Hochschule RheinMain
Fachbereich Design Informatik Medien
Marcus Thoss, M.Sc.

Hardwarenahe Programmierung II
SS 2020
LV 2512

Übungsblatt 2

Aufgabe 2.1 (Die erste eigene C++-Klasse):

In dieser Aufgabe realisieren Sie die `Animal`-Klasse des ersten Übungsblatts in C++.

a) Wenn Sie Ihren Quellcode des letzten Aufgabenblatts noch nicht mit `git` versioniert haben: holen Sie dies jetzt nach.

b) Legen Sie sich dann in Ihrem `git`-Repository ein neues Unterverzeichnis für diese Aufgabe an und darin ein noch leeres `Makefile`.

c) Wir verwenden ab hier für C++-Quellcode die Dateiendung `.h` für Deklarationen (z.B. von Klassen) und `.cc` für Implementierungen (z.B. der Member Functions einer Klasse). Für die Klasse `Animal` legen Sie zunächst eine Datei an, in der ein Testprogramm für die `Animal`-Klasse implementiert wird: `animal_test.cc`

d) Implementieren Sie in `animal_test.cc` eine einfache `main()`-Funktion, die zunächst nur den Text *Animal Test* ausgibt (nutzen Sie `std::cout`).

   Erweitern Sie das `Makefile` so, dass das Programm `animal_test` erzeugt wird (der Compiler-Aufruf erfolgt nun mit `g++` statt `gcc`).

   Sehen Sie auch gleich ein `clean`-Target zum "Aufräumen" vor.

e) Legen Sie die Datei `animal.h` mit einer Deklaration der Klasse `Animal` an. Die Klasse soll die in Aufgabe 1.3 beschriebenen Data und Function Members besitzen.

   Entscheiden Sie dabei auch, welche Teile der Klasse `public` und welche `private` sein sollen.

f) Fügen Sie der Klasse einen Constructor (ohne Parameter) und einen Destructor hinzu.

g) Implementieren Sie die Member Functions Ihrer Aufgabe entsprechend in einer neuen Datei `animal.cc`, (einschließlich `#include "animal.h"` !). Constructor und Destructor sollen dabei leere Implementierungen besitzen.

   In `animal.h` sollen in dieser Aufgabe nur die Deklarationen, nicht die Implementierungen der Member Functions stehen.

h) Erweitern Sie jetzt `animal_test.cc` so, dass

   - eine `Animal`-Variable angelegt wird
   - das Initialgewicht des Tieres ausgegeben wird
   - 1,5 kg gefüttert werden und
   - das Endgewicht ausgegeben wird.

Aufgabe 2.2 (Die Dokumentation mit `doxygen`):

a) Legen Sie eine Datei `animal.puml` an mit einer PlantUML-konformen Beschreibung der Klasse entsprechend Aufgabe 1.3. Berücksichtigen Sie dabei nun auch die Sichtbarkeit (Visibility) der Elemente der Klasse.

   Erweitern Sie das `Makefile` so, dass daraus mit PlantUML die Grafikdatei `animal.png` erzeugt wird.

b) Um mit `doxygen` arbeiten zu können, müssen Sie zunächst die Konfigurationsdatei `Doxyfile` generieren (wie geht das?). Die generierte Datei (eine Textdatei) können Sie später nach Ihren Wünschen anpassen.

c) Fügen Sie in `animal.h` über der Klassendeklaration einen `doxygen`-Kommentar (beginnend mit `/**`) ein. Darin beschreiben Sie mit `@brief` kurz die Klasse in einem Satz und, noch in demselben Kommentar, mit einem weiteren Absatz einige Details, wozu die Klasse verwendet werden könnte.

   Rufen Sie `doxygen` auf und, falls kein Fehler auftritt, sehen Sie sich mit einem Webbrowser die erzeugte Datei `html/index.html` an. Sie sollten darin zu dem gerade geschriebenen Kommentartext navigieren können.

d) Erweitern Sie `animal.h` um jeweils einen `doxygen`-Kommentar vor jeder Member Function und fügen Sie einen `doxygen`-Kommentar *hinter* dem Data Member `weight` ein. Nutzen Sie für die Member Functions jeweils `@brief` und, sofern zutreffend, `@param` und `@return`.

e) Erweitern Sie das `Makefile` so, dass `doxygen` bei jeder relevanten Quellcode-Änderung erneut ausgeführt wird, um die erzeugte Dokumentation zu aktualisieren.

f) Fügen Sie schließlich eine neue Datei `hwp2_p2.dox` hinzu, die die "Main Page" Ihrer Dokumentation füllen soll (`.dox`-Dateien werden von `doxygen` wie Quellcode-Dateien automatisch berücksichigt).

   Schreiben Sie in die Datei einen `doxygen`-Kommentar mit dem Tag `@mainpage`, um einen Text auf der Einstiegsseite erscheinen zu lassen, der die Aufgabenstellung erwähnt und verwenden Sie `@image`, um `animal.png` in Ihren Text einzubetten.

Aufgabe 2.3 (Erweiterungen):

a) Erweitern Sie die Klasse um einen Data Member `name` mit einem Namen für das jeweilige Tier (nutzen Sie hierfür die `string`-Klasse) und einer Member Function `getName()`, die den Namen (als `string`) zurückgibt.

b) Erweitern Sie den Constructor um einen `string`- und einen `float`-Paramter, die den passenden Data Members für Name und Gewicht zugewiesen werden.

c) Passen Sie die Dokumentation entsprechend an.

d) Erweitern Sie das Testprogramm `animal_test` so, dass die Tiere *Frank* und *Igor* mit 2,5 und 4,7 kg in einem Array angelegt werden.

e) Sehen Sie eine *Schleife* vor, in der

- eine Eingebeaufforderung *(o)utput (f)eed (q)uit?* ausgegeben wird,
- ein Zeichen eingelesen wird, das bestimmt, ob
  - die Namen und Gewichte der Tiere ausgegeben werden (`o`)
  - ein Tier gefüttert wird (`f`, daraufhin wird nach dem Index des Tieres im Array und der Fütterungsmenge gefragt) oder
  - das Programm beendet wird (`q`).

f) Testen Sie Ihr Programm und überlegen Sie sich andere Erweiterungen.

g) Vergessen Sie nicht, Ihren Quellcode regelmäßig zu versionieren ...