Nachdenkzettel Clean Code

1. Klassenexplosion (Schwierig..)

class Formularfeld;
class Textfeld extends Formularfeld;
class Zahlfeld extends Formularfeld;
class TextUndZahlFeld extends Formularfeld;
class TextfeldOCR extends Textfeld;
class ZahlfeldOCR extends Zahlfeld;
class TextUndZahlFeldOCR extends TextUndZahlFeld;
class TextfeldSonderZ extends TextUndZahlFeld;
class TextfeldOCRSonderZ extends TextUndZahlFeldOCR;
class .....

> Jede weitere Eigenschaft oder Spezialisierung führt zu vielen neuen Klassen durch Kombination. Die Folge ist explosives Anwachsen der Zahl der Klassen mit identischem Code. (Lösung?)

There could be 2 solutions.

a) Make a class Factory

b) Use Interfaces

2. Der verwirrte und der nicht-verwirrte Indexer
was genau unterscheidet die beiden Indexer? Wieso ist der eine „verwirrt“?

With the confused indexer, there is a possibility of contradicting values within the object. You could set the languageIsoCode to german and the languageId to french. There is no option in the class to check if the IsoCode is the same language as the languageId. Another weird thing is that the confused indexer uses only setters instead of class constructors.

In the non confused indexer this problem is solved by actually using two different constructors. In these constructors each attribut is either checked and mapped or checked and stored to prevent the above mentioned problem.

3. Korrekte Initialisierung und Updates von Objekten

```
public class Address {

	private String City;
	private String Zipcode;
	private String Streetname;
	private String Number;

	public void setCity (String c) {
		City = c;
```

```
        }
        public void setZipcode (String z) {
                Zipcode = z;
        }
}
```

Wie initialisieren Sie Address richtig? Wie machen Sie einen korrekten Update der Werte?

Address should be initialized by using a constructor. City and zipcode should be in one constructor and Streetname and Number in another constructor.

4. Kapselung und Seiteneffekte

```
 public class Person {

        public Wallet wallet = new Wallet();
        int balance = 0;

        public Wallet getWallet(void) {
                return wallet;
        }

        public addMoney(int money) {
                wallet.add(money);
                balance = wallet.size();

        public int getBalance() {
                return balance;
        }
}
```

Reparieren Sie die Klasse und sorgen Sie dafür, dass die Gültigkeit der Objekte erhalten bleibt und keine Seiteneffekte auftreten.

```
 public class Person {

        private Wallet wallet = new Wallet();
        int balance = 0;

        private Wallet getWallet(void) {
                return wallet;
        }

        public addMoney(int money) {
                wallet.add(money);
                balance = wallet.size();
        }
```

```
        private int getBalance() {
                return balance;
        }
}
```