Nachdenkzettel: Clean Code

1. Gegeben sind folgende Klassendefinitionen:

```
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;
//...
```

   Jede weitere Eigenschaft oder Spezialisierung führt zu vielen neuen Klassen durch Kombination. Die Folge ist explosives Anwachsen der Zahl der Klassen mit identischem Code. Wie können Sie dies umgehen?

2. Korrekte Initialisierung und Updates von Objekten

   Betrachen Sie die Klasse `Address`:

```
public class Address {
    private String city;
    private String zipCode;
    private String streetName;
    private String houseNumber;

    public void setCity (String c) {
        this.city = c;
    }
    public void setZipcode (String z) {
        this.zipCode = z;
    }

    // weitere Setter für alle Attribute
}
```

   Was kann bei der initialisierung von `Address`-Objekten schiefgehen? Wie korrigieren Sie dies? Wie ist Ihre Meinung bzgl. der Parameterbenennung der aufgeführten `set`-Methoden?

Nachdenkzettel

- Clean Code -

Aufgabe 1)

Man kann Code der in vielen Klassen benötigt wird über abstrahlte Klassen bereitstellen um Doppelungen zu vermeiden. Zusätzlich helfen Interfaces bei der Wartbarkeit, Erweiterbarkeit und Strukturierung des Codes.

Aufgabe 2)

Es könnten einem oder mehreren Attributen keine Werte zugewiesen sein. Dies kann man über einen Konstruktor vermeiden in dem alle Attribute als Parameter übergeben werden.
Bei den Variablen sollte man sprechende Namen benutzen, damit klar ist was gemeint ist. Man könnte hier city und zipFile als Beispiel nehmen.