Übungsaufgabe Strategie-Muster

Im Folgenden soll zur Demonstration des STRATEGIE-Musters ein Algorithmus zum Filtern dienen: Aus einer Liste von Werten sollen einige Werte nach speziellen Kriterien herausgefiltert werden. Die hier gezeigte Variante beschränkt sich auf das Filtern von int-Werten. Weil Generics nicht für primitive Typen anwendbar sind, wird hier zur Speicherung eine Liste von Integer-Objekten verwendet.

Erste, intuitive Lösung ohne STRATEGIE-Muster Es soll eine Menge von Zahlen gefiltert werden, wobei der erlaubte Wertebereich durch zwei int-Werte entweder in Form eines geschlossenen oder offenen Intervalls angegeben werden kann. Die Art der Filterung wird mithilfe einer enum-Aufzählung FilterType mit den Werten OPEN_INTERVAL und CLOSED_INTERVAL bestimmt. Innerhalb einer Schleife werden die Eingabewerte geprüft und gegebenenfalls in die Ergebnismenge aufgenommen. Die folgende erste Implementierung setzt diese Anforderungen funktional korrekt um:

```
enum FilterType { OPEN_INTERVAL, CLOSED_INTERVAL }
// ..
public static List<Integer> filterAll(final List<Integer> inputs, final FilterType filterStrategy,
                                      final int lowerBound, final int upperBound)
{
   final List<Integer> filteredList = new LinkedList<>();

   if (filterStrategy == FilterType.CLOSED_INTERVAL)
   {
      for (final Integer value : inputs)
      {
         if (value >= lowerBound && value <= upperBound)
            filteredList.add(value);
      }
   }
   if (filterStrategy == FilterType.OPEN_INTERVAL)
   {
      for (final Integer value : inputs)
      {
         if (value > lowerBound && value < upperBound)
            filteredList.add(value);
      }
   }
   return resultSet;
}

public static void main(final String[] args)
{
   final List<Integer> inputs = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9);

   System.out.println("Filtering values for Intervall 2-7");
   System.out.println("Using CloseInterval [2,7] " + filterAll(inputs, FilterType.CLOSED_INTERVAL,
                                                               2, 7));

   System.out.println("Using OpenInterval ]2,7[ " + filterAll(inputs, FilterType.OPEN_INTERVAL,
                                                               2, 7));
}
```

Führen wir das Programm aus, so erhalten wir erwartungsgemäß folgende Ausgabe:

```
Filtering values for Intervall 2-7
Using CloseInterval [2,7] [2, 3, 4, 5, 6, 7]
Using OpenInterval ]2,7[ [3, 4, 5, 6]
```

Funktional können wir also zufrieden sein. Was ist aber zum Design zu sagen? Wie können wir es nach einer Analyse mithilfe des Strategy-Musterrs korrigieren? Eine kurze Analyse deckt folgende Probleme auf:

1. Der Sourcecode zum Filtern für geschlossenes bzw. offenes Intervall ist fast 1:1 dupliziert.
2. Jede weitere Filterstrategie erfordert Erweiterungen in der filterAll()-Methode.
Dadurch reduziert sich schnell die Lesbarkeit.

3. Werden weitere Auswahlkriterien gewünscht, etwa halboffene Intervalle, so muss die ursprüngliche Klasse um if-Anweisungen zur Auswahl der entsprechenden Strategie und deren Realisierung erweitert werden. Zudem müssen neue Aufzählungswerte zur Unterscheidung definiert werden.
4. Die realisierte Filterung setzt jeweils die Angabe von zwei Grenzwerten voraus. Wollte man eine vollständig andersartige Auswahlstrategie anwenden, so wäre dies mit dieser Implementierung extrem aufwendig, vielleicht sogar unmöglich: Eine Filterung auf eine Menge erlaubter Werte ließe sich so nicht realisieren.

Aufgaben:

a) Gegen welches Design Principle verstösst der Code?
b) Schreibe den Code so um, dass er das Strategy-Pattern nutzt.
c) Erstelle einen Ungerade-Filter / isInRange
d) Erstelle einen Not-Filter
e) Erstelle einen AND / OR-Filter
f) Welche Muster nutzt man dazu?