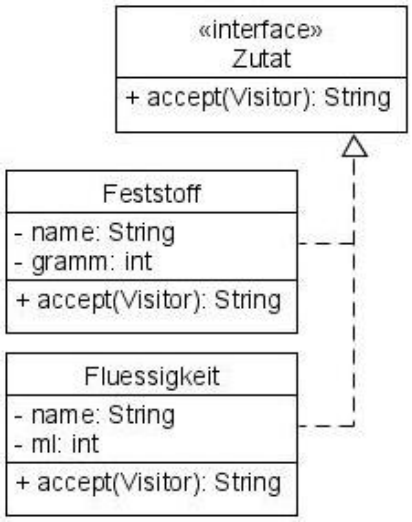
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Objektorientierte Analyse und Design

Sommersemester 2022
9. Aufgabenblatt

Aufgabe 25 (1 Punkt)

Gegeben seien die Klassen auf der rechten Seite aus dem Projekt ooadAufgabeVisitor, die bereits für die Nutzung eines Visitors vorbereitet sind und nicht verändert werden dürfen. Schreiben Sie zu den Klassen zwei Visitors, die die Objekte einmal mit einer Angabe im klassischen SI-Maßsystem (ml, g) und einmal im nordamerikanischen Maßsystem mit (fl oz, oz) ausgeben. Ergänzen Sie unter Nutzung Ihrer Visitor das Hauptprogramm main.Main so, dass die folgende Ausgabe generiert wird.




```
Mehl: 400 g
Salz: 200 g
Wasser: 500 ml
Oel: 45 ml
Zitronensaeure: 40 g
Mehl: 14.1095848 oz
Salz: 7.0547924 oz
Wasser: 16.9067425 fl oz
Oel: 1.5216068249999999 fl oz
Zitronensaeure: 1.41095848 oz
```

Aufgabe 26 (8 Punkte)

In der Veranstaltung wurde anhand eines einfachen Taschenrechners der Ansatz des Command-Patterns erklärt. Dies wird in dieser Aufgabe weiter untersucht und eine Möglichkeit zur Realisierung einer Undo-Funktionalität entwickelt. Ausgangspunkt ist das Programm aus der Vorlesung, das von der Veranstaltungsseite im Projekt ooadAufgabeCommand geladen werden kann. Die Klasse Rechner soll nicht verändert werden.

a) Das Programm enthält in der Klasse Dialog die Zeile

```
System.out.println("(0) Programm beenden");
```

Man kann diese Zeile einfach löschen, indem man eine weitere Realisierung eines Command-Patterns ergänzt, von der ein Objekt in die Position 0 des HashMaps eingefügt wird.

b) Ergänzen Sie ein Kommando Reset mit dem der Rechner zurückgesetzt werden kann, also alle Werte wieder auf 0 gesetzt werden. Schieben Sie dies als neuen Menü-Punkt in die Auswahl ein.

```
(1) Rechner zuruecksetzen
```

c) Ergänzen Sie mit dem in der Veranstaltung als zweites vorgestellten Ansatz eine Möglichkeit, die letzten Schritte rückgängig zu machen. Der Menü-Punkt dazu soll wie folgt aussehen und kann von Hand (ohne Nutzung der HashMap) in das Programm eingebaut werden. Die Ausgabe muss in der Methode ausgabe() erfolgen.

```
(98) letzte Aktion rueckgaengig (undo)
```

Ändern Sie dazu das Command-Interface so ab, dass die Ausführung von execute() ein Undo-Objekt liefert, mit dem die ausgeführte Aktion rückgängig gemacht werden kann.

```
public Command execute();
```

Das Undo-Objekt hat selbst eine execute()-Methode, die dann genutzt wird, wenn eine Aktion rückgängig gemacht werden soll.

d) Erweitern Sie das Programm so, dass man nach Undo-Schritten die Möglichkeit hat, auch alle diese wieder schrittweise rückgängig zu machen (Redo). Ergänzen Sie dazu den Dialog manuell um den folgenden Punkt.

```
(99) rueckgaengig rueckgaengig machen (redo)
```

Macht man nach einem Undo-Schritt einen normalen Schritt, werden alle Redo-Schritte natürlich verworfen, da diese ohne zuletzt ausgeführtes Undo keinen Sinn mehr haben.

e) Erstellen Sie mit dem Sequencediagrammer ein Diagramm aus dem laufenden Betrieb, beim ein undo und ein redo sinnvoll ausgeführt werden.

Die folgende Programmnutzung kann die geforderte Funktionalität weiter veranschaulichen. Beachten Sie, dass am Ende einer Aktion immer zuerst der Wert im Speicher und dann der aktuelle Wert des Rechners ausgegeben werden. Nutzen Sie die Klasse main.SystemTest zum Testen Ihres Programms.

```
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
2
Wert eingeben: 10
Speicher: 0  Wert: 10
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
2
Wert eingeben: 20
Speicher: 0  Wert: 30
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
2
Wert eingeben: 40
Speicher: 0  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
4
Speicher: 70  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
5
Speicher: 70  Wert: 140
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
98
Speicher: 70  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
(99) rueckgaengig rueckgaengig machen
(redo)
98
Speicher: 0  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
```

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -


```
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
(99) rueckgaengig rueckgaengig machen
(redo)
98
Speicher: 0  Wert: 30
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
(99) rueckgaengig rueckgaengig machen
(redo)
99
Speicher: 0  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
(99) rueckgaengig rueckgaengig machen
(redo)
99
Speicher: 70  Wert: 70
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
(99) rueckgaengig rueckgaengig machen
(redo)
2
Wert eingeben: 20
Speicher: 70  Wert: 90
(0) Programm beenden
(1) Rechner zuruecksetzen
(2) addieren
(3) subtrahieren
(4) Anzeige in Speicher
(5) Speicher addieren
(6) Speicher subtrahieren
(98) letzte Aktion rueckgaengig
(undo)
0
Speicher: 70  Wert: 90
```

Hinweis: Werden die Ausgabemöglichkeiten nicht in die Methode ausgabe() integriert, laufen die Tests nicht, obwohl das Programm sonst korrekt sein kann. Die Tests prüfen u. a. ob die gewünschten Menüpunkte durch die Methode ausgabe() angezeigt werden.