Nachdenkzettel Clean Code

Bianca Knülle, 43373/ Despoina Sfantou, 43368/ Julia Fellmeth, 43391/ Kethella Oliveira, 43385

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?)

A probable solution is using composition:

```
        interface FormularFeld()
                interface Textfeld exntends FormularFeld;
                interface Zahlfeld extends Formularfeld;
                interface OCRfeld extends Formularfeld;
                interface SonderZfeld 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;
```

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

The difference between confused and not confused indexers is the fact that the constructor of the confused indexer is empty and not initialised. In the not confused one, the are not setters and thus the code should be immutable.

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?

The problems in this code ist that the names of the Strings are capital and that if one value is changed, the others do not (the others are not affected and hence, the code doesn't make sense).

solution:

```
public class Address{

	private String city;
	private String zipcode;
	private String streetname;
	private String number;

	public Address (String city, String zipcode, String streetName, String number) {
		update (city, zipcode, streetName, number);
	}

	public void update (String city, String zipcode, String streetName, String number)
{
		this.city = city;
		this.zipcode = zipcode;
		this.streetName = streetName;
		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.

```
class Scratch{

        public static class Person{

                private final Wallet wallet = new Wallet( );

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

                public WalletView getWallet( ){
                        return wallet;
                }
        }

                public static void main(String[] args) {
                        Person person = new Person( );
                        person.getWallet( ).add( );
        }
                interface WalletView {
                        public int size( );
        }

                static class Wallet implements WalletView {
                        public void add (int money) {
```

```
                }

                public int size( ){
                        return 0;
                }
        }
}
```