Nachdenkzettel: Interfaces und Software-Architektur

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

copyright Aunkrig, CC-BY-SA-4.0

static final material;
static final durchmesser;
static final kontaktstiftAnzahl;
static final kontaktstiftDurchmesser;
static final kontaktstiftLänge;
static final kontaktstiftAchsenastand;

public static final void einstecken();
public static final void ausstecken();
public static final void istDreiadrig();

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

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

Korrekte Implementation, da alle Attribute und Methoden implementiert wurden.

3. Und das? Autor: somnusde, wikimedia-commons, PD

Korrekte Implementation, da alle Attribute und Methoden implementiert wurden.

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

220 V und das Material sind ein Interface, da sie implementiert werden können.

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

Wäre problematisch, denn dann hätte jeder Hersteller z.B. einen eigenen Stecker.
Nicht jeder Stecker würde mehr in jede Steckdose passen.

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?)

-Zum Interface einer Klasse gehören konstante Attribute und Methoden.
-Wenn in der Klasse etwas geändert wird, kann es vorkommen, dass bestimmte Funktionen nicht mehr funktionieren.

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

Dann steht auch diese Methode in B zur Verfügung.

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

```
Interface X {
  public void foo();
}
```

```
Interface Y {
   public void foo();
}
```

```
class B implements Y  { ...}
```

```
X x = new B();  Funktioniert
x.foo();
```

Funktioniert nicht, es nicht spezifiziert werden kann, auf welche Methode der Aufruf zugreifen soll. Deswegen funktioniert Polymorphie nicht.

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

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

Der Nutzer ist eingeschränkt, da im Fall von *X xvar = new X();* nur ein Objekt vom Typ X erstellt werden kann. Verschiedene Kunden/Branchen haben verschiedene Anforderungen/Bedürfnisse, weshalb man eine Factory verwenden sollte, die das gewünschte/spezifische Objekt zurückliefert.