Nachdenkzettel Beziehungen/Vererbung

1. „Class B extends X“.

Jetzt fügen Sie eine neue Methode in X ein. Müssen Sie B anpassen?

If the method in X is public or protected (in case they are in the same package), it will be possible to use it in class B when calling it properly.

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?

```
static class letterX {
    public static void sayIt() {
        System.out.println("Saying X");
    }
}

static class letterB extends letterX {
    public void newMethodinB() {
        sayIt();
    }
    public static void main(String[] args) {
        letterX obj = new letterB();
        obj.newMethodinB();

}
```

It is not going to work because newMethodinB can not be solved. While building the object we are using the newMethodinB inside the letterX class. If there is none, it cannot be read.

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?

```
static class letterX {
    public static void saylt() {
        System.out.println("Saying X");
    }

    protected void newMethodinB() {

    }
}

static class letterB extends letterX {
    @Override
    public void newMethodinB() {
        saylt();
    }
    public static void main(String[] args) {
        letterX obj = new letterB();
        obj.newMethodinB();
    }

}
```

Putting @override and creating a newMethodinB() class, will solve the problem of the previous question and make the code more organized and easier to understand.

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)

```
static class Rectangle {
    public static void saylt(String type) {
        System.out.println("I am a: " + type);
    }
    public void edgeSum(int x, int y, int z, int h) {

    }
}

static class Square extends Rectangle {
    @Override
    public void edgeSum(int x, int y, int z, int h) {
        int sum = x + y + z + h;
        System.out.println("The sum of my edges is: " + sum);
    }
    public static void main(String[] args) {
        Rectangle obj = new Square();
        obj.saylt("Rectangle");    //Use method direct from Base class
        obj.edgeSum(8, 8, 8, 8);  //Use method overrided at child class
    }
```

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

```
static class Rectangle extends Square {
    public static void saylt(String type) {
        System.out.println("I am a: " + type);
    }
    public void edgeSum(int x, int y, int z, int h) {
```

```
        int sum = x + y + z + h;
        System.out.println("The sum of my edges is: " + sum);
    }

}

static class Square  {
    public static void main(String[] args) {
        Rectangle obj = new Rectangle();
        obj.saylt("Rectangle");
        obj.edgeSum(8, 8, 8, 8);
    }
}
```

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?

No. It is not correct. Because even though “String” would not be final, it is still a variable type. So we cannot extends int, double, char...

Besides that, the String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values **cannot be changed after they are created.** String buffers support mutable strings. Because String objects are immutable they can be shared.