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?)

Kompositionen statt Vererbungen.

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

Bei einem Zustand mit LanguageID, beim anderen LanguageID und IsoCode
➔ Inkonsistenz
Bei einem Immutable beim anderen nicht.
➔ Schutz vor inkonsistenz

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?

Alle Felder müssen nach dem Konstruktor Aufruf initialisiert sein.
Bei Update müssen alle Felder auf Konsistenz geprüft werden.

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();

        public addMoney(int money) {
                wallet.add(money);
        }

        Public int getBalance() {
                return wallet.size();
        }
}
```