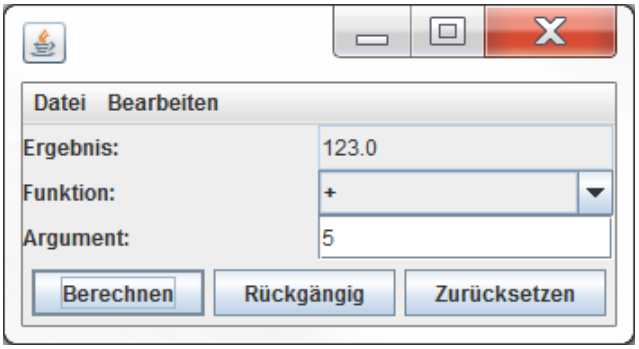
Einstieg in Java und OOP

Christian Silberbauer

Übungsblatt 6

Die Aufgaben dieses Übungsblatts beziehen sich auf die sukzessive Erstellung eines einfachen Taschenrechners.



Durch Auswahl einer binären Funktion aus einer Combobox und der Angabe eines Wertes wird auf Basis des bisherigen Ergebnisses ein neues berechnet und angezeigt.

Zudem wird die Möglichkeit angeboten, Rechenoperationen rückgängig zu machen.

Der eigentliche Taschenrechner soll getrennt von der GUI implementiert werden. Einzelne Funktionen des Taschenrechner sollen beliebig ergänzt werden können und zwar auf Basis des Strategy-Patterns: Funktionen bilden *Strategien* des Taschenrechners.

Aufgabe 1

Jede Funktion wird durch eine Klasse gekapselt und implementiert folgendes Interface:

```
public interface Function {

	double getResult(double x, double y);
}
```

Die Methode `getResult()` ermittelt das Ergebnis auf Basis der zwei Argumente. Implementieren Sie bitte die Funktionen `Addition` und `Multiplication`, welche eine Addition bzw. eine Multiplikation realisieren.

Aufgabe 2

Als nächster Schritt ist der eigentliche Taschenrechner zu implementieren. Er bietet die Möglichkeit, Operationen rückgängig zu machen. Speichern Sie daher die bisherigen Ergebnisse in einem Stack und implementieren Sie die Methoden wie vorgegeben:

```
public class Calculator {

	private Stack<Double> history;
```

```
	public Calculator() { … }
	public void clear() { … }
	public void apply(Function f, double y) { … }
	public void undo() { … }
	public boolean canUndo() { … }
	public double getResult() { … }
}
```

Der `history`-Stack hält initial den Wert `0`, `apply()` wendet eine Funktion auf Basis des bisherigen Ergebnisses und dem Parameter `y` an und fügt das neue Ergebnis dem Stack hinzu, `clear()` löscht den `history`-Stack bis auf den Initialwert, `getResult()` gibt das aktuelle Ergebnis zurück (jenes, das im `history`-Stack ganz *oben* liegt), `canUndo()` liefert `true`, sofern sich mehr als nur der Initialwert im `history`-Stack befindet und `undo()` entfernt das letzte Ergebnis vom Stack, sofern `canUndo()` dies erlaubt.

Sie können den Taschenrechner mit folgendem Code testen:

```
public static void main(String[] args) {
	Calculator c = new Calculator();
	c.apply(new Addition(), 5.0);
	c.apply(new Multiplication(), 3.0);
	c.apply(new Addition(), 6.0);
	System.out.println(c.getResult());
	c.undo();
	System.out.println(c.getResult());
	c.apply(new Addition(), 7.0);
	System.out.println(c.getResult());
}
```

Dies sollte zu folgender Ausgabe führen:

```
21.0
15.0
22.0
```

Aufgabe 3

Implementieren Sie bitte die GUI zum Taschenrechner wie oben abgebildet. Realisieren Sie dazu eine Klasse `CalculatorFrame` mit der Oberklasse `JFrame`, die für die eigentlichen Taschenrechner-Funktionalitäten die Klasse `Calculator` verwendet. Die Aktionen *Berechnen* (neues Ergebnis anhand der Benutzereingaben ermitteln), *Rückgängig* (letzte Berechnung rückgängig machen) und *Zurücksetzen* (Ergebnishistorie löschen) können sowohl über Buttons als auch über das *Bearbeiten*-Menü ausgelöst werden.

Das Feld *Ergebnis* ist ein ReadOnly-Feld. Das Feld *Funktion* beinhaltet die definierten Funktionen. Sie können die Funktionen direkt der Combobox hinzufügen. Zur Anzeige verwendet die Combobox deren `toString()`-Methode.

Wird im Feld *Argument* ein ungültiger Wert eingegeben, so soll im *Ergebnis*-Feld der Text „Ungültige Eingabe!“ ausgegeben werden.

Die *Rückgängig*-Aktion ist nur dann aktiviert, wenn dies auch möglich ist (festgelegt durch `canUndo()`).