Arbeitsblatt 04 Abstraction

1 Einleitung

In diesem Arbeitsblatt vertiefen in eine praktische Übung das erste und zweite Grundprinzip der objektorientierten Programmierung. Im ersten Grundprinzip haben Sie gelernt, dass Sie Code entsprechend Ihrer Zughörigkeit in separate Klassen verkapseln. Im zweiten Grundprinzip geht es nun darum die Komplexität für die Interaktion mit den erstellten Klassen und Objekten zu vereinfachen, indem wir abstrahieren.

Was bedeutet der Begriff abstrahieren oder das Subjekt Abstraktion überhaupt? Nach Wikipedia bedeutet Abstraktion: «*Abstraktion bezeichnet meist den induktiven Denkprozess des erforderlichen Weglassens von Einzelheiten und des Überführens auf etwas Allgemeineres oder Einfacheres.*»

In anderen Worten bedeutet Abstraktion eine vereinfachte auf die wichtigen Punkte reduzierte Ansicht einer Sache.

Was bedeutet Abstraktion im Hinblick auf die objektorientierte Softwareentwicklung?

In der objektorientierten Entwicklung wird mit Klassen und Objekten gearbeitet, die Abstraktion wird auf die Klassen bzw. Objekte angewendet. Wenn eine Klasse oder ein Objekt abstrahiert wird, wird also dessen Komplexität reduziert und nur die wichtigsten Punkte zur Interaktion bleiben für den Anwender sichtbar. Dies ist ein alltägliches Prinzip welches überall Anwendung findet. Beispielsweise müssen Sie nicht die komplexe Logik eines Autos verstehen und kennen, sondern es reicht aus die Steuerelemente innerhalb des Cockpits zu verstehen und anwenden zu können, damit Sie von Bern nach Zürich fahren können.

Erinnern Sie sich an die Employee-Klasse die wir in den ersten Blöcken des Moduls 404 erstellt haben:

```
package ch.csbe.modul404;


public class Employee {
   private int baseSalary;
   private int extraHours;
   private int balanceRate = 30;


   public void setBaseSalary(int baseSalary) {
      if (baseSalary <= 0)
         throw new IllegalArgumentException("BaseSalary cannot be 0 or less.");
      this.baseSalary = baseSalary;
   }


   public void setExtraHours(int extraHours) {
      if (extraHours < 0)
         throw new IllegalArgumentException("ExtraHours cannot be less than 0.");
```

```
        this.extraHours = extraHours;
    }

    public int getBaseSalary() {
        return baseSalary;
    }

    public int getExtraHours() {
        return extraHours;
    }

    public int calculateWage() {
        return baseSalary + (extraHours * balanceRate);
    }
}
```

Hier wurde beispielsweise bereits Abstraction angewendet, weil nur Methoden nach aussen sichtbar sind welche auch effektiv verwendet werden. Bspw. können die Extra-Hours pro Mitarbeiter abgefragt werden, jedoch nicht die balanceRate, da diese, in diesem Beispiel, für alle Mitarbeiter gleich ist. Aus dem Grund wurden für diese Eigenschaft keine Setter- und Getter-Methode implementiert, da diese sonst nur unnötige Schnittstellen nach aussen bilden würde.

Aufgabe 1) Kreisberechnungen

Gegeben sei folgende Kreis-Klasse mit folgenden vier Eigenschaften:

- Radius
- Durchmesser
- Umfang
- Fläche

Sie sollen aus dieser Klasse heraus einzelne Kreisobjekte erstellen können.

Implementieren Sie eine Kreis-Klasse, welche die minimale benötigte Anzahl an Methoden und Eigenschaften nach aussen preisgibt. Jeder Kreiss soll zur Erstellzeit einen Radius und demzufolge auch einen Durchmesser besitzen. Verwenden Sie für die Berechnung von Umfang und Fläche für Pi, die Math-Klasse.

Rufen Sie mit dem Konstruktor die beiden privaten Methoden auf, welche den Umfang und die Fläche berechnen.

Zur Erinnerung:

Kreisfläche = r * r * pi
Kreisumfang = d * pi

Aufgabe 2) CustomTextField

Als zweite Übung erstellen Sie eine eigene TextField-Klasse. Diese soll folgende Eigenschaften aufweisen:

- Text
- TextLength
- IsEmpty
- HasNumber
- NumberCount
- HasLetter
- LetterCount

Die Klasse soll folgende öffentliche Methoden aufweisen:

- GetText()
- GetTextLength()
- GetIsEmpty()
- GetHasNumber()
- GetNumberCount()
- GetHasLetter()
- GetLetterCount
- SetText

Nachfolgende Implementierungsdetails sollen in privaten Methoden abgehandelt werden:

- Leerzeichen am Anfang und am Ende des Inhalts sollen entfernt werden – verwenden Sie hierfür die trim()-Methode der String Klasse.
- Nachfolgende Sonderzeichen - , . sollen durch einen Leerschlag ersetzt werden – verwenden Sie hierfür die replace() Methode der String Klasse.

Melden Sie sich bei der Lehrperson, sobald Sie die Klasse fertiggestellt haben.

Aufgabe 3) PokemonArena

Bearbeiten Sie die Aufgabe 12 PokemonArena weiter und wenden Sie nun zusätzlich zum Prinzip der Encapsulation auch das Prinzip der Abstraction an.

Melden Sie sich bei der Lehrperson, sobald Sie fertig sind.