Abschlussübung Programmieren 1

Autoren: Skroch, Trapp

Richtwert Bearbeitungszeit für diese Abschlussübung: vier Stunden „open book“,
wenn Sie Ihre „open book“ Unterlagen gut vorbereitet haben
(Zeit in der Rechnerprüfung: drei Stunden).

Es ist hier natürlich nicht der komplette Stoff erforderlich, der in der Rechnerprüfung relevant wird; es fehlen z.B. die abgeleiteten Klassen mit den virtuellen Methoden.

Allgemeine Hinweise

- *Bleiben Sie ruhig und lesen Sie sich, bevor Sie anfangen, diese Aufgabenstellung erst einmal komplett durch.*
- Stellen Sie sicher, dass der von Ihnen abgegebene Quellcode fehlerfrei (und möglichst warnungsfrei) übersetzt, z.B. bei der gcc Toolchain mit der Compileroption `-Wall`, und dass Ihr Programm läuft.
- Lesen Sie die Angaben genau, machen Sie nicht weniger, aber auch nicht mehr, arbeiten Sie zügig in der Reihenfolge der Fragestellungen.
- Gehen systematisch Schritt für Schritt in kleinen Schritten vor, sorgen Sie dafür, dass Ihr Quellcode möglichst immer übersetzt und läuft. Denken Sie daran, Ihren Quellcode regelmäßig zu speichern.
- Kommentieren Sie Ihren Quellcode prägnant an komplizierten Stellen.
- Fertigen Sie als Lösung der Aufgabe genau ein komplettes Programm an. Sorgen Sie dafür, dass Sie am Ende keine Teilprogramme mit Teillösungen mehr haben. Erstellen Sie auch nicht mehrere Programmversionen.

Beachten Sie bei Ihrer Lösung auch:

- Formale Klarheit der Programme (u.a. Formatierung und Aufteilung des Quellcodes).
- Inhaltliche Klarheit der Programme nach den Regeln der Kunst.
- Verwendung der in dieser Angabe vorgegebenen Namen und Bezeichner exakt so, wie sie vorgegeben sind.
- Einsatz von L-Referenzen, sowie von Qualifizierern wie *const*, *static*, *virtual* usw., z.B. bei der *const*-Korrektheit von Methoden, oder z.B. bei Basisklassen und abgeleiteten Klassen. (Hinweis: ist in der Angabe i. Allg. nicht spezifiziert, Sie müssen hier also selbst überlegen.)

Legen Sie in einem Arbeitsverzeichnis, welches Sie eindeutig über Ihre Matrikelnummer identifiziert, ein neues C++ Projekt ***prog1-fp*** an und stellen Sie die Optionen Ihrer Toolchain/IDE für das Projekt richtig ein. Ihre Lösung für diese Abschlussaufgabe besteht aus drei Dateien:

- (Nur) Die Anwendung in der Quellcode-Datei namens *fp-main.cpp*
- Alle Deklarationen in der Quellcode-Datei namens *fp.h*
- Alle Definitionen in der Quellcode-Datei namens *fp.cpp*

Es sollen – im Zeitrahmen der Aufgabenstellung stark vereinfacht – Grundfunktionen für ein „social network" namens *LinkedLiwanzen* programmiert werden. Es soll zur Vereinfachung der in der Vorlesung besprochene *std::vector* Containertyp eingesetzt werden.

Legen Sie zunächst die folgende Klasse an:

Liwanze
- name : string *- loc : Region = ndef*
+ Liwanze( ) = delete *+ Liwanze(string, Region)* *+ get_name( ) : string* *+ get_loc( ) : Region* *+ print( ) : void*

Liwanze::Region ist dabei eine Aufzählung (Enumeratoren: *ndef*, *amer*, *apac*, *emea*), die auch Klassenmember ist.

Sorgen Sie dafür, dass ein *Liwanze* Objekt unter keinen Umständen die leere Zeichenkette als Name haben kann.

Auf der Grundlage der *Liwanze* Klasse entwickeln Sie nun einige grundlegende Funktionalitäten; der *Liwanze* Typ ist dabei nicht unbedingt auf die Member des Klassendiagramms beschränkt.

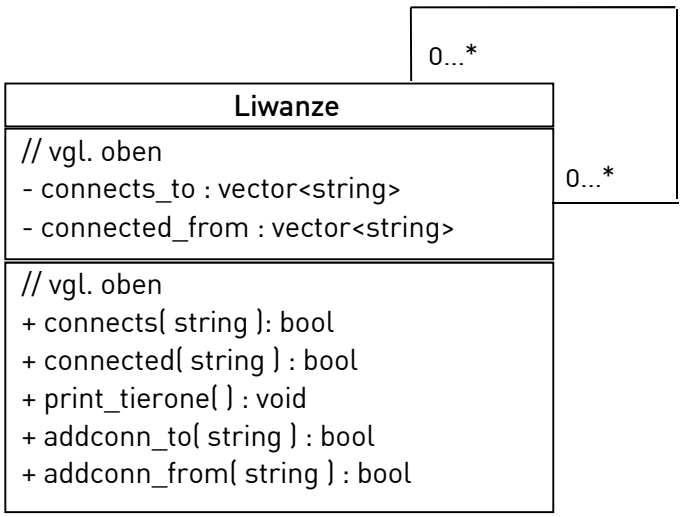
Legen Sie in *main()* ein geeignetes Objekt namens *vL* für die Speicherung von beliebig vielen *LinkedLiwanzen*-Mitgliedern an und füllen Sie es mit sechs Mitgliedern: *Joey* (*amer*), *Johnny* (*amer*), *DeeDee* (*amer*), *Tommy* (*emea*), *Suzy* (apac) und *Sheena* (*amer*).

Geben Sie in *main()* mittels einer weiteren Supportfunktion namens *print_members()* eine komplette Liste mit allen gespeicherten Mitgliedern und ihren Details möglichst benutzerfreundlich aus (benutzerfreundlich bedeutet auch, dass *Region* Werte nicht als Zahl erscheinen).

Lassen Sie den Benutzer mittels einer Supportfunktion namens *add_Liwanze()* beliebig viele weitere Mitglieder hinzufügen, wobei jeder *LinkedLiwanzen* Name in *vL* eindeutig sein muss. Lassen Sie zur Vereinfachung keine Eingaben für die Region zu, die Sie immer auf *ndef* setzen. Die Eingabe eines bereits vorhandenen Namens wird abgelehnt, eine sinnvolle Meldung wird ausgegeben und das Programm fordert den Benutzer erneut zur Eingabe auf. Bei sonstigen Eingabefehlern lösen Sie Ausnahmen aus, die zum Programmabbruch mit Fehlermeldung führen.

Nach jeder gelungenen Registrierung einer neuen "Liwanze" wird eine komplette Liste mit allen *LinkedLiwanzen* Mitgliedern und deren Details möglichst benutzerfreundlich ausgegeben. Durch Eingabe von *q* oder *Q* wird die Registrierung weiterer Mitglieder beendet.

Erweitern Sie nun das Programm wie hier dargestellt:



Die Methode *Liwanze::connects()* überprüft, ob es zu der Liwanze laut Parameter eine direkte Verbindung (im sog. "tier one") gibt; die Namen der ausgehenden Verbindungen sind in *Liwanze::connects_to* gespeichert. Analog dazu prüft die Methode *Liwanze::connected()*, ob es von der Liwanze laut Parameter eine direkte Verbindung (im sog. "tier one") gibt; die Namen der eingehenden Verbindungen sind in *Liwanze::connected_from* gespeichert..

Auf der Grundlage dieser reflexiven Assoziation für *Liwanze* Objekte entwickeln Sie Ihr Programm nun weiter; der *Liwanze* Typ ist dabei nicht auf die Member des Klassendiagramms beschränkt.

Denken Sie daran, auch Ihre bisherigen Programmteile anzupassen.

Vernetzen Sie im Quellcode die sechs ursprünglich registrierten *LinkedLiwanzen*-Mitglieder über eine Supportfunktion namens *init_connections()* wie folgt:

	DeeDee	Joey	Johnny	Sheena	Suzy	Tommy
DeeDee					x	
Joey	x		x	x	x	
Johnny		x			x	
Sheena		x				
Suzy						
Tommy					x	

Zeilen *connects_to* (z.B. Joey connects_to Suzy)
Spalten *connected_from* (z.B. Suzy connected_from Tommy)

Lösen Sie dies mit einer Supportfunktion namens *make_connection()*, die aus *init_connections()* aufgerufen wird; setzen Sie *make_connection()* auch in den weiteren Aufgabenteilen ein.

Geben Sie dieses initiale Netzwerk komplett und so übersichtlich und benutzerfreundlich wie möglich aus.

Lassen Sie den Benutzer das Netzwerk beliebig durch Eingabe von zwei unterschiedlichen Namen erweitern – es wird dann vom zuerst eingegebenen Namen zu dem zweiten eingegebenen Namen eine Verbindung gewünscht. Ist mindestens einer der beiden Namen nicht registriert, oder wird derselbe Name zweimal eingegeben, wird eine sinnvolle Meldung ausgegeben und erneut zur Eingabe zweier Namen aufgefordert (bei sonstigen Eingabefehlern wird eine Ausnahme ausgelöst, die zu einer Fehlermeldung und zum Programmabbruch führt).

Ist die Verbindung bereits vorhanden, wird eine informative Meldung angezeigt und es wird erneut zur Eingabe zweier Namen aufgefordert. Ist die Verbindung noch nicht vorhanden, wird sie erstellt, es wird eine Erfolgsmeldung ausgegeben und wiederum zur Eingabe zweier Namen aufgefordert.

Stellen Sie sicher, dass jede Verbindung konsistent unter beiden Namen abgespeichert wird.

Nach jeder erfolgreich abgespeicherten neuen Verbindung wird das vollständige Tier-One-Netzwerk (d.h. alle direkten Beziehungen) des ersten Namens *und* das des zweiten Namens, mit allen Details (einschließlich der Anzahlen der jeweils ein- und ausgehenden Beziehungen) möglichst benutzerfreundlich angezeigt.

Durch Eingabe von *q* oder *Q* wird das Programm beendet.

VIEL ERFOLG !