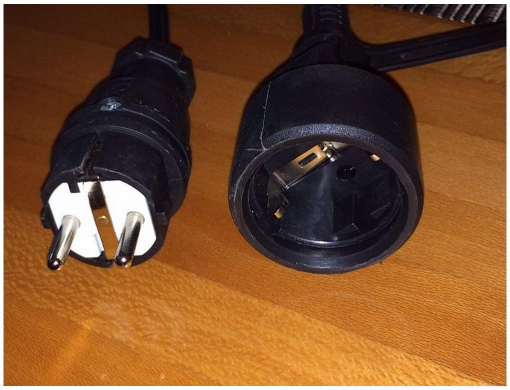
Nachdenkzettel: Interfaces und Software-Architektur

……………………………………………………………
Vorname, Name, Matrikelnummer

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

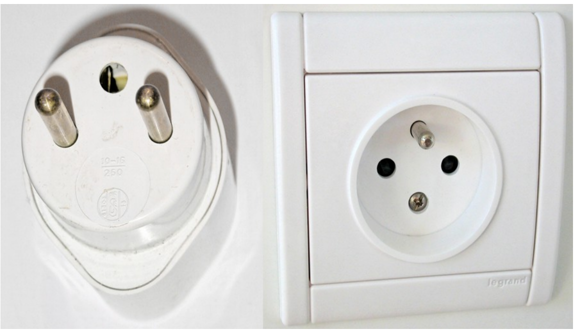


copyright Aunkrig, CC-BY-SA-4.0

```
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();
}
-------------------------------------------------
```

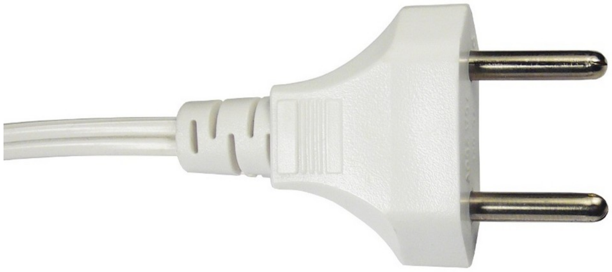
```
public class Schuko_Plug implements Plug, Ground_Contact, Live_Contact_EU_Voltage,
Neutral_Contact{

    @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



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

a) No, because it provides another type of ground contact, (b) however it implements all the methods of above Plug-interface and so even a french socket and plug can be considered valid (even though they look really weird)



3. Und das? Autor: somnusde,

wikimedia-commons, PD

3.)

It is a valid implementation for the Plug-interface, however it is not a valid child of Schuko-Plug as it is lacking a ground connector and thus does not implement the Ground_Contact-interface.

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

We have decided to include it as an interface, but the more logic approach would be to employ it through implementation. But due to our architecture we decided to build a Plug from multiple components.

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

It would be a mess because we could not be sure about the way a device connects before we look at it. Or, even after we look at it because it could be running on 110V although it uses a 220V plug.

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?)

Everything that is hardly related to how I use a class as an external. Mostly – given that the programmer of the class I want to use complies with common java standards, such as using encapsulation – these will be methods.

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

Class B has to implement the new method. Otherwise you’ll get a compiler error.

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();        ??
```

No, they cannot.

9. Ihr code enthält folgendes statement: X xvar = new X();
Was ist daran problematisch, wenn Sie eine Applikation für verschiedene Branchen/Kunden/Fälle bauen?

If X is a class implementing an interface you should use the interface type for variables because that way implementations can easily be swapped (e.g. use „List<Object> list = new ArrayList<>()“ instead of „ArrayList<Object> list = new ArrayList<>()“)

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.

Rather than inheriting from base classes you should implement the same respective interfaces in order to achieve what you want.