UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA
OBERÖSTERREICH

Übungen zu **Objektorientierte Programmierung**
SS 2023
FH-Prof. Dipl.-Ing. Dr. Marc Kurz
Alexander Palmanshofer, MSc

Letzter Abgabetermin: **Dienstag 25.04.2023, 23:55 Uhr**
Abgabe: https://elearning.fh-hagenberg.at/

Polymorphismus & Dynamische Bindung

Aufgabe 1 Modifikation des Binärbaumes — 9 Punkte

Um dem Binärbaum aus Übung 3 nicht nur Integer-Elemente verwalten zu können, ist die Implementierung des Binärbaumes dahingehend zu erweitern, dass als Datenmember der Klasse **BinaryTreeNode** der Typ **java.lang.Comparable** verwendet wird.

```
public class BinarySearchTree<T extends Comparable<T>> {
  protected class BinaryTreeNode {
    public BinaryTreeNode left;
    public BinaryTreeNode right;
    public T data;

    public BinaryTreeNode(T data) {
      this.data = data;
      this.left = null;
      this.right = null;
    }
  }

  ...
}
```

Die Klasse **Comparable** ist ein Interface, welche als einzige Methode die Vergleichsoperation **int compareTo(T other)** anbietet. Implementiert eine Klasse das **Comparable**-Interface, dann muss über diese Methode eine spezialisierte Vergleichsoperation bereitgestellt werden.

Beispiel: Sie haben eine Klasse **Car**. Ein **Car** ist dann „größer“ als ein anderes, wenn seine Länge größer ist. Daher könnte die Implementierung etwa so aussehen:

```
public class Car implements Comparable<Car> {
  private double length; // relevant for comparison
  private double width; // not relevant for comp.
  private double engineDisplacement; // not relevant for comp.

  @Override
  public int compareTo(Car other) {
    if (this.length > other.length) {
      return 1;
    } else if (this.length == other.length) {
      return 0;
    } else {
      return -1;
    }
  }
}
```

In der Implementierung der Klasse **BinarySearchTree** müssen Sie dann einerseits die Schnittstelle all jener Methoden anpassen, die vorher einen Integer als Wertparameter übernommen haben.

```
public boolean insert (int elem) { ... }
```

würde somit in der neuen Version als

```
public boolean insert (T elem) { ... }
```

geschrieben werden.

Außerdem müssen sämtliche Vergleiche anpasst werden, da die Vergleichsoperatoren „<“, „>“, „==“, etc. für Objekte nicht verfügbar sind.
Aus `if (data > tempRoot.data) { … }` wird dementsprechend

```
if (data.compareTo(tempRoot.data) > 0) { … }
```

Aufgabe 2 Kostenstruktur eines Sportvereins — 15 Punkte

Ein Sportverein möchte sich ein klares Bild von den Einnahmen und Ausgaben seiner Mitglieder in einem Jahr machen. Mitglieder können dabei entweder physische Personen oder wiederum Vereine (d.h. Zweigstellen des Vereins) sein. Die Mitglieder (abstrakte Klasse **AbstractMember**) haben einen eindeutigen Namen, d.h. es können nicht mehrere Mitglieder mit dem gleichen Namen einem bestimmten Verein angehören.

Es existieren folgende Arten von physischen Personen als Mitglieder:

- **Unterstützende Mitglieder** (Klasse **SupportingMember**): Diese Mitglieder bezahlen einen Jahresbeitrag von €100,- und verursachen bei Vereinsfesten Ausgaben von durchschnittlich €15,- pro Jahr.
- **Aktive Mitglieder** (Klasse **ActiveMember**): Diese Mitglieder besitzen einen ganzzahligen Aktivitätsgrad im Bereich von 0 bis 10, und gliedern sich in:
  - Spitzensportler (Klasse **TopAthlete**): Monatlicher Beitrag: €10,-, Ausgaben in € pro Monat: Aktivitätsgrad * 5
  - Amateure (Klasse **AmateurAthlete**): Monatlicher Beitrag: €25,- Ausgaben in € pro Monat: Aktivitätsgrad * 2,5
  - Trainer (Klasse **Trainer**): Monatlicher Beitrag: €10,- Ausgaben in € pro Monat: Aktivitätsgrad * 40
- **Vorstandsmitglieder** (Klasse **ChairMember**): Vorstandsmitglieder haben einen ganzzahligen Kompetenzwert im Bereich von 0 und 10. Ein Vorstandsmitglied erzeugt durch das Lukrieren von Sponsorengeldern und Förderungen Jahreseinnahmen von Kompetenz * 100 € und verursacht Ausgaben, indem es 20% Provision für lukrierte Einnahmen erhält.
- **Ehrenmitglieder** (Klasse **HonoraryMember**): Ehrenmitglieder haben sich durch besondere Verdienste hervorgetan, und müssen daher keinen Jahresbeitrag mehr bezahlen. Bei Vereinsfesten verursachen sie Ausgaben von durchschnittlich €20,-/Jahr.

Weiters existieren Sektionen, die ihrerseits wiederum als Vereine organisiert sein können (Composite Design Pattern[1]) und ihrerseits Mitglieder eines Vereins sein können. Beispiel: die Sportunion Hintertupfing ist vom Typ **Section** und besitzt ihrerseits Sektionen, z.B. Fußball, Tennis und Skifahren.

[1] siehe http://en.wikipedia.org/wiki/Composite_pattern

- Sektionen (Klasse **Section**): Die Klasse **Section** speichert eine Menge von Mitgliedern in Form eines Binärbaumes, und kann die gesamten Einnahmen, Ausgaben und den Überschuss berechnen. Stellen sie für **Section** folgende Methoden zur Verfügung:
  - **boolean addMember(AbstractMember m)**
  - **boolean removeMember(AbstractMember m)**
  - **boolean isMember(AbstractMember m)**

Die Klasse **AbstractMember** sollte folgende Schnittstelle besitzen:

- **double getIncome()**: berechnet die gesamten Einnahmen, die der Verein durch dieses Mitglied pro Jahr erzielt.
- **double getCosts()**: berechnet die gesamten Ausgaben, die dieses Mitglied pro Jahr verursacht.
- **double getSurplus()**: berechnet den finanziellen Überschuss, den dieses Mitglied dem Verein pro Jahr bringt.
- **String toString(boolean ascending)**: gibt ordentlich formatiert und strukturiert (mit Einrückungen im Fall einer Sektion) den Namen sowie Einnahmen, Ausgaben und Überschuss des Mitglieds (der Mitglieder im Fall einer Sektion) zurück. Wenn **ascending** true ist, soll die Ausgabe aufsteigend sortiert, ansonsten absteigend sortiert erfolgen. Die Default-Methode **String toString()** soll eine Ausgabe in aufsteigender Reihenfolge erzeugen.