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?)

Interface/abstracte Klasse mit extra Methode für Spezialisierung oder alle Spezielisierung in eine Klasse und dabei Funktionen beim Instanziieren mit true enablen oder disablen.

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

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?

Man sollte immer ein gültiges Objekt erzeugen, ändert man nur die Stadt, stimmen Straßennamen, Postleitzahl usw. nicht mehr überein. Es sollte nur einen setter geben der alles ändert oder man erzeugt direkt ein neues Objekt.

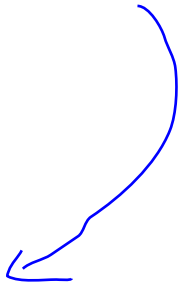
4. Kapselung und Seiteneffekte

```
public class Person {

    ~~public~~ Wallet wallet = new Wallet();   //Parameter sonst ungültiger Wallet
    ~~int balance = 0;~~   //Annahme: balance ist wallet.size() -> wir es nicht gebraucht sondern über die Klassenmethoden geholt

    ~~public Wallet getWallet(void) {~~
        ~~return wallet;~~
    }

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

    public int getBalance() {
        ~~return balance;~~   return wallet.size()
    }
}
```

private

private

in wallet Klasse/ braucht man die Methode überhaupt???



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

Kein Zugriff von außen auf das komplette Objekt, sondern nur auf das was man braucht, bzw. an alle weitergeben kann