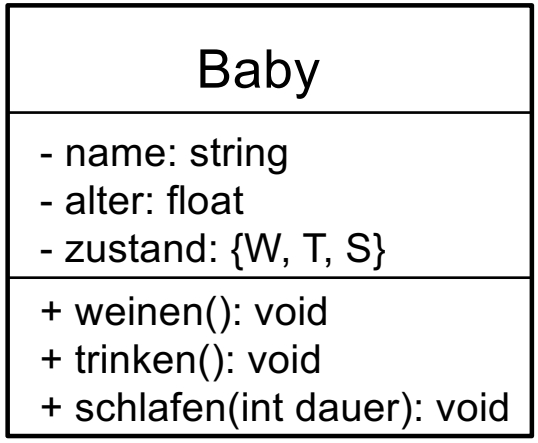
Workshop Übungen

Objektorientierung: Classen und Objekte

Übung: Baby

- Implementieren Sie Klasse Baby in C++.
- In main(): Kreiieren Sie ein Baby Kilian, 3 Monate alt, der zuerst weint, dann zu trinken bekommt und anschließend 3 Stunden schläft. Der Zustand des Babies soll sich jedesmal aktualisieren (W = weinen, T = trinken, S = schlafen).



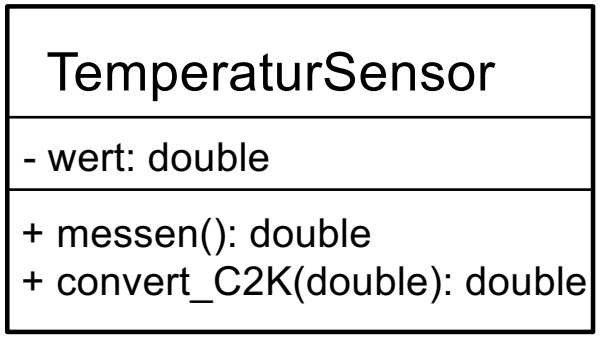
- Allgemeine Hinweise: Achten Sie darauf, dass im Design keine Konstruktor, Destruktor, Getter und Setter modelliert sind. In der Implementierung sind sie jedoch zu berücksichtigen.

Übung: Baby

1. In einer *.h Datei die Klasse Baby mit einem Default Konstruktor, Destruktor, Attributen und Funktionen (inkl. Getter und Setter-Funktionen) deklarieren.
2. Implementieren Sie die Funktionen in der *.cpp Datei. Die Funktionen weinen(), trinken() und schlafen() geben immer den aktuellen Zustand auf der Konsole aus.
3. Realisieren Sie im Hauptprogramm den Tagesablauf von Baby Kilian.

Übung: TemperaturSensor

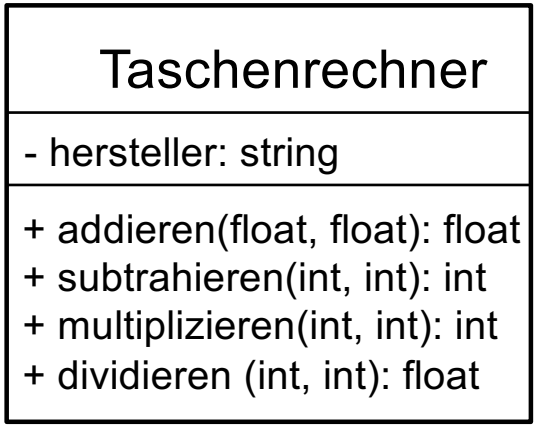
- Realisieren Sie die Klasse TemperaturSensor in C++.
- Kreiieren Sie ein Objekt der Klasse TemperaturSensor in main(). Das Objekt misst anfangs de aktuelle Temperatur in Celsius und konvertiert diese anschließend nach Kelvin um. Geben Sie den Kelvin-Wert auf dem Bildschirm aus.
- Funktioniert die Signatur convert_C2K():double auch?



- Allgemeine Hinweise: Achten Sie darauf, dass im Design keine Konstruktor, Destruktor, Getter und Setter modelliert sind. In der Implementierung sind sie jedoch zu berücksichtigen.

Übung: Taschenrechner

- Realisieren Sie die Klasse Taschenrechner in C++.
- Achten Sie darauf, dass im Design keine Konstruktor, Destruktor, Getter und Setter modelliert sind. In der Implementierung sind sie zu berücksichtigen.
- Kreiieren Sie zwei Objekte der Klasse Taschenrechner in main(). Der erste Taschenrechner soll 5*3 berechnen. Der zweite Taschenrechner soll 5/3 berechnen. Anschließend soll die Summe davon ausgegeben werden.



Übung CandyJar
Teil 1: Klassen Design

Entwerfen Sie eine UML Klasse, die die Eigenschaften und Verhalten eines Bonbon-Glases (*CandyJar*) realisiert:

Ein Bonbon-Glas hat einen maximalen Fassungvermögen von 100 Bonbons.

Man kann eine bestimmt Anzahl von Bonbons entnehmen (*takeCandies*) und einfüllen (fillCandies). Bei beiden Operationen wird der neue Füllstand des Glases ausgegeben.

- Modellieren Sie die Klasse *CandyJar* als UML Klasse mit Attributen und Methoden. Achten Sie auf Sichtbarkeit.
- Tipp: Eventuell benötigen Sie einen Attributen, der den Zustand des Glases (z.B. voll, leer) beschreibt.

Übung CandyJar
Teil 2: Implementierung

Implementieren Sie die Klasse in C++.

- Ergänzen Sie Konstruktoren für *CandyJar*: ein leeres Glas und ein gefülltes Glas, welches auch eine maximale Füllung hat.
- Ergänzen Sie den Destruktor für *CandyJar.*
- Implementieren Sie getter und setter Methoden zur Klasse *CandyJar*.
- Implementieren Sie die Operationen *takeCandies()* und *fillCandies().*
- Erstellen Sie ein Objekt der Klasse CandyJar mit einem vollen Glas. Entleeren Sie anschließend das komplette Glas und füllen Sie es wieder zur Hälfte auf.