Nachdenkzettel Beziehungen/Vererbung

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

↳ Not necessary

2. 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? Unable to call method because the object is an instance of class X (not class B)

2. Class B extends X {

```
    @override
    public void methodinB() { …. }
```

}

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

```
X x = new B();
x.methodinB();
```

Was stellen Sie fest? The code calls the corresponding method in the child class (even though the object an instance of class X

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)

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

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?

A filename is a String, thus the class can in fact be implemented as a derived class from class "String" using inheritance.

3)

Rectangle

getLength
getWidth
setLength
setWidth
getArea
getCircumference
toString
equals

Square (extends Rectangle)

setSize (len/wid)
getSize (len/wid)

toString (override)
equals (override)

4)

Square

getSize
setSize
getArea
getCircumference
toString
Equals

Rectangle (extends Square)

getSize
setSize
getArea
getCircumference
toString
Equals

} Override