Nachdenkzettel: Interfaces und Software-Architektur

1. Spezifizieren Sie das Interface „Stecker“ für diese Implementation.


```
public interface Plug {
    void plugIn();
    void pullOut();
}
-------------------------------------------------
public interface Neutral_Contact {
}
-------------------------------------------------
public interface Ground_Contact {
    void protectAgainstElectricShock();
}
-------------------------------------------------
public interface Live_Contact_EU_Voltage {
    void carryAlternatingCurrent();
}
-------------------------------------------------
public class Schuko_Plug implements Plug, Ground_Contact,
Live_Contact_EU_Voltage, Neutral_Contact{

    @Override
    public void plugIn() {
        System.out.println("Plugged in!");
    }

    @Override
    public void pullOut() {
        System.out.println("Pulled out!");
    }

    @Override
    public void protectAgainstElectricShock() {
        System.out.println("Stay protected!");
    }
```

```
    @Override
    public void carryAlternatingCurrent() {
        System.out.println("Carrying 220V - as every socket and plug do");
    }
}
```

2. Ist das a) eine korrekte Ableitung von der obigen Implementation?
   b) eine korrekte Implementation Ihres Interfaces

   a) Nein, denn es bietet eine andere Art von Bodenkontakt

   b) Es implementiert alle Methoden des obigen Stecker Interfaces, so dass auch eine französische Steckdose und ein französischer Stecker als gültig angesehen werden könnend

copyright hic et nunc, Cc-by-sa-3.0-migrated

3. Und das?

Ja, die Implementierung für die Implementation des Plug ist gültig, aber kein gültiges Kind vom Schuko-Plug, da es keinen Bodenkontakt hat und daher das Interface des Ground_Contact nicht implementiert

Autor: somnusde, wikimedia-commons, PD

4. Wie seht es mit 220 V aus? Interface oder Implementation? Und das Material des Schukosteckers?

Wir denken, dass es sich um eine Interface handelt, aber der logischere Ansatz wäre, eine Implementierung bereitzustellen. Aufgrund unserer Architektur haben wir beschlossen, ein Plug-In aus mehreren Komponenten zu erstellen.

5. Wieviel Spass hätten wir ohne die DIN Norm für Schukostecker oder Eurostecker?

6. Was gehört alles zum „Interface einer Klasse“ in Java? (Anders formuliert für UX-Leute: wenn ich von jemandem eine Klasse in meinem Code benutze: was ärgert mich, wenn es geändert wird?)
Alles, was kaum damit zu tun hat, wie ich eine Klasse als Externe verwende. Meistens - vorausgesetzt, der Programmierer der Klasse, die ich verwenden möchte, hält sich an gängige Java-Standards.

7. „Class B implements X“. Jetzt fügen Sie eine neue Methode in Interface X ein. Was passiert?

Klasse B muss die neue Methode implementieren. Andernfalls erhalten Sie einen Compilerfehler.

8. Zwei Interfaces sind nicht voneinander abgeleitet, haben aber zufällig die gleiche Methode. Können Sie Implementationen dieser Interfaces polymorph behandeln?

```
Interface X {                    Interface Y {                    class B implements Y  { ...}
  public void foo();               public void foo();
}                                  }
X x = new B();  ??
x.foo();          ??
```

Nein, es kann nicht polymorph behandelt werden.

9. Ihr code enthält folgendes statement: X xvar = new X();

Was ist daran problematisch, wenn Sie eine Applikation für verschiedene Branchen/Kunden/Fälle bauen?

10. Von ArrayList ableiten oder eigene Klasse „Catalog“ oder ähnlich bauen und ArrayList<> verwenden? Sprich: soll man von Java Basisklassen ableiten? Beispiele: Vegetable, VegetableCatalog Task, TaskList, GameObject, GameObjectList etc.