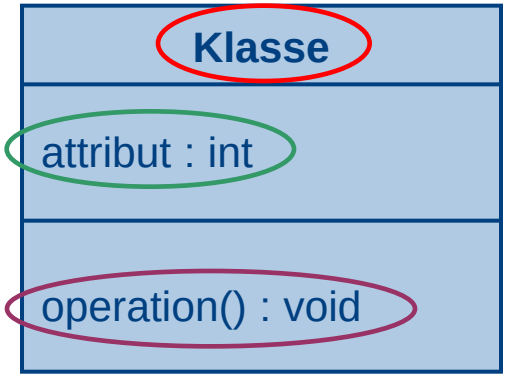
o1o1seda o1o1oo

software engineering dependability

Software Entwicklung 2

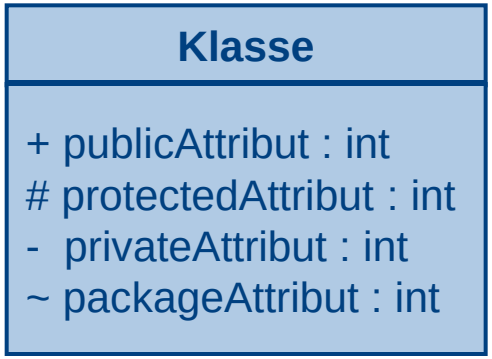
UML… umgesetzt

Klasse (vereinfacht)



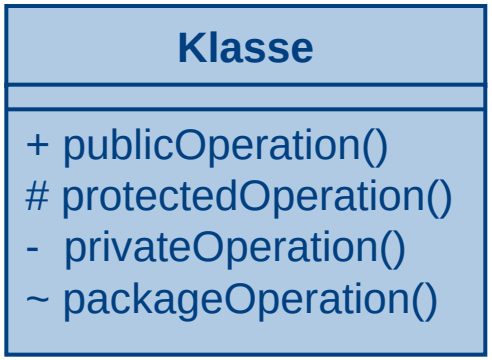

```
public class Klasse {
  int attribut;
  void operation() {…}
}
```

Sichtbarkeit der Attribute



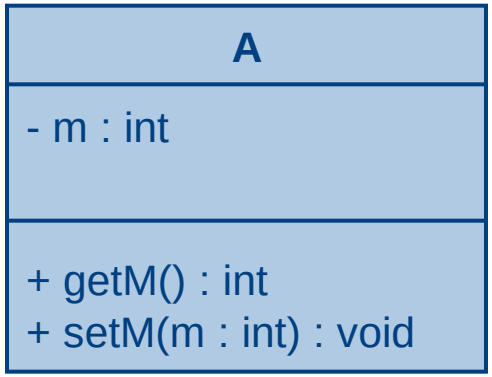

```
public class Klasse {
   public int publicAttribut;
   protected int publicAttribut;
   private int privateAttribut;
   int packageAttribut;
}
```

Sichtbarkeit der Operationen

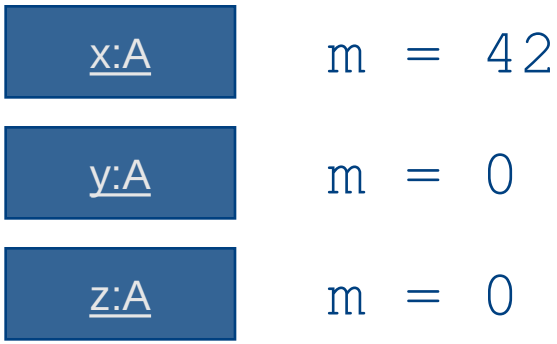



```
public class Klasse {
   public void publicOperation(){}
   protected void publicOperation() {}
   private void privateOperation() {}
   void packageOperation() {}
}
```

Beispiel

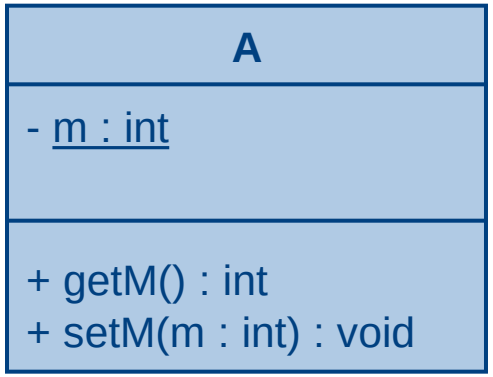



```
public class A {
  private int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```



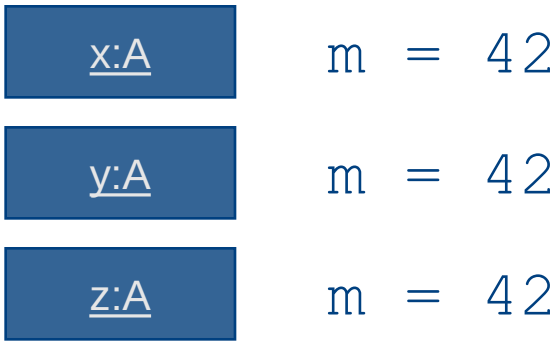

```
A x = new A();
A y = new A();
A z = new A();
x.setM(42);
```

Statische Attribute



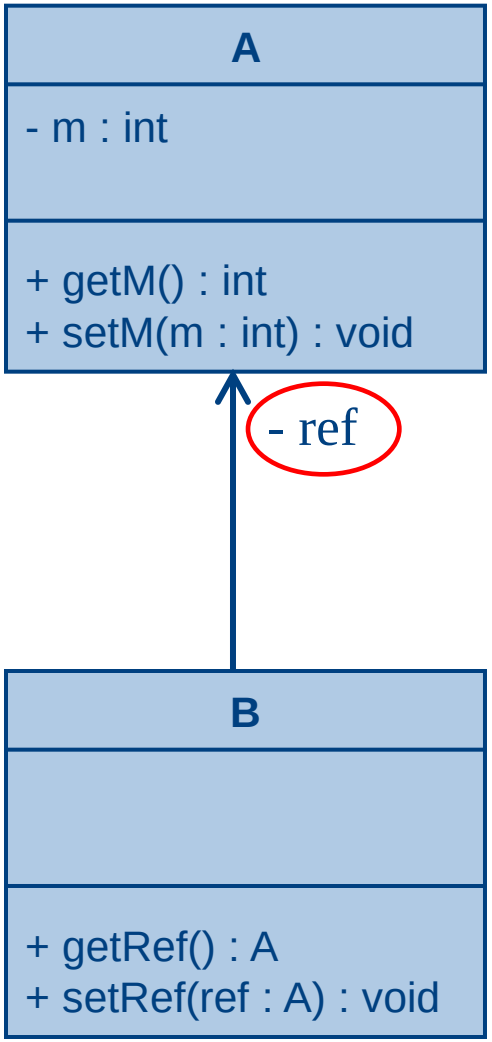

```
public class A {
  private static int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

entspricht: A.m = m;




```
A x = new A();
A y = new A();
A z = new A();
x.setM(42);
```

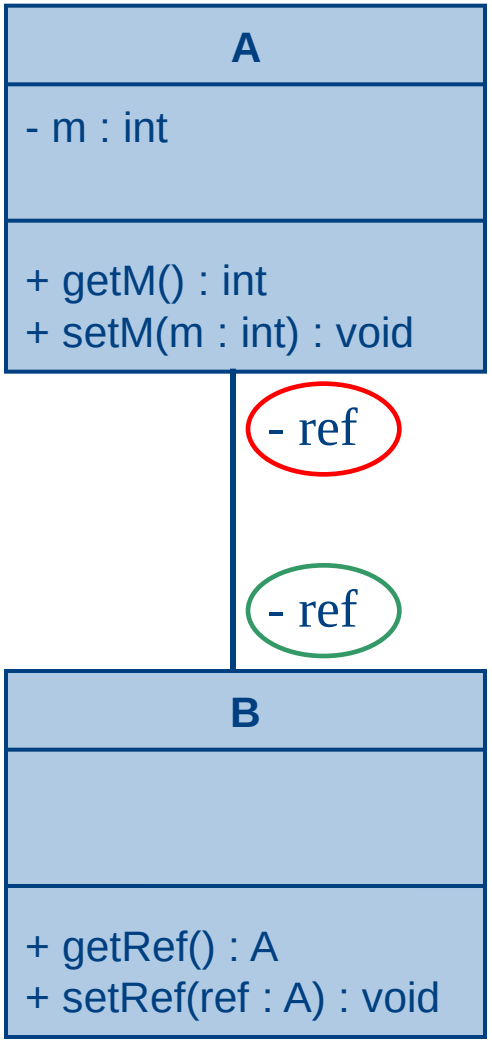
Assoziationen: gerichtet




```
public class A {
  private int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

```
public class B {
  private A ref;
  public A getRef() { return ref; }
  public void setRef(A ref) { this.ref = ref; }
}
```

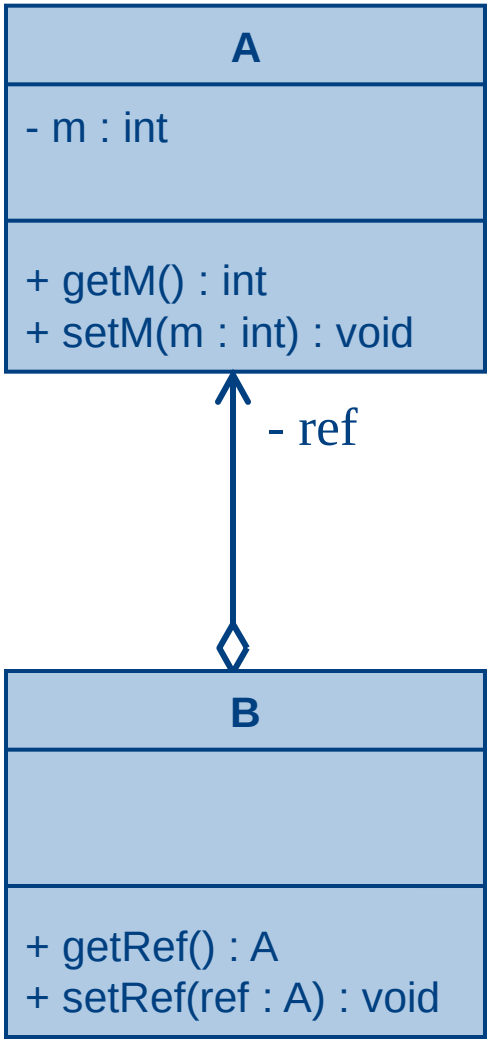
Assoziationen: ungerichtet




```
public class A {
  private int m;
  private B ref;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

```
public class B {
  private A ref;
  public A getRef() { return ref; }
  public void setRef(A ref) { this.ref = ref; }
}
```

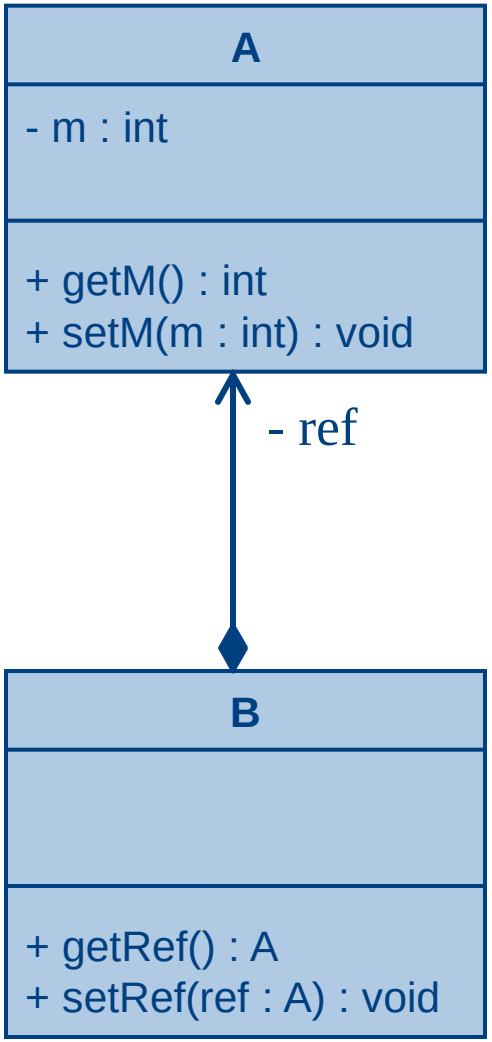
Assoziationen: Aggregation (Java)




```
public class A {
  private int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

```
public class B {
  private A ref;
  public A getRef() { return ref; }
  public void setRef(A ref) { this.ref = ref; }
}
```

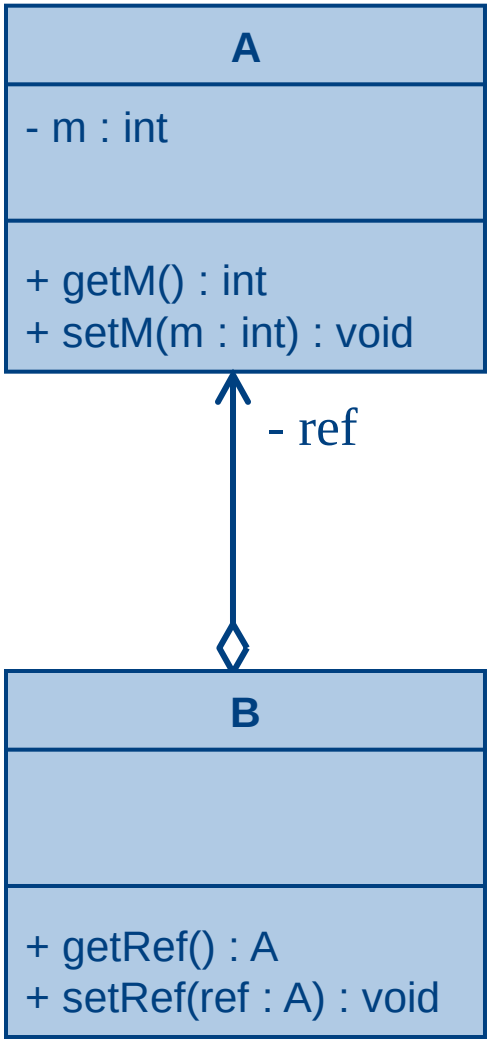
Assoziationen: Komposition (Java)




```
public class A {
  private int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

```
public class B {
  private A ref;
  public A getRef() { return ref; }
  public void setRef(A ref) { this.ref = ref; }
}
```

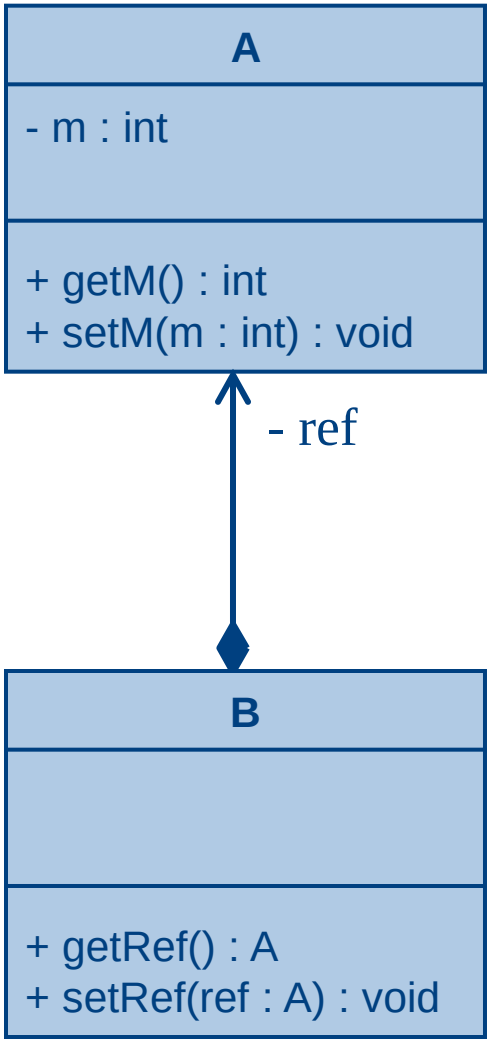
Assoziationen: Aggregation (C++)




```
class A {
  private:
    int m;
  public:
    int getM() { return m; }
    void setM(int m) { this->m = m; }
};
```

```
class B {
  private:
    A * ref;
  public:
    B() {}
    ~B() {}
    A getRef() { return ref; }
    void setRef(A ref) { this->ref = ref; }
};
```

Assoziationen: Komposition (C++)




```
class A {
  private:
    int m;
  public:
    int getM() { return m; }
    void setM(int m) { this->m = m; }
};
```

```
class B {
  private:
    A * ref;
  public:
    B() {}
    ~B() { delete ref; }
    A getRef() { return ref; }
    void setRef(A ref) { this->ref = ref; }
};
```

Aggregation vs. Komposition

Aggregation (Java)

```
B typeBObject = new B();
typeBObject.setRef(new A());
A obj = typeBObject.getRef();
typeBObject = null;
int x = obj.getM();
```

Aggregation (C++)

```
B * typeBObject = new B;
typeBObject->setRef(new A);
A * obj = typeBObject->getRef();
delete typeBObject;
typeBObject = NULL;
int x = obj->getM();
```

Komposition (Java)

```
B typeBObject = new B();
typeBObject.setRef(new A());
A obj = typeBObject.getRef();
typeBObject = null;
int x = obj.getM();
```

Vorsicht!

Komposition (C++)

```
B * typeBObject = new B;
typeBObject->setRef(new A);
A * obj = typeBObject->getRef();
delete typeBObject;
typeBObject = NULL;
int x = obj->getM();
```

Crash!

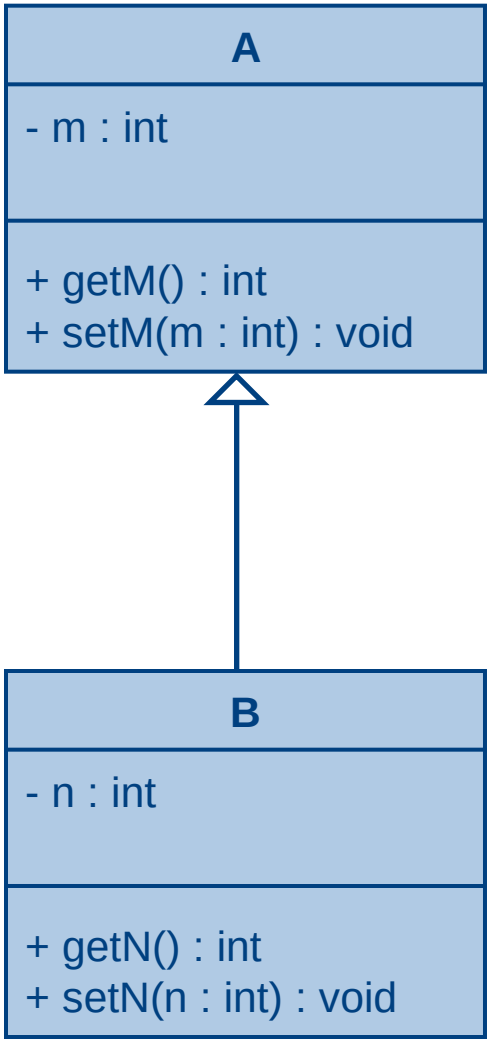
Kardinalitäten


```
public class B {
  private A ref;
}
```


```
public class B {
  private LinkedList<A> ref;
}
```

mögliche Realisierung

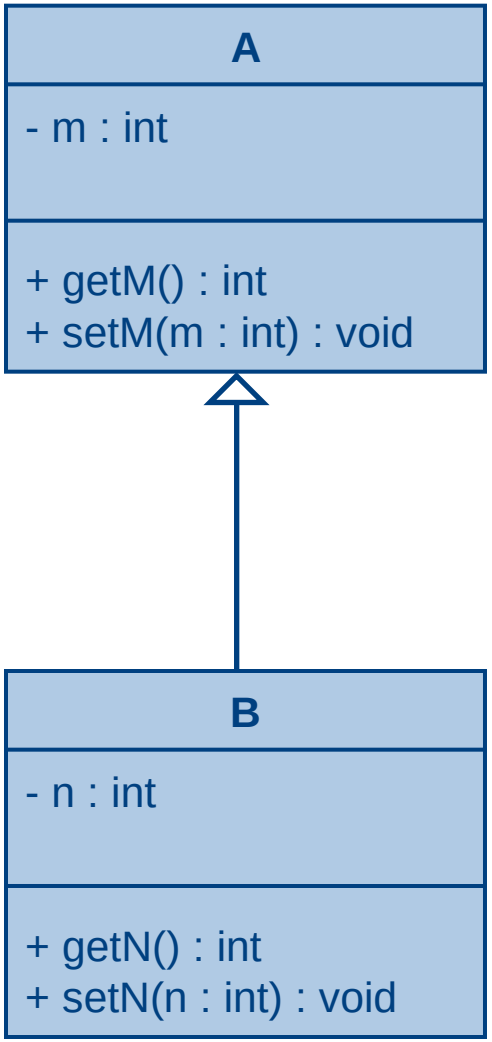
Vererbung




```
public class A {
  private int m;
  public int getM() { return m; }
  public void setM(int m) { this.m = m; }
}
```

```
public class B extends A {
  private int n;
  public int getN() { return n; }
  public void setN(int n) { this.n = n; }
}
```

Vererbung



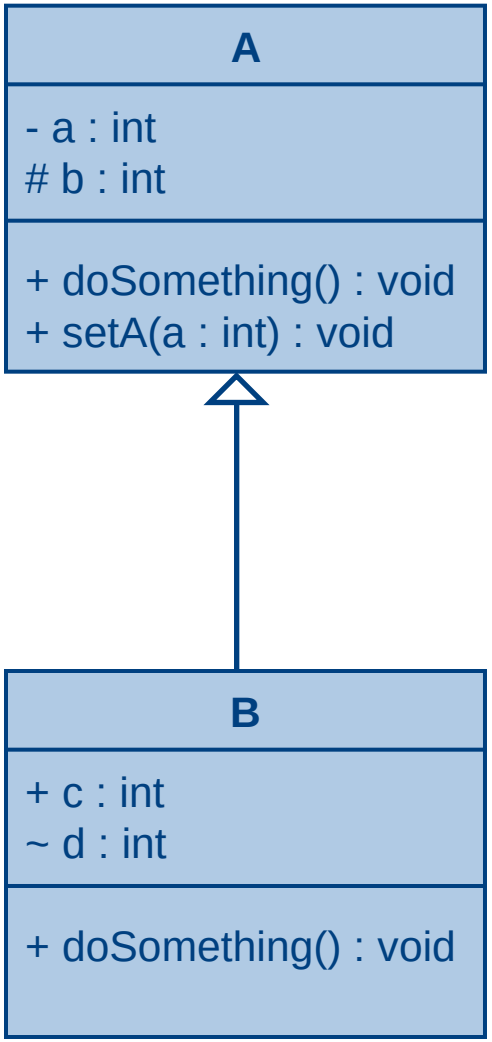

```
A obj = new A();
obj.getM();
obj.getN();
```

```
A obj = new B();
obj.getM();
obj.getN();   // ((B)obj).getN();
```

```
B obj = new B();
obj.getM();
obj.getN();
```

```
B obj = new A();
```

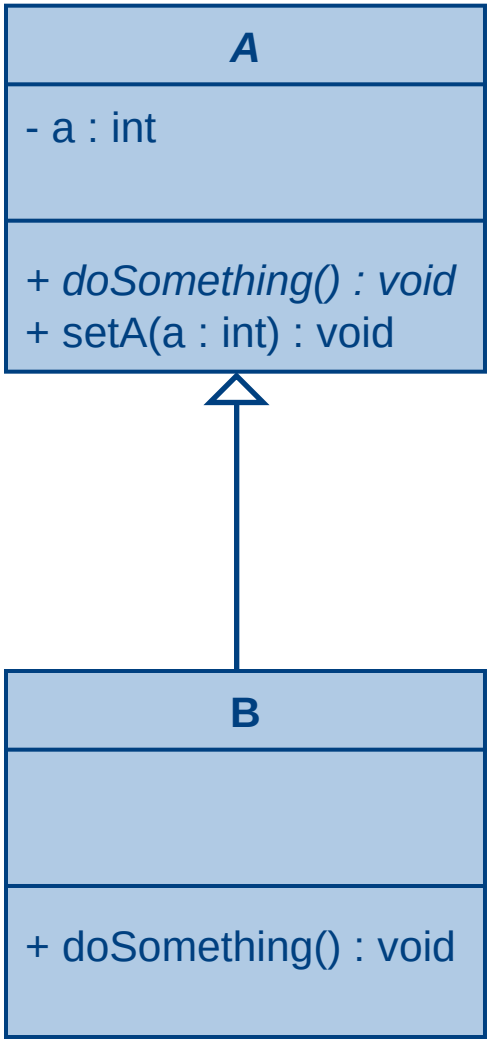
Vererbung




```
public void doSomething() {
   a = 1;
   b = 2;
   c = 3;
   d = 4;
}
```

```
public void doSomething() {
   a = 1;      //setA(1);
   b = 2;
   c = 3;
   d = 4;
}
```

Abstrakte Klassen



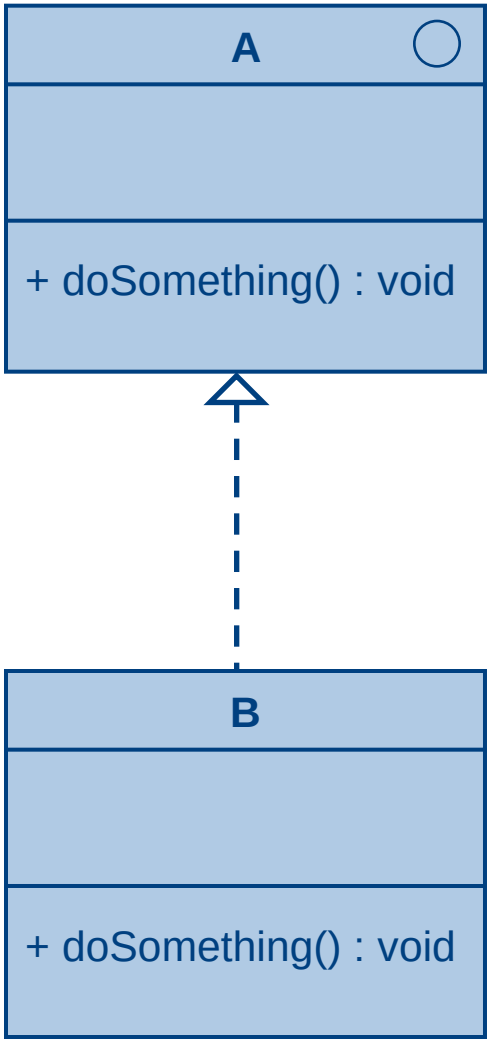

```
public abstract class A {
  private int a;
  public abstract void doSomething();
  public void setA(int a) { this.a = a; }
}
```

~~A obj = new A();~~

```
public class B extends A{
  public void doSomething() { setA(42); }
}
```

```
A obj = new B();
```

Interfaces




```
public interface A {
    void doSomething();
}
```

~~`A obj = new A();`~~

```
public class B implements A{
  public void doSomething() { System.out.println(42); }
}
```

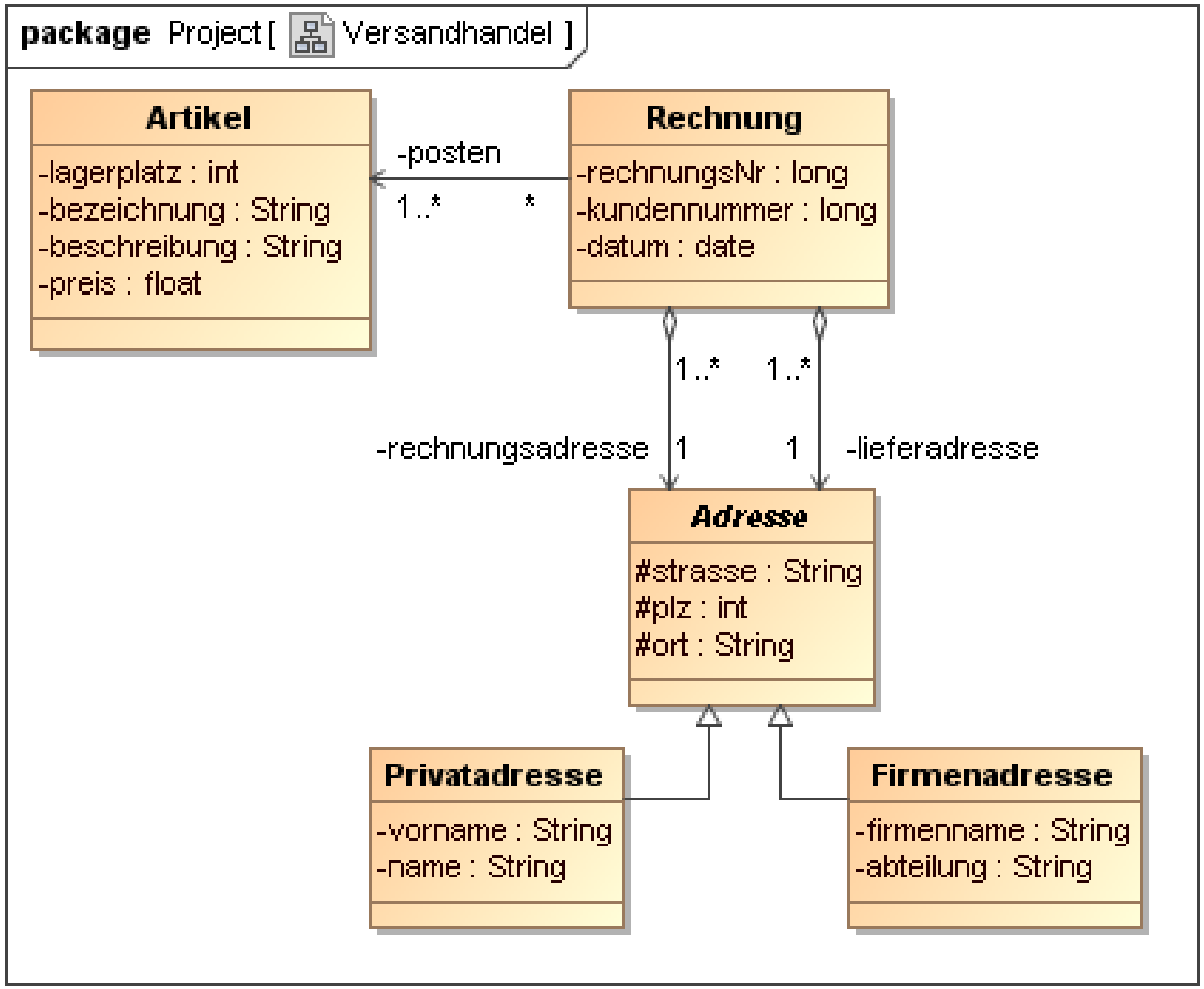
```
A obj = new B();
```

Aus einem Kundengespräch

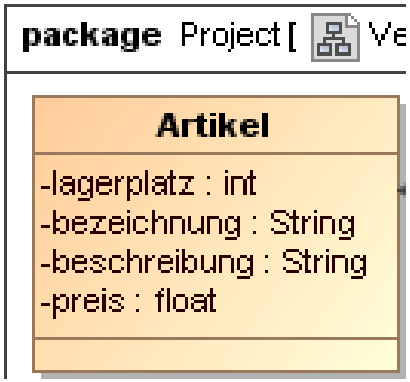
…und wie Sie sehen können, haben alle Artikel in unserem Lager einen festen Platz mit Lagerplatznummer. Außerdem soll zu jedem Artikel auch die Bezeichnung, der Preis und eine kurze Beschreibung angezeigt werden.

Eine Rechnung enthält, neben den Rechnungsposten, eine eindeutige Rechnungsnummer, die Kundennummer, ein Rechnungsdatum und zwei Adressen. Eine ist die Rechnungsadresse, die andere die Lieferadresse. Wir trennen beide selbst dann, wenn die Adressen identisch sind. Wir unterscheiden zwischen Privatadressen und Firmenadressen. Bei Firmenadressen gibt es die Felder Vorname und Name nicht, dafür aber den Firmen- und Abteilungsnamen. Die Kundennummer ist einfach eine Zahl. Vielleicht können wir sie in Zukunft nutzen um alle Bestellungen eines Kunden zu suchen. Ist es möglich, die Software bereits in 4 Wochen…

Klassendiagramm

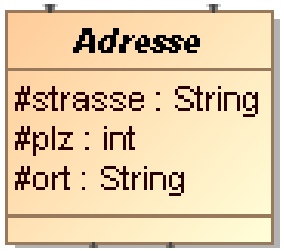


Klasse Artikel



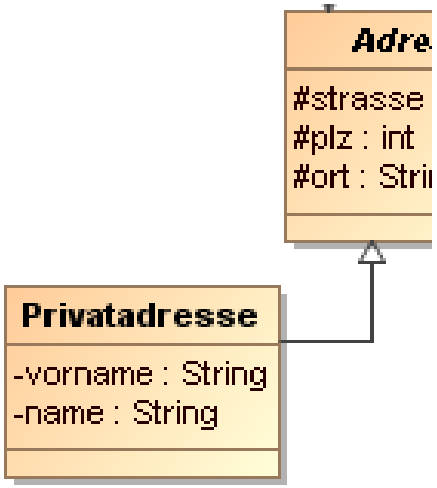

```
public class Artikel {

    private int lagerplatz;
    private String bezeichnung;
    private String beschreibung;
    private float preis;

    public String getBeschreibung() {
        return beschreibung;
    }

    public void setBeschreibung(String beschreibung) {
        this.beschreibung = beschreibung;
    }

    public String getBezeichnung() {
        return bezeichnung;
    }
   ...
}
```

(abstrakte) Klasse Adresse



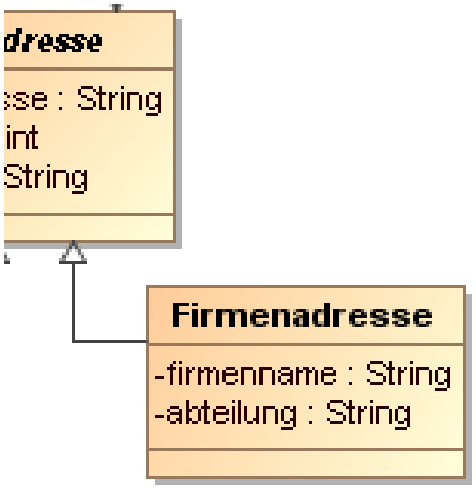

```
public abstract class Adresse {

    protected String strasse;
    protected int plz;
    protected String ort;

    public String getOrt() {
        return ort;
    }

    public void setOrt(String ort) {
        this.ort = ort;
    }

    public int getPlz() {
        return plz;
    }
    ...
}
```

Klasse Privatadresse



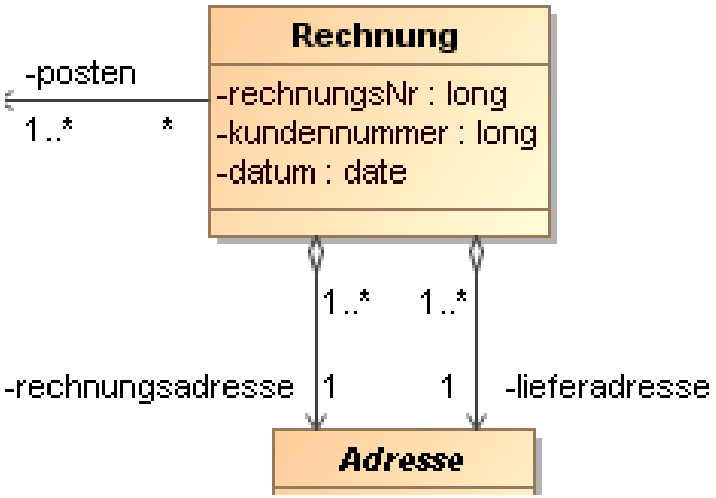

```
public class Privatadresse extends Adresse {

    private String vorname;
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
   ...
}
```

Klasse Firmenadresse




```
public class Firmenadresse extends Adresse {

    private String firmenname;
    private String abteilung;

    public String getAbteilung() {
        return abteilung;
    }

    public void setAbteilung(String abteilung) {
        this.abteilung = abteilung;
    }
   ...
}
```

Klasse Rechnung




```
public class Rechnung {
    private Adresse rechnungsadresse;
    private Adresse lieferadresse;
    private long rechnungsNr;
    private long kundennummer;
    private Date datum;
    private LinkedList<Artikel> posten;

    public Rechnung(Artikel a, Adresse r, Adresse l) {
        posten = new LinkedList<Artikel>();
        posten.add(a);
        rechnungsadresse = r;
        lieferadresse = l;
    }
    public Iterator<Artikel> postenIterator() {
        return posten.iterator();
    }
    public void addPosten(Artikel posten) {
        this.posten.add(posten);
    }
    public void removePosten(Artikel posten) {
        this.posten.remove(posten);
    }
   ...
}
```