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?)
Statt mehrfach zu vererben kann auch ein Interface genutzt werden. Dieses enthält konstante und abstrakte Klassen, welche von der Klasse implementiert. Diese wird wiederum einem Interface zugewiesen. Oder Composition statt Vererbung.

2. Der verwirrte und der nicht-verwirrte Indexer - was genau unterscheidet die beiden Indexer? Wieso ist der eine „verwirrt“?
Nicht-verwirrter Indexer: Veränderung aller abhängigen Felder
Verwirrter Indexer: Veränderung einzelner Felder (mgl. Logikfehler)

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?
Initialisieren: Adress adress= new Adress();
Geändert werden die Werte mit Settern, weil sie privat sind..
Updaten: changeAdress(City,Zipcode,Streetname,Number)

4. Kapselung und Seiteneffekte

```
public class Person {
public Wallet wallet = new Wallet();
      private int balance = 0;

      public Wallet getWallet() {
            return wallet;
      }

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

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

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