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?)

Anstatt für jedes Objekt ein *extends* zu verwenden, sollte man lieber eine Factory verwenden, die das gewünschte Formular-Objekt zurückgibt.

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

Verwirrte Indexer:
-Leerer Konstruktor
-Es werden Setter und Getter verwendet. Das hat zur Folge, dass bei einer Änderung nur z.B. eine Instanzvariable geändert wird, weshalb das Objekt ungültig bzw. beschädigt wird.

Nicht-verwirrte Indexer:
-Die Instanzvariablen werden direkt im Konstruktor initialisiert.
-Es sind keine Setter und Getter vorhanden.
-Wenn man die Sprache geändert werden möchte, wird diese nicht über einen Setter aktualisiert, sondern ein neues Objekt mit der neuen Sprache erstellt → 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?

Es wird ein Konstruktor verwendet, der alle Instanzvariablen initialisiert.

Zum Updaten der Werte wird ein neues Objekt erstellt, anstatt es über Setter zu aktualisieren.
Dabei werden alle Variablen im Konstruktor direkt initialisiert.

4. Kapselung und Seiteneffekte

```
 public class Person {

	private Wallet wallet = new Wallet();
	private int balance = 0;

	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.