**Cazim Ukela 40064 | Daniel Hiller 40181 | Niklas Mäckle 40060**

**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?)

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?

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.

## Lösung

1.
Interface oder Composition

2.
Der Indexer wird im zusammenhang mit einer for schleife verwendet. Wenn man jetzt z.B. einen zwei-dimensionalen Array durchgehen will, muss man ja insgesammt 2 for-schleifen im zwei indexer benutzen. Dabei sollten die Indexernamen im zusammenhang mit der aufgabe stellen. Sonst ist das bei verschachtelten schleifen schwierig zu überblicken.

Gut:

```
for(int x = 0; x < array.length(); x++){
   for(int y = 0; y < array.length(); y++){
       value = array[x][y]
   }
}
```

Schlecht:

```
for(int o= 0; o < array.length(); o++){
   for(int p = 0; p < array.length(); p++){
        value = array[x][y]
   }
}
```

3.
Das Adress-Object sollte über den Konstruktor initialisiert werden. Unter anderem sollten werte die zusammen hängen wie Stadt und Postleitzahlt sowie Straße und Hausnummer zusammen gesetzt werden, um fehler zu vermeiden.

```
setCity(String c, String z){
    City = c;
    Zipcode = z;
}

setStreet(String s, String n){
   Streetname = s
   Number = n
}
```

4.

```
public class Person {
        public Wallet wallet = new Wallet();
}

public class Wallet{
        private int balance = 0;

        public addMoney(int money) {
                balance += money;

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