# Nachdenkzettel Vererbung/Beziehungen

**1.** “Class B extends X“. Jetzt fügen Sie eine neue Methode in X ein. Müssen Sie B anpassen?

*Nein sofern die neue Methode nicht abstrakt ist, muss diese nicht in B implementiert werden, sondern steht direkt zu Verfügung.*

**2.1** Class B extends X {

public void newMethodinB() { …. }

}

Jetzt fügen Sie eine neue public Methode in ihre abgeleitete Klasse ein. Sie möchten diese neue Methode im Code verwenden. Prüfen Sie die folgenden Codezeilen:

X x = new B();

x.newMethodinB();

Was stellen Sie fest?

*Da newMethodinB() in B definiert wurde kann man von einem aus X instantisierten Objekt (x) nicht darauf zugreifen. Vererbung geht nur nach unten niemals nach oben in der Hierarchie.*

**2.2** Class B extends X {

@override

public void methodinB() { …. }

}

Jetzt überschreiben Sie eine Methode der Basisklasse in ihrer abgeleiteten Klasse. Sie möchten diese neue Methode im Code verwenden. Prüfen Sie die folgenden Codezeilen:

X x = new B();

x.methodinB();

Was stellen Sie fest?

*Da wieder ein Objekt aus X instantisiert wird, kann nicht auf die überschriebene Methode ein B zugegriffen werden. Nur auf die schon in X definierte, sollte diese nicht abstrakt sein.*

**3.** Versuchen Sie „Square“ von Rectangle abzuleiten (geben Sie an welche Methoden Sie in die Basisklasse tun und welche Sie in die abgeleitete Klasse tun).

```
public class Rectangle{
        private int width, height;
        public Rectangle(int width, int height){
                this.width = width;
                this.height = height;
        }
        public int getArea(){
                return width * height;
        }
        public int getCircumfrence(){
                return 2*width + 2*height;
        }
        Public int getWidth(){ return width; }
        Public int getHeight(){ return height; }
}
```

```
public class Square extends Rectangle{
        public Square(int side){
                super(side, side);
        }
}
```

**4.** Jetzt machen Sie das Gleiche umgekehrt: Rectangle von Square ableiten und die Methoden verteilen.

```
public class Square{
        private int width;
        public Square(int side){
                this.width = side;
        }
        public int getArea(){
                return width * width;
        }
        public int getCircumfrence(){
                return 4*width;
        }
        public int getWidth(){ return width; }
}
```

```
public class Rectangle extends Square{
        private int height;
        public Rectangle(int width, int height){
                super(width);
                this.height = height;
        }
        @Override
        public int getArea(){
                return getWidth() * height;
        }
        @Override
        public int getCircumfrence(){
                return 2*getWidth() + 2*height;
        }
        public int getHeight(){ return height; }
}
```

**5.** Nehmen Sie an, „String“ wäre in Java nicht final. Die Klasse Filename „extends“ die Klasse String.

Ist das korrekt? Wie heisst das Prinzip dahinter?

*Da die Syntax „extends“ bei normalen und abstrakten Klassen benutzt wird ist hier die Benutzung durchaus korrekt. Das Prinzip dahinter ist das der Vererbung bzw. „ist-eine“ – Beziehung, da die Klasse „Filename“ eine „String“ ist.*