12008073_AndreasMariusBaisan

5.3_Refactoring Steps

```java
public abstract class Person {
    public String vorname,nachname;
    public Integer alter;

    public Person(String vorname, String nachname, Integer alter) {
        this.vorname = vorname;
        this.nachname = nachname;
        this.alter = alter;
    }

    public String getVorname() {
        return vorname;
    }

    public String getNachname() { return nachname; }

    public Integer getAlter() { return alter; }

    public abstract String getJobBeschreibung();

}
```

Person Abstract gestaltet, nur die für jede einzelne Person relevanten Variablen in Klasse Person behalten, getJobBeschreibung belassen wir als Abstrakte Methode in Person, da wir diese Methode in Jeder von Person erbenden Klassen wollen

```java
public class Architekt extends Person {
    private Feld feld;

    public Architekt(String vorname, String nachname, Integer alter, Feld feld) {
        super(vorname, nachname, alter);
        this.feld = feld;
    }

    public Feld getFeld() {
        return feld;
    }

    @Override
    public String getJobBeschreibung() { return "Designed die Architektur des Projekts"; }
}
```

```java
public class Entwickler extends Person {
    private String bevorzugteProgrammierSprache;
    private Ide ide;
    private boolean datenbanken;

    public Entwickler(String vorname, String nachname, Integer alter, String bevorzugteProgrammierSprache, Ide ide, boolean datenbanken) {
        super(vorname, nachname, alter);
        this.bevorzugteProgrammierSprache = bevorzugteProgrammierSprache;
        this.ide = ide;
        this.datenbanken = datenbanken;
    }

    public String getBevorzugteProgrammierSprache() {
        return bevorzugteProgrammierSprache;
    }

    public Ide getIde() { return ide; }

    public boolean isDatenbanken() { return datenbanken; }

    @Override
    public String getJobBeschreibung() { return "Entwickelt Code."; }
}
```

```java
public class Tester extends Person {
    private String bevorzugtesTestFramework;

    public Tester(String vorname, String nachname, Integer alter, String bevorzugtesTestFramework) {
        super(vorname, nachname, alter);
        this.bevorzugtesTestFramework = bevorzugtesTestFramework;
    }

    public String getBevorzugtesTestFramework() {
        return bevorzugtesTestFramework;
    }

    @Override
    public String getJobBeschreibung() { return "Testet Code."; }
}
```

Die Verschiedenen Berufsmöglichkeiten als eigenständige Klasse angelegt, da Jede Person einen Beruf ausübt und jeweils den Berufen ihre Zuständige Variable hinzugefügt z.B. Tester – bevorzugtesTestFramework.
Außerdem die Methode getJobBeschreibung() jeweils jedem Berufstypen angepasst.

```java
public enum Feld {
    ENTERPRISE, APPLIKATION
}
```

```java
public enum Ide {
    ECLIPSE, INTELLIJ, NANO
}
```

Aus Feld und Ide jeweils ein Enum erstellt, da nur die dazugehörigen jeweils bestimmten Begriffe verwendet werden sollen.