Klassendefinitionen I

Lernziele

- Sie können eine einfache Klasse auf Papier analysieren und finden Fehler und Verstösse gegen die Clean Code Regeln.
- Sie haben auf Ihrem Laptop eine funktionierende BlueJ-Umgebung eingerichtet und können diese verwenden.
- Sie können einfache Klassen basierend auf einer Spezifikation und unter Berücksichtigung der Clean Code-Regeln programmieren und diese in BlueJ testen.

Aufgabe 1 (auf Papier!)

Die nachfolgende Klasse Buch dient dazu, ein Buch in einem Buchhandlungsprogramm zu modellieren. Die Funktion der Klasse und der Methoden ist in den Kommentaren angegeben. Bei der Implementierung haben sich aber einige Fehler und Unschönheiten bez. Clean Code eingeschlichen. Identifizieren Sie die Probleme und korrigieren Sie diese. Kopieren Sie die Klasse dabei nicht in eine Entwicklungsumgebung, sondern finden Sie die Fehler auf Papier. Markieren und korrigieren Sie die Probleme direkt im Code oder beschreiben Sie diese auf den leeren Zeilen nach dem Code.

```
/**
 * Die Klasse Buch repräsentiert ein Buch und beinhaltet
 * Buchtitel, ISBN-Nummer (ISBN-13, z.B. 978-3868949070)
 * und den Lagerbestand.
 * @author Marc Rennhard
 */
public class Buch {
    public String titel;
    private int isbn;
    // Der aktuelle Bestand
    private int b;

    /**
     * Erzeugt ein Buch mit den spezifizierten Werten für
     * Titel und ISBN-Nummer und setzt den Bestand auf 0.
     * @param buchTitel Der Buchtitel.
     */
    public Buch(String buchTitel) {
        titel = buchTitel
        isbn = buchISBN;
    }

    /**
     * Liefert den aktuellen Bestand.
     * @return Der Bestand.
     */
```

private String titel;

private String isbn;

private int bestand;

public Buch(String buchTitel, String buchISBN) {

```
    public int bestand() {
        return b;
    }

    /**
     * Verändert den Bestand um den angegebenen Wert.
     * @param veraenderung Die Veraenderung des Bestands.
     */
    public void veraendereBestand(int veraenderung) {
        b = b + veraenderung;
        return b;
    }

    /**
     * Gibt die Informationen eines Buches aus.
     */
    public String output() {
        System.out.println("Titel: " + titel);
        System.out.println("ISBN-13: " + isbn);
        System.out.println("Bestand: " + b);
    }
}
```

public int gibbestand () {

return bestand;

public void veraendereBestand (int veraenderung) {

bestand = bestand + veraenderung;

public void output () {

+ bestand);

Aufgabe 2

Installieren Sie eine funktionsfähige BlueJ-Umgebung auf Ihrem Laptop. Gehen Sie dazu auf http://www.bluej.org und dort auf den Download-Bereich. Verwenden Sie JDK 8 als Basis. Um die aktuell installierte Java-Version zu bestimmen, führen Sie **`java -version`** in einem Terminal aus.

Kopieren Sie ebenfalls das Verzeichnis Projekte von der Begleit-CD des Buchs auf Ihren Laptop. Damit erhalten Sie alle Beispiele, die im Buch verwendet werden.

Prüfen Sie dann, ob alles korrekt funktioniert. Öffnen Sie dazu in BlueJ den naiven Ticketautomaten aus Kapitel 2 und interagieren Sie mit diesem, um mit der Verwendung von BlueJ vertraut zu werden.

Aufgabe 3

Studieren Sie das Dokument *99_Anleitung-Arbeiten-mit-Git.pdf* bei den Praktikumsunterlagen auf OLAT. Überspringen Sie Abschnitte, die die Eclipse Entwicklungsumgebung betreffen. Studieren Sie diese Abschnitte sobald wir auf Eclipse umsteigen.

Aktivieren Sie anschliessend wie in der Anleitung beschrieben Git in Ihrer BlueJ-Umgebung und forken Sie das Projekt: https://github.engineering.zhaw.ch/prog1-kurs/02_Praktikum-1_Buch.
Nutzen Sie jetzt BlueJ um die eigene Projektkopie auf Ihren Computer zu holen und zu bearbeiten.

Kompilieren Sie die Klasse, was auf Grund der Fehler nicht funktionieren sollte. Gehen Sie nun wie folgt vor:

- Korrigieren Sie jeweils den gemeldeten Fehler und kompilieren Sie die Klasse erneut, bis sämtliche Fehler korrigiert wurden. Haben Sie jeden dieser Fehler bei der Analyse in Aufgabe 1 gefunden? Wenn nein, so geben Sie jeden übersehenen Fehler an und überlegen Sie sich, wieso Sie ihn übersehen haben

  ISBN kann nicht ein integer sein, da es nicht eine ganze Zahl ist, sondern ein Minus enthält. Daher muss es vom typ String sein.

- Sobald die Klasse kompiliert werden kann: Haben Sie in Aufgabe 1 weitere Fehler entdeckt, die der Compiler nicht findet? Wenn ja, geben Sie die Fehler an mit jeweils einer Begründung, wieso dem so ist. Korrigieren Sie auch diese Fehler im Code.

  Die Variable "String titel" sollte private sein. Public ist gemäss compiler ok. - Variable "b" ist nich eindeutig. Deshalb ist die Variable "bestand" besser.

- Hat Ihnen der Compiler Verstösse gegen die Clean Code-Regeln gemeldet? Wieso? Passen Sie den Code gemäss Ihren Erkenntnissen an, damit die Clean Code-Regeln erfüllt sind.

  In der Zeile "public String Output {" verlangt der Compliler ein anschliessendes "return". Mit einem "void konnte dies umgangen werden.

Erzeugen Sie anschliessend ein Buch-Objekt und interagieren Sie mit diesem, um sicherzustellen, dass die Klasse korrekt implementiert ist. Verwenden Sie den Objektinspektor um zu prüfen, wie sich die Datenfelder verändern und experimentieren Sie auch mit der Direkteingabe (Menü *Ansicht → Direkteingabe anzeigen*), um in Java“ Objekte zu erzeugen und mit ihnen zu interagieren.

Aufgabe 4

Forken Sie das Projekt: https://github.engineering.zhaw.ch/prog1-kurs/02_Praktikum-2_Konto.
Nutzen Sie BlueJ um die eigene Projektkopie auf Ihren Computer zu holen und zu bearbeiten.

Implementieren Sie dann eine Klasse, welche in einer Bankapplikation für die Modellierung eines Bankkontos dienen soll. Nachfolgend ist diese Klasse spezifiziert:

- Ein Konto hat einen Kontoinhaber, einen Kontostand (in Franken, ganzzahlig) und einen Zinssatz (in %, ganzzahlig).
- Ein Konto kann auf zwei Arten erzeugt werden:
  - Mit Angabe des Kontoinhabers. Der Kontostand soll initial 0 und der Zinssatz 2 sein.
  - Mit Angabe des Kontoinhabers und des Zinssatzes. der Kontostand soll initial 0 sein.

  Hinweis: Eine Klasse kann mehrere Konstruktoren haben, wenn sich diese durch die Parameterlisten eindeutig unterscheiden lassen.
- Der Zinssatz kann durch geeignete Methoden abgefragt und neu gesetzt werden.
- Es gibt je eine Methode, um Geld einzuzahlen und um einen Betrag abzuheben. Sie müssen nicht prüfen, ob die verwendeten Beträge sinnvoll sind oder ob der Kontostand negativ wird.
- Es gibt eine Methode, welche den effektiven Jahreszins basierend auf dem aktuellen Kontostand zurückgibt. Ist der Kontostand z.B. CHF 4000 und der Zinssatz ist 3%, dann wird 120 zurückgegeben. Kümmern Sie sich nicht um allfällige Präzisionsverluste und rechnen Sie „ganzzahlig“.
- Der Zustand eines Kontos kann auf der Kommandozeile ausgegeben werden. Die Formatierung soll dabei genau wie folgt aussehen:

```
Informationen zum Konto:
Kontoinhaber: Max Meier
Kontostand: CHF 4000
Zinssatz: 3%
```

Testen Sie die Klasse nach der Implementierung um sicherzustellen, dass sie korrekt funktioniert und alle Anforderungen erfüllt.

Beachten Sie bei dieser Aufgabe insbesondere folgende Punkte:

- Die Klasse selbst, die Konstruktoren und jede public Methode soll einen prägnanten Javadoc-Kommentar (/** ... */) erhalten. Weitere Kommentare sollten nicht nötig sein.
- Achten Sie auf die Organisation in der Klasse: Erst die Datenfelder, dann Konstruktoren, dann Methoden.
- Beachten sie die Clean Code Regeln. Wählen Sie entsprechend sinnvolle Namen für die Klasse, die Datenfelder, die Methoden und die Parameter.