Klassenentwurf I

Lernziele

- Sie können Programme im Umfang von einigen Klassen bezüglich Kohäsion, Codeduplizierung, Kopplung und Kapselung analysieren und verbessern.

Aufgabe 1 (auf Papier!)

Das nachfolgende Codebeispiel weisst eine geringe Kohäsion auf:

```
public class Kohaesion {

	private int zahl = 1;

	public int manipuliereZahl(int befehl, int zahl) {
		if (befehl == 0) {
			this.zahl = zahl;
		}
		if (befehl == 1) {
			this.zahl = this.zahl + zahl;
		}
		if (befehl == 2) {
			this.zahl = this.zahl - zahl;
		}
		return this.zahl;

	}
}
```

Woran erkennen Sie dies und wieso ist das schlecht?

Die Methode hat verschiedene Aufgaben. Dies sieht man daran, dass die Methode einen Paramter Befehl hat.

Jede Methode sollte genau einen Befehl haben. Man muss den richtigen Int wert nehemen pro Befehl.

Machen Sie es besser: Wie müsste die Klasse aussehen, damit sie eine hohe Kohäsion aufweist? Sie dürfen die Klasse inkl. Schnittstelle komplett neu schreiben.

```java
public class Kohaesion {

    6 usages
    private int zahl = 1;

    public void setZahl(int zahl) {
        this.zahl = zahl;
    }

    public int getZahl() {
        return zahl;
    }

    public void erhoeheZahl(int zahl){
        this.zahl = this.zahl + zahl;
    }

    public void senkeZahl(int zahl){
        this.zahl = this.zahl - zahl;
    }
}
```

Aufgabe 2

Forken Sie für diese Aufgabe das Projekt https://github.zhaw.ch/prog1-kurs/06_Praktikum-1_Zuul-schlecht
Nutzen Sie BlueJ um die eigene Projektkopie auf Ihren Computer zu holen und zu bearbeiten.

Machen Sie sich mit dem Programm und dem Quellcode vertraut. In der Klasse `Spiel` gibt es doppelten Code. Finden Sie diesen Code und schreiben Sie eine neue Methode, so dass der Code nicht mehr redundant vorhanden ist. Die Methode sollte folgende Signatur haben:

```
private void rauminfoAusgeben()
```

Hinweis: Beim gesuchten Code handelt es sich um Systemausgaben die in den Methoden `wechsleRaum` und `willkommenstextAusgeben` zu finden sind.

Aufgabe 3

Untersuchen Sie sich nun die Klasse `Raum`. Darin finden sich folgende Zeilen:

```
public String beschreibung;
public Raum nordausgang;
public Raum suedausgang;
public Raum ostausgang;
public Raum westausgang;
```

Bezüglich der Kopplung sowie der Kapselung ist das ein schlechter Klassenentwurf. Erklären Sie wieso dem so ist:

public ist schlecht, da andere Klassen direkt darauf zugreifen können und diese veränderen können. Eine Hashmap wäre auch besser, da es einfacher wäre diese zu ergänzen.

Benutzen Sie für die Räume nun statt der obigen Variante eine HashMap und setzen alle Datenfelder der Klasse auf `private`.

```
private String beschreibung;
private HashMap<String, Raum> ausgaenge;
```

Passen Sie anschliessend die ganze `Raum` Klasse entsprechend an. Unter anderem sollte die angepasste Klasse die folgenden beiden Methodensignaturen aufweisen:

```
public void setzeAusgaenge(String richtung, Raum raum)

public Raum gibAusgang(String richtung)
```

Passen Sie schliesslich noch die Klasse `Spiel` an die neue Situation an.

Aufgabe 4

In der Aufgabe 2 haben Sie die Methode `rauminfoAusgeben` in der Klasse `Spiel` erstellt. Wenn Sie sich diese Methode genauer ansehen, bereitet diese Klasse Informationen auf, die von der Klasse `Raum` verwaltet werden.

Ein besseres Design ist, wenn diese Informationen in der Klasse `Raum` aufbereitet werden und von der Klasse `Spiel` abgerufen werden. Schreiben Sie eine solche Methode in der Klasse `Raum` mit der folgenden Signatur.

```
/**
 * Liefere eine Beschreibung der Ausgänge dieses Raumes, bespielsweise
 * "Ausgänge: north west".
 *
 * @return eine Beschreibung der verfügbaren Ausgänge
 */
public String gibAusgaengeAlsString()
```

Erklären Sie, wieso dieses Design besser ist:

Aufgabe 5

In der letzten Aufgabe haben Sie die Methode `gibAusgaengeAlsString` in der Klasse Raum erstellt. Bezüglich Ausgabe verbleibt trotz dieser Änderung noch eine Kopplung, die sich entfernen lässt. Finden und entfernen Sie diese Kopplung. Dazu müssen Sie unter anderem die nachfolgende Methode zur Klasse `Raum` hinzufügen:

```
/**
 * Liefere eine lange Beschreibung dieses Raumes, in der Form
 *
 * Sie sind in der Küche
 * Ausgänge: north west
 *
 * @return eine lange Beschreibung dieses Raumes
 */
public String gibLangeBeschreibung()
```

Aufgabe 6

Ergänzen Sie das Objektdiagramm, dass alle Objekte Ihres Spiels zu dem Zeitpunkt zeigt, zu dem das Spiel gerade gestartet wurde.