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?)

Eine mögliche Lösungsidee ist, statt Vererbung Boolsche Variablen zur Auswahl gewünschter Funktionalitäten zu nehmen. In der Basisklasse erfolgt dann ein Aufruf an die zu aktivierende Methode.

Aber: die Lösung stößt auch schnell an ihre Grenzen, da die Basisklasse zu einer One-for-all Funktionssammlung von Optionen degeneriert wird. Anpassung von Klassen durch neue Methoden erzwingen immer eine Anpassung beim Verwender.

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

Der verwirrte Indexer hat 2 Member-Variablen langID und langISOCode die eigentlich einem Constraint unterworfen sein sollten, da die ID immer zum ISO-Code passen muss. Dieser Constraint existiert aber nicht in diesem Code. Die Übereinstimmung beider Argumente muss vom User manuell überprüft werden, da dies nicht im Code passiert.

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?

Das Problem ist: Ändert man die Stadt, werden die anderen 3 Komponenten der Adresse ungültig -> die gesamte Adresse wird also ungültig und stimmt nicht mehr.

Lösung Idee: Konstruktor der alle 4 Argumente anfordert:

```
Public Address(String City, String Zipcode, String Streetname, String Number) {
This.city = city;
…
This.number = number;
}
```

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();
protected int balance = 1000000;

private Wallet getWallet(void) {
return wallet;
}
```

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

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