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

Hardwarenahe Programmierung II
SS 2020
LV 2512

Übungsblatt 4

Aufgabe 4.1 (Instrumentierung und Klassenpflege):

Erweitern Sie die Klassenhierarchie von "Beings" und "Animals" aus Aufgabenblatt 3.

a) Wenn nicht bereits geschehen, schreiben Sie ein Makefile für Ihren Code und verwenden Sie es mit `make`, um Ihr Testprogramm zu generieren (testen Sie dann die Ausführbarkeit Ihres Programms auf der Kommandozeile).

b) Ergänzen Sie Destructors, Copy Constructors und Assignment Operators für Ihre Klassen. Ist dies für jede Klasse sinnvoll?

c) Fügen Sie in allen Constructors, Destructors und Methoden Ausgabeanweisungen auf `cerr` hinzu, die angeben, welche Methode (in welcher Klasse) gerade ausgeführt wird.

d) Testen Sie Ihre neuen Ausgaben mit den bisherigen Tests.

e) Fügen Sie neue Tests `TestLocals()` `TestLocalStatics()` und `TestGlobals()` hinzu, in denen Sie von Ihren Objekten jeweils eine lokale Variable in einer Funktion, eine loakle `static`-Variable in einer Funktion und eine globale Variable anlegen.

f) Erweitern Sie ggf. die Tests so, dass *alle* Ausgaben mindestens ein Mal angezeigt werden. Für die Copy Constructors und Assignment Operators genügt dabei die beispielhafte Verwendung an einer der Klassen.

g) Protokollieren Sie die gesamte Ausgabe mittels Umleitung (auf der Shell mittels `2>` für `cerr`) in einer Textdatei und versionieren Sie diese ebenfalls.

Aufgabe 4.2 (Test 1):

Die Ergebnisse zu den folgenden Testaufgaben müssen für die Berücksichtigung der Leistung als Archiv mit dem Dateinamen `<Nachname>_<Vorname>_hwp2s20_t01.tar.gz` am

28.05.2020 bis zum *Tagesende* per Email an marcus.thoss@hs-rm.de gesendet werden. Das Archiv soll ein PDF-Dokument namens `<Nachname>_<Vorname>_hwp2s20_t01.pdf` mit Antworten zu allen Fragen und separate Quellcode-Dateien für die Lösung der Aufgaben b) und c) enthalten.

Sofern nicht anders angegeben, wird für eine vollständig gelöste Teilaufgabe ein Punkt vergeben. Jeglicher C++-Code muss den Styleguide (`StyleGuide.pdf`) einhalten, der auf dem Laborserver zur Verfügung gestellt ist und soll mit `g++ -Wall` sauber compilierbar sein.

a) Erläutern Sie das Prinzip der Vererbung und mit welchen Mitteln es in C++ umgesetzt werden kann. Auf die Vererbungsvarianten `public`, `protected` und `private` brauchen Sie an dieser Stelle noch nicht einzugehen. Überlegen Sie sich zur Erläuterung Ihrer Ausführungen auch drei Beispielklassen, die Sie beschreiben und Quellcode für ihre Definition angeben. (2 Punkte).

b) Zeichnen Sie für die folgende Anforderungsdefinition ein UML-Klassendiagramm und geben Sie C++-Quellcode für dessen Umsetzung an. Hierzu gehört die Klassendefinitionen inkl. Data Members und Methoden. Die Implementierung des tatsächlichen Herunterzählens und Anzeigens des Zählers können Sie hier ignorieren, das Setzen von Werten durch Methoden soll aber berücksichtigt werden. Achten Sie auf sauber compilierbaren Code (Test mit `g++ -Wall`). (2 Punkte):

> A cooking timer has a counter that counts down seconds. It can be bought with different predefined counter start values. A user can perform these actions: The timer can be triggerd to start counting down, it can be stopped, and it can be reset, which automatically sets its counter value back to the predefined start value. The start value can be modified by a user.

c) Betrachten Sie folgende in UML notierte Klassenhierarchie und realisieren Sie sie in C++. Ergänzen Sie eine `main()`-Funktion, die daraus verschiedene Objekte erzeugt und verwendet. Ergänzen Sie ggf. Elemente, die für die Realisierung notwendig, im Diagramm aber nicht erfasst sind. (6 Punkte).