Cazim Ukela 40064 | Daniel Hiller 40181 | Niklas Mäckle 40060

Nachdenkzettel Beziehungen/Vererbung

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

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?

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?

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?

Cazim Ukela 40064 | Daniel Hiller 40181 | Niklas Mäckle 40060

Antworten:

1. Nein, da die **Klasse B** von der **Klasse X** erbt, übernimmt sie auch deren Methoden und muss somit nicht angepasst werden

2. Das **x** Objekt kennt die Methode *newMethodinB()* nicht. Dies liegt daran, dass das Objekt mit **X** initialisiert wurde, die Methode allerdings in der **Klasse B** liegt. Möchte man mit **x** also auch diese Methode verwenden, so muss man in der **Klasse X** auch diese Funktion unterbringen.

   Überschreibt man die Methode *methodinB()* in der **Klasse B** und verwendet **x** dann wie vorher auch wird die überschriebene Methode in **B** verwendet.

3. In der Klasse **Rectangle** müssen nur *gegenüberliegende Geraden* die gleiche Länge haben. Bei **Square** jedoch *alle 4 Seiten* die gleiche. So kann man im Rectangle Getter und Setter für beide Seiten separat unterbringen. Außerdem werden dem Konstruktor zwei verschiedene Parameter für Weite und Länge des Rechtecks übergeben. In der Klasse square benötigt man einen eigenen Setter bzw Getter, der Weite und Länge gleichzeitig setzt. Dies ist nötig, damit sich die Seitenlängen abhängig voneinander verhalten und somit kein „falsches“ Quadrat entstehen kann, dessen Länge und Breite nicht gleich sind.

4. Macht man das Gleiche wie in 3. umgekehrt, reicht in der **Square** Klasse eine Methode, die die Seitenlänge festlegt. Die Klasse **Rectangle** kann mit dieser Methode jedoch recht wenig anfangen und benötigt wieder zwei Methoden, die unabhängig voneinander Länge und Weite des Rechtecks festlegen können.

5. Da ein Filename die gleichen Methoden benötigt wie ein String ist es nicht nötig hier mit „extends“ zu arbeiten. So wäre Filename lediglich ein Objekt vom Typ String, dass dann die gleichen Methoden wie ein String verwenden kann.