ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Science

Programmierung 2 - Sommersemester 2024

Prof. Dr. Markus Esch

Übung Nr. 4
Abgabe KW 22

1. Aufgabe (Codeopolis)

   Die Klassen `Depot` und `Silo` verwenden derzeit ein Array, um die `Silo` bzw. `Harvest`-Objekte zu speichern. Die Verwendung eines Arrays hat den Nachteil, dass es nicht einfach erweitert werden kann. Deshalb soll das Array durch eine dynamische Datenstruktur ersetzt werden.

   In Informatik 1 haben Sie eine einfach verkettete Liste kennen gelernt. Implementieren Sie eine solche verkettete Liste als generische Klasse und ersetzen Sie die Arrays in den oben genannten Klassen.

   Gehen Sie bei der Implementierung wie folgt vor:

   - Deklarieren einer generischen Klasse `LinkedList`.
   - Implementieren Sie eine Klasse `Node`, die einen Knoten in der Liste repräsentiert, als innere Klasse. Die Klasse Node benötigt ein Feld um ein Datenobjekt zu speichern und eine Referenz auf das nächste Element in der Liste. Der Typ des Datenobjekts muss generisch sein.
   - Die Klasse `LinkedList` benötigt eine Referenz auf den ersten Knoten in der Liste und die folgenden Methoden:
     - Standardkonstruktor, der eine leere Liste initialisiert.
     - `addLast`: Fügt ein Element am Ende der Liste hinzu.
     - `removeFirst`: Entfernt das erste Element aus der Liste und gibt es zurück.
     - `isEmpty`: Prüft, ob die Liste leer ist.
     - `size`: Gibt die Anzahl der Elemente in der Liste zurück.
     - `get)`: Gibt das Element am angegebenen Index zurück. x
     - `set`: Ersetzt das Element an einem bestimmten Index und gibt das alte Element zurück.
     - `clear`: Entfernt alle Elemente aus der Liste.
     - `remove`: Entfernt das Element am angegebenen Index und gibt es zurück.
     - `iterator`: Gibt einen Iterator für die Liste zurück. Implementieren Sie den Iterator als innere Klasse. Auch der Iterator muss generisch sein. Sie können sich bei der Implementierung des Iterators an dem in Aufgabe 3 implementierten Iterator orientieren.
   - Ersetzen Sie die Arrays in den Klassen `Depot` und `Silo` durch die Klasse `LinkedList`. Verwenden Sie dabei, wenn immer möglich, den Iterator, um über die Elemente der Liste zu iterieren.

2. Aufgabe (Codeopolis)

   Erweitern Sie die Klasse `LinkedList` um eine Methode `sort`, die die Elemente der Liste mit Hilfe des Bubblesort-Algorithmus sortiert (Wenn Sie möchten, können Sie auch einen komplexeren Sortieralgorithmus implementieren).

   Um die Elemente der Liste sortieren zu können, müssen Sie sicherstellen, dass sie vergleichbar sind. Verwenden Sie dazu das Interface `java.lang.Comparable<T>`[1], welches wir in der Vorlesung kennengelernt haben. Stellen Sie sicher, dass die Elemente in der Liste dieses Interface implementieren.

   Um die Klassen `Silo` und `Harvest` weiterhin als Elemente in der Liste verwenden zu können, müssen diese das Interface implementieren. Vergleichen Sie `Silo`-Objekte anhand des Füllstands und `Harvest`-Objekte anhand des Erntejahres.

   Passen Sie anschließend die Methode `Depot.toString()` so an, dass die Silos des Depots sortiert nach dem Füllstand ausgegeben werden.

3. Aufgabe
   Entwickeln Sie eine Klasse `Matrix`, die mathematische Operationen auf Matrizen durchführen kann. Die Klasse soll flexibel unterschiedliche numerische Typen handhaben können.

   Klassenstruktur

   - Entwickeln Sie eine Klasse, die eine zweidimensionale Matrix speichert. Die Klasse sollte fähig sein, Operationen auf Matrizen beliebiger numerischer Typen durchzuführen.
   - Der Konstruktor nimmt die Dimensionen der Matrix (Zeilen und Spalten) entgegen und initialisiert die Matrix.
   - Implementieren Sie Methoden, um Elemente in der Matrix zu setzen und zu erhalten.

   Matrixoperationen

   - Implementieren Sie eine Methode zur Addition zweier Matrizen. Das Ergebnis der Addition soll als `Matrix`-Objekt zurückgegeben werden. Die Werte der ursprünglichen Matrix sollen nicht verändert werden. Es soll möglich sein, zwei Matrizen unterschiedlichen numerischen Typs zu addieren.
   - Fügen Sie eine Methode hinzu, die eine Skalarmultiplikation jeder Zelle der Matrix mit einem numerischen Wert durchführt. Das Ergebnis soll als `Matrix`-Objekt zurückgegeben werden. Die Werte der ursprünglichen Matrix sollen nicht verändert werden.
   - Schreiben Sie eine Methode, die die Matrix auf eine lesbare Weise ausgibt.

   Testen Ihrer Klasse

   - Erstellen Sie eine `main`-Methode, um Ihre Matrixklasse zu testen. Verwenden Sie dabei verschiedene numerische Typen und Operationen, um die Funktionalität zu demonstrieren.

   Überlegungen

   - Überlegen Sie, wie Sie generische Typen verwenden können, um ein hohes Maß an Flexibilität bei gleichzeitiger Typsicherheit zu erreichen.
   - Überlegen Sie, wie Operationen, die numerische Berechnungen beinhalten, typsicher implementiert werden können.

[1] https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/Comparable.html

Lernziele:

Nach der Bearbeitung des Übungsblattes sollten Sie ...

- ... in der Lage sein eine generische Klasse zu implementieren.
- ... generische Klassen zu instanziieren und zu nutzen.
- ... Wildcards und Bounded Typ Parameter erklären und nutzen können.
- ... die Vorteile generische Implementierungen hinsichtlich der Typ-Sicherheit erklären können.
- ... Beschränkungen der Generics erkennen und erklären können.
- ... in der Lage sein, auf der Grundlage gegebener Anforderungen zu entscheiden, welche der von den Generics bereitgestellten Implementierungsmöglichkeiten für die Umsetzung am besten geeignet ist.

Zusatzaufgabe

Die folgenden Aufgaben sind Zusatzaufgaben und nicht Teil des Peer Reviews! Sie können die Aufgaben freiwillig bearbeiten, um die Inhalte weiter zu üben. Selbstverständlich können Sie auch Unterstützung und Beratung zu den Zusatzaufgaben erhalten.

1. Aufgabe (Codeopolis)

   Implementieren Sie JUnit-Tests für die Klasse `LinkedList`.