

## Nachdenkzettel: Interfaces und Software-Architektur

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



copyright Aunkrig, CC-BY-SA-4.0

- Maße vom Stecker
- Belegung des Steckers (Abstand, Dicke, Länge der Stifte)
- Anzahl der Stifte
- Erdung
- Material
- Form

### 2. Ist das a) eine korrekte Ableitung von der obigen Implementation?

-Nein, die Erdung ist anders, Interfaces sind inkompatibel

### b) eine korrekte Implementation Ihres Interfaces

-Nein



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



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

Nein, es fehlt die Erdung. Es würde reinpassen, aber es ist keine Ableitung.  
Unvollständiges Interface.

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

Beides Interfaces. 220V gehören zur DIN Norm.

**5. Wie viel Spaß hätten wir ohne die DIN Norm für Schukostecker oder Eurostecker?**

Stecker könnten nicht zum Anschluss passen, was zu Stromschlägen und Bränden führen könnte.  
Gefahr für Verbraucher, durch falsche Erdung.

**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?)**

- Methoden (public) wie sie beschrieben sind,
- Signaturen (Parameter)
- return type

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

B ist keine vollständige Implementierung von X mehr, man muss die Implementierung der neuen Methode auch in B durchführen.

**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(); x.foo();

geht nicht, da B Y implementiert und nicht X. Methodenaufruf funktioniert auch nicht. Sie sind nicht voneinander abgeleitet. Sie sind zwei komplett unterschiedliche Typen.

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

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

Jeder Kunde kann unterschiedliche Anforderungen und Geschäftsideen haben, die sich auf die Klasse x auswirken können. Stellt die Klasse x z.B. ein Auto dar, kann es ein Problem sein, wenn ein anderer Kunde ein Motorrad will.

Dort wo ich Änderungen erwarte, kann ich das nicht verwenden.

**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.**

Bei Ableitung kann man keine speziellen Dinge mehr hinzufügen.

Man sollte ein Neue bauen wenn man spezielle Anforderungen hat. Von ArrayLists nichts ableiten.