Nachdenkzettel Beziehungen/Vererbung

1. „Class B extends X“. Jetzt fügen Sie eine neue Methode in X ein. Müssen Sie B anpassen?
No, except X is abstract and it added an abstract Method. In this case B has to add it to.

---

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?
This should work.

---

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?
It should probably work, if the code does not create a specific object of X because X is probably abstract (@override).

---

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)

BaseClass: public Rectangle(int x, int y) ,getX(), getY(), getArea(), toString()
SubClass: public Square(int x) { super(x,x) } , toString()

---

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

BaseClass: public Square(int x) , toString(), getX(), getAreaSqr(),
SubClass: public Rectangle(int x, int y) { super(x), this.y = y }, getY(), getArea(), toString()

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?

Yes, it´s called inheritance. (“is-a”).