Dokumentation

Pflichtpräsentation Moritz Glaser — Fechtclub-Mitglieder-Management

**Gliederung**:

**1. Einleitung**

**2. Konzept der Anwendung**

2.1 Anforderungen an die Anwendung

2.2 Funktionsumfang der Anwendung

**3. Darstellungen der Grundlegenden Algorithmen**

3.1 Binary Search Tree (BST)

3.2 Sortieralgorithmus: Selection Sort

3.3 Validierung der Eingaben

**4. UML-Diagramm**

**5. Programmquellcode**

**6. Quellen und Literaturverzeichnis**

**7. Selbstständigkeitserklärung**

1. **Einleitung**

In dieser Dokumentation wird die Entwicklung meiner Java-Anwendung zur Verwaltung von Mitgliedsdaten für einen Fechtclub beschrieben. Die Anwendung bietet Funktionen zur Speicherung, Sortierung und gezielten Suche nach Mitgliederdaten. Um diese Funktionen zu ermöglichen, wird ein Binärer Suchbaum (BST) verwendet, ergänzt durch ein einfaches Sortierverfahren.

1. **Konzept der Anwendung**
   1. **Anforderungen an die Anwendung**

Implementieren einer Anwendung in Java, die Kundendaten verwaltet, diese nach bestimmten Datensätzen sortieren und nach einzelnen Kriterien durchsuchen kann.

Das Programm beinhaltet einen binären Suchbaum und ein einfaches Sortierverfahren.

* 1. **Funktionsumfang der Anwendung**

Die Anwendung kann:

• Mitgliedsdaten (Name, Beitrittsdatum, Vertragsart, Beitrag) speichern (bis das Programm beendet wird)

• Mitgliedsdaten effizient verwalten (Erstellung, Bearbeitung, Löschung)

• Mitglieder suchen

• Daten sortiert ausgeben

• Mitgliedern anhand der ID und des Namen suchen

• Ausgabe sortierter Mitgliedsdaten

• Validierung einiger Eingaben, z.B. korrektes Datum (LocalDate), Name etc.

Ein binärer Suchbaum ist integriert, genauso ein einfaches Sortierverfahren in Form von Selection-Sort.

1. **Darstellungen der Grundlegenden Algorithmen**
   1. **Binary Search Tree (BST):**Der BST kann Mitglieder anhand der ID Einfügen, Suchen und löschen. Dies erfolgt durch die Speicherung in einer hierarchischen Datenstruktur.

**Operationen**

**Einfügen:**

Vergleicht mit der Wurzel:

Kleiner → Geht in den linken Teilbaum.

Größer → Geht in den rechten Teilbaum.

Fügt das Element als Blatt ein.

**Suchen:**

Vergleicht mit der Wurzel:

Gefunden → Rückgabe.

Kleiner → Suche im linken Teilbaum.

Größer → Suche im rechten Teilbaum.

**Löschen:**

Fall 1: Knoten ist ein Blatt → Direkt löschen.

Fall 2: Knoten hat ein Kind → Ersetzt den Knoten durch das Kind.

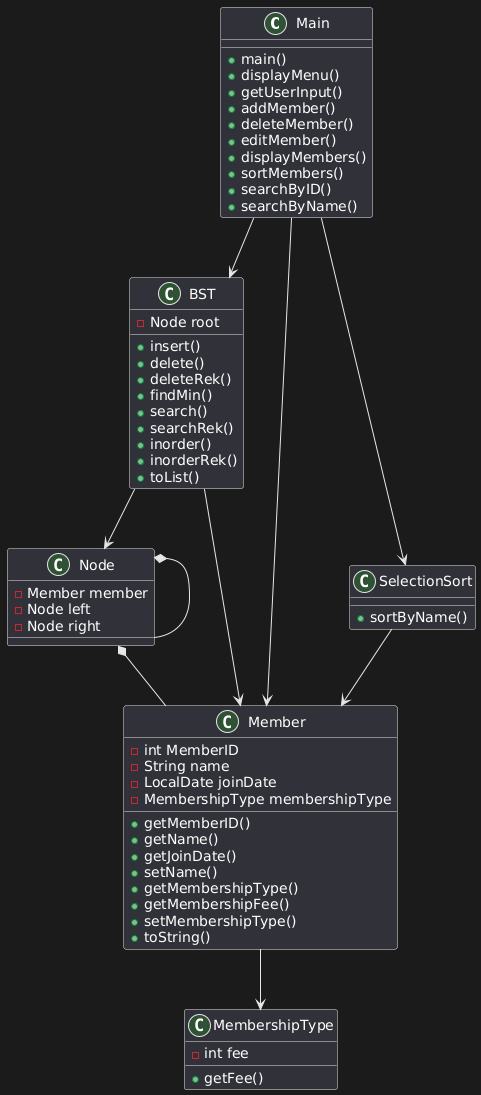
Fall 3: Knoten hat zwei Kinder → Ersetzt den Knoten durch das kleinste Element im rechten Teilbaum und löscht dieses dort.

**Traversierung:** Inorder (Links → Wurzel → Rechts)

Gibt die Elemente in aufsteigender Reihenfolge aus.

* 1. **Sortieralgorithmus: Selection Sort:** Der Algorithmus wird verwendet, um die Mitglieder basierend auf der ID oder dem Namen zu sortieren, bevor sie angezeigt werden. Dabei wird das kleinste Elemente der Liste gesucht und mit dem ersten Element getauscht. Dies wird so lange im unsortierten Teil der Liste wiederholt bis alles an der richtigen Position steht.
  2. **Validierung und LocalDate:** Bei der Validierung wird geprüft, ob die eingegebenen Daten korrekt sind (z. B. ein gültiges Beitrittsdatum). Das Beitrittsdatum wird im LocalDate-Format gespeichert, um für eine gleichbleibende Handhabung zu sorgen. Außerdem wird überprüft ob es überhaupt eine Eingabe gibt (Die Eingabe darf nicht leer sein)

1. **UML-Diagramm**



Leider habe ich die Website vergessen, auf der wir unsere UML-Diagramme im Unterricht erstellt haben, diese habe ich als Ersatz gefunden und genutzt (siehe Quellen). Die Darstellung von MembershipType als Enum und Node als Innere Klasse war mir auch nicht wirklich möglich.

1. **Programmquellcode**

Siehe Projektordner/ZIP-Datei

1. **Quellen und Literaturverzeichnis**

Bücher:

• Java: The Complete Reference. (Existiert als PDF auf Github: Nitin96Bisht/Java-Books)

• Data Structures and Algorithms made easy in Java. (Existiert als PDF auf Github: Nitin96Bisht/Java-Books)

Online-Ressourcen:

• plantuml.com (UML Diagramm

• Oracle Java Dokumentation: <https://docs.oracle.com> (Zugriff: 10.11.2024)

• GeeksforGeeks: Einführung in Binary Search Trees, weitere Tutorials <https://geeksforgeeks.org> (letzter Zugriff: 22.11.2024)

• ChatGPT zum besseren Verständnis der Programmanforderungen, zur Fehlerbehebung, und für eine beispielhafte Gliederung der Dokumentation (Was muss mit rein?)

Weitere Änderungen durch ChatGPT:   
Nach einfügen der fertigen Main Class:

1. **getUserInput-Methode:** Wiederverwendbare Methode zur Eingabeprüfung, um redundanten Code zu vermeiden.
2. **Kommentare:** Bessere Struktur und Klarheit in den Kommentaren.
3. **Fehlerbehandlung:** Robustere Eingabeverarbeitung für Benutzeroptionen.
4. **Lesbarkeit:** Verbesserte Einrückung und konsistenter Stil.

Einrückung und Stil in allen Klassen verbessern lassen, um den Code übersichtlicher zu gestalten.

„Ich versichere, dass die Präsentation von mir selbstständig erarbeitet wurde und ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Diejenigen Teile der Präsentation, die anderen Werken im Wortlaut oder dem Sinn nach wissentlich entnommen wurden, sind als solche kenntlich gemacht.“

Habichtswald, 22.11.24 Moritz Glaser

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) Unterschrift