Nachdenkzettel Beziehungen/Vererbung

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

Nein, man kann aber einen Override schreiben wenn man denn möchte.

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?

Da x vom Typ X ist, X diese methode aber nicht besitzt, kann sie nicht angesteuert werden.

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?

Da x vom Typ X ist und die Methode in X existiert und in B nur überschrieben wird, kann sie verwendet werden.

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)

Class Rectangle

int a;
int b;

area(){return a*b;}
circumference(return 2*a + 2*b)

Class Square extends Rectangle

No methods needed

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

Class Square

area(){return a*a}
circumference(){return a*4}

Class Rectangle extends Square

@Override
area(){return a*b;}

@Override
circumference(){return 2*a + 2*b;}

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?

Ja, Final können nicht vererben
Polymorphie