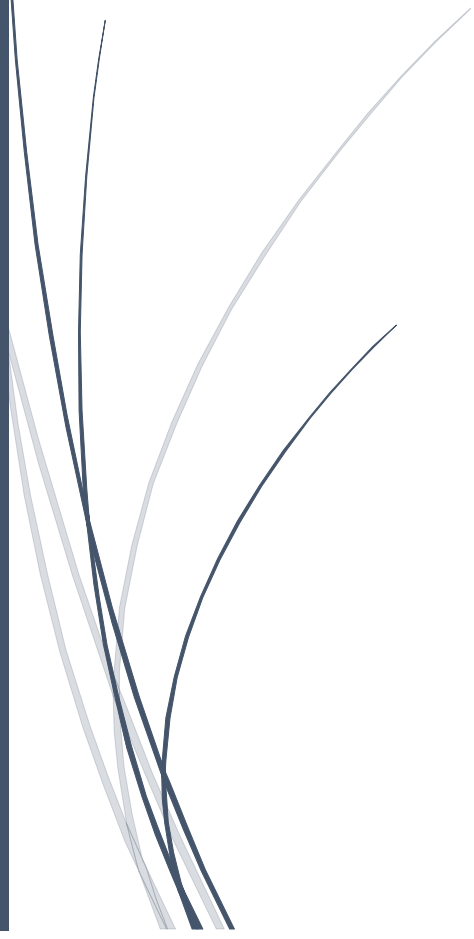
12.11.2014

Calculator

S03 – Strategy-Pattern

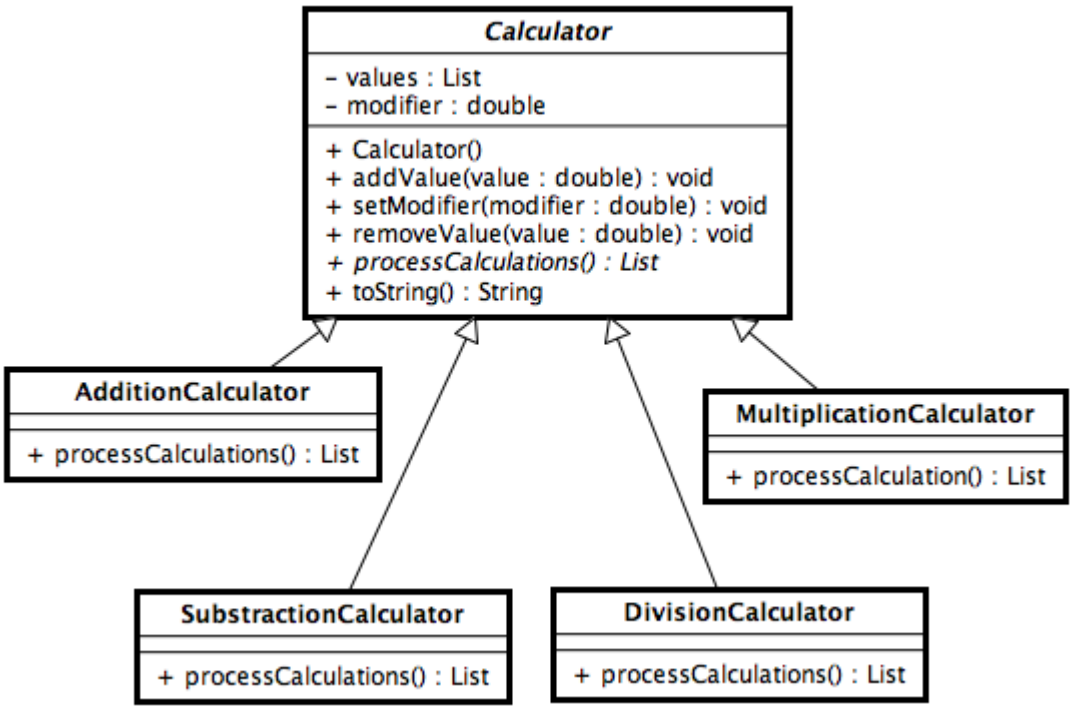


Martin Kritzl
4AHIT

Inhaltsverzeichnis

Aufgabenstellung

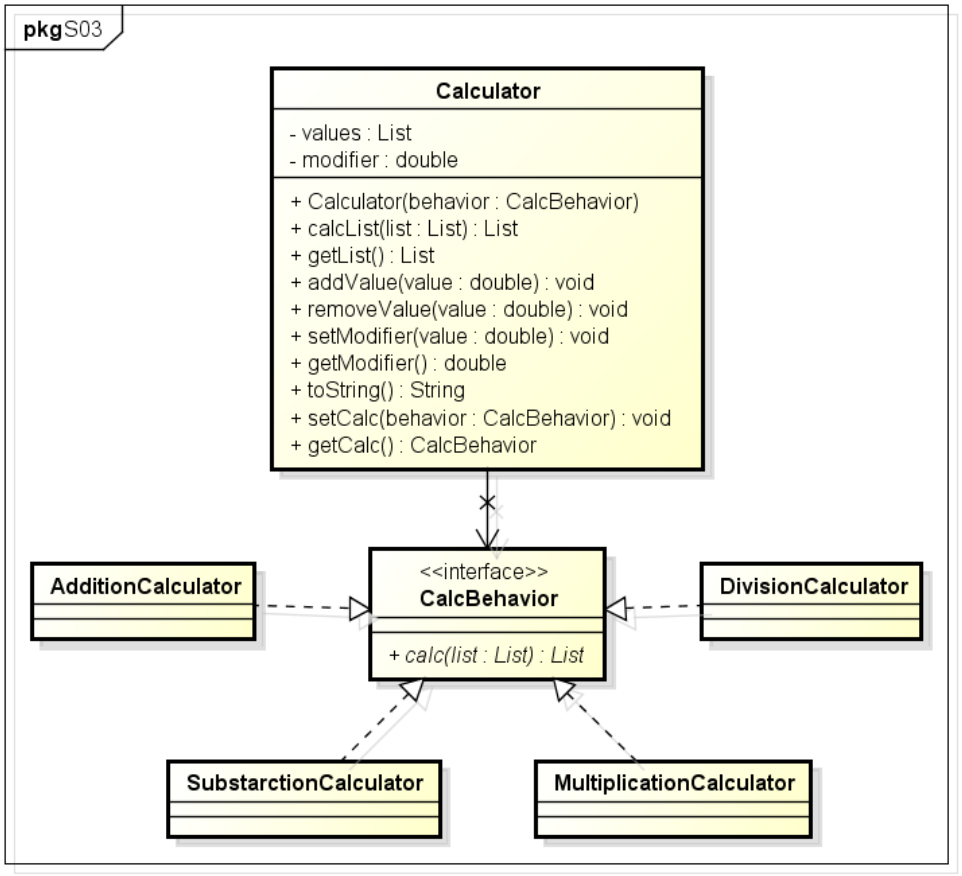
Die abstrakte Klasse Calculator hat die Aufgabe, Werte aus einer Liste mit einem modifier zu verändern und das Ergebnis als neue Liste zurück zu geben. Dazu dient die abstrakte Methode processCalculations, die in den konkreten Subklassen so überschrieben wurde, dass sie je nach Klasse die Werte aus der Liste mit dem modifier addiert, subtrahiert, multipliziert oder dividiert.



Designüberlegung

Benötigte Klassen:

- Calculator
  - Verwalten der Liste(hinzufügen/löschen) und Modifier
  - Aufrufen der Berechnung
- CalcBehavior
  - Setzt eine Methode calc(List<Double>, double) voraus
- AdditionCalculator
- SubstractionCalculator
- MultiplicationCalculator
- DivisionCalculator
- Testklassen
- Test-Suite



Aufwandabschätzung

Schätzung

Design	20 min
Calculator	30 min
Addition, substraction, multiplication, division	20 min
Testen	30 min
Protokoll	20 min
Gesamt	120 min = 2 Stunden

Tatsächlich

Zusammen: 3 Stunden

Arbeitsdurchführung

- Erstellen des Interfaces
- Erstellen des Calculators
  - Hinzufügen von Zahlen
  - Löschen von Zahlen
  - Setzen und auslesen des Modifiers
  - Wiedergeben eines repräsentierenden Strings
  - Setzen und auslesen der verwendeten Berechnungsart(z.B. Addition)
  - Aufrufen der Berechnung
- Erstellen der Berechnungsarten
  - AdditionCalculator
  - SubstractionCalculator
  - MultiplicationCalculator
  - DivisionCalculator
- Erstellen der Testklassen für jede Klasse
- Eine Test-Suite erstellen
- Die Test-Suite als Main im runnable-jar eintragen

Leasons Learned

- Durch diese lose Kopplung der Rechenarten ist es ganz einfach möglich weitere hinzuzufügen.
- Durch die Interfaces hat die Konstruktion leicht ersichtliche Zusammenhänge
- Auch die Testung solcher Klassen wird damit erleichtert
- Code kommt nicht doppelt vor
- Die Vererbungshierarchie bleibt frei für andere Anforderungen