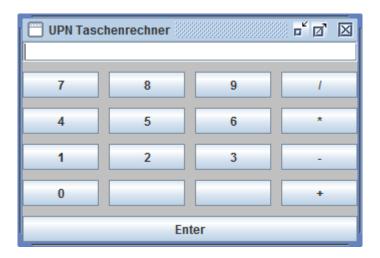


# Übung Taschenrechner

In dieser Übung wird ein einfacher Taschenrechner erstellt. Er soll das Rechnen mit den Grundfunktionen Addition, Subtraktion, Multiplikation und Division mit Ganzzahlen ermöglichen. Der Rechner soll wie ein HP-Tasschenrechner zu bedienen sein ([2] [ENTER] [2] [+]). Die Applikation ist als BorderLayout() mit einem TextField tfOut im Norden, einem Panel panelButton im Zentrum und einem Button btEnter im Süden organisiert. Das panelButton ist als GridLayout(4, 4, 10, 10) aufgebaut.

### Bildschirmausgabe:



### Aufgabe

- a) Beschreiben Sie das Model View Controller Konzept mit eigenen Worten.
- b) Dokumentieren Sie die Methoden jeweils direkt im Source Code mittels Javadoc.
- c) Erstellen Sie das gezeigte User Interface. Die Referenzen für die Tasten sollen in einem zweidimensionalen Array abgelegt sein. Benutzen Sie einen zweiten Array um die Beschriftung der Tasten zu erzeugen.
- d) Erarbeiten Sie die benötigte Struktur, um die Berechnungen auszuführen. Überlegen Sie sich die Aufgaben der einzelnen Klassen und Methoden.
- e) Implementieren Sie die Methoden und ergänzen Sie allenfalls ihre Dokumentation.

Fachhochschule Nordwestschweiz

Klassendiagramm:

### **JFrame** Taschenrechner erzeugt die Struktur und bildet Taschenrechner das Frame der Applikation. - serialVersionUID : long = 1L + Taschenrechner() + main(args : String[]) : void Observable Controller Model - controller - model ENTER: int = 0 - stack : int[] = new int[4] ZAHL : int = 1 + add(): void - OPERATION : int = 2 + subtract(): void - zustand : int = ENTER + divide(): void + Controller(model: Model) + multiply(): void + add(): void + enter(): void + setValue(value : int) : void + subtract(): void + multiply(): void + getValue(): int + divide(): void + notifyObservers(): void + enter(): void - pull(): void + number(i : int) : void - push(): void + getStackInfo(): String Observer

Die VIEW ist das Graphical User Interface der Applikation und Listener für die GUI - Elemente. Die VIEW ändert via CONTROLLER das MODEL und bringt allenfalls geänderte Daten via Observer - Pattern zur Darstellung.

- btEnter

Der CONTROLLER steuert das Programm und besorgt die Kommunikation zwischen VIEW und MODEL.

Das MODEL beinhaltet Daten und zugehörige Methoden zur Manipulation der Daten. Falls die Daten ändern wird notifyObservers() ausgelöst.

---- Model ist weitgehend entkoppelt. ----->

View und Controller sind eng gekoppelt. --

**JButton** 

button

- panelButton

**JPanel** 

View

+ update(modelObject : Observable, dataObject : Object) : void

serialVersionUID : long = 1L

+ View(controller : Controller)

tfOut

JTextField

- stButtonText : String[][] = {},},}}

+ actionPerformed(e : ActionEvent) : void

ActionListener



### Funktionsweise des Stack - Rechners:

stack[3]	0	0	0	0	0	0	0	0	0	0	0
stack[2]	0	0	0	0	0	0	0	0	4	0	0
stack[1]	0	5	5	0	35	35	0	4	4	4	0
stack[0]	5	5	7	35	3	31	4	4	4	16	64

Eingabe	5	Enter	7	*	3	1	-	Enter	Enter	*	*
zustand	ZAHL	ENTER	ZAHL	OPERATION	ZAHL	ZAHL	OPERATION	ENTER	ENTER	OPERATION	OPERATION

## Zugehöriges Zustandsdiagramm:

