5. Semester, 1. Prüfung

Name	

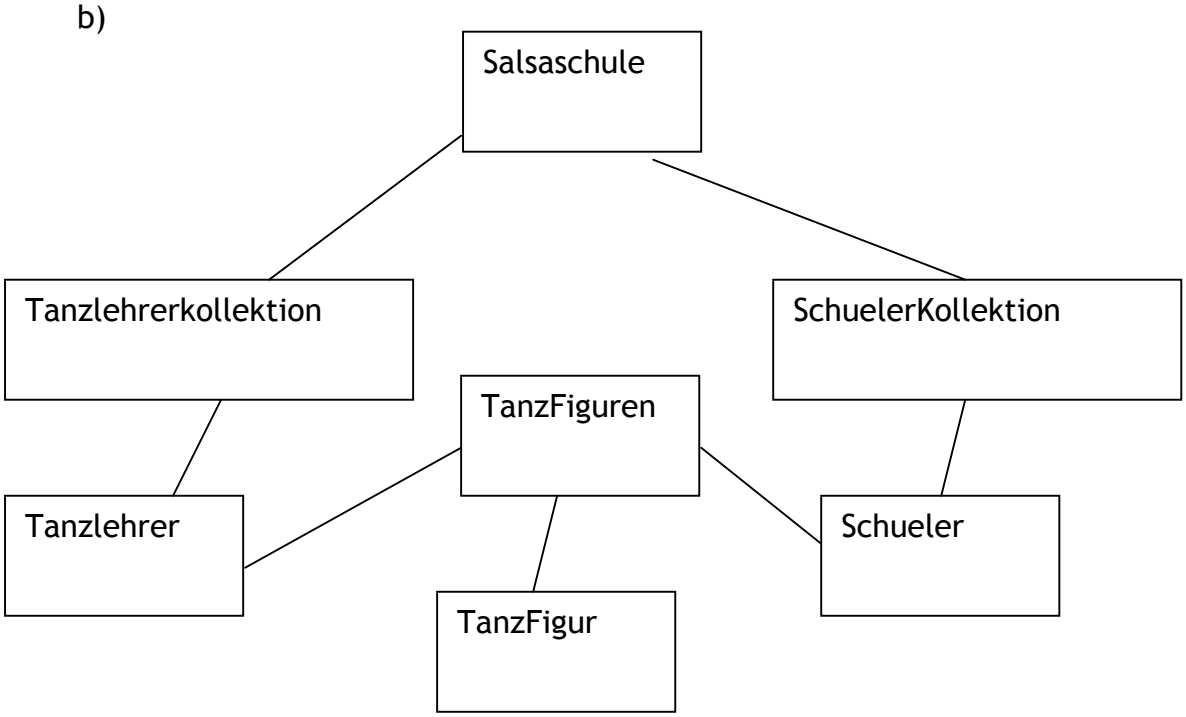
- Die gesamte Prüfung bezieht sich auf die Programmierung in C++!
- Prüfungsdauer: 90 Minuten
- Lösungen können direkt auf die Aufgabenblätter oder auf deren Rückseite geschrieben werden
- PC's sind nicht erlaubt
- Unterlagen und Bücher sind erlaubt
- Achte auf Details wie Punkte, Kommas und Semikolons

Aufgabe	Punkte	
Analyse und Design	10	
Windows-API	10	
Objektorientierte Programmierung	10	
Programmverständnis	10	
Weitere Fragen	10	
Total	50	

1. Analyse und Design

a) Zeichne für folgende Problemstellung die möglichen Klassen auf (als einfaches Klassendiagramm) und verdeutliche ihre Abhängigkeiten mit Linien untereinander. Du musst keine Methoden, Funktionen oder Datenelemente definieren.
Ein Tanzlehrer für Salsa, der ein wenig was von Computern versteht, möchte seine Salsaschule modellieren. Er setzt sich hin und schreibt sich auf, was seine Salsaschule ausmacht.
In seiner Salsaschule gibt es mehrere Tanzlehrer. Ein Tanzlehrer beherrscht verschiedene (Tanz-) Figuren. Er will auch seine Studenten erfassen und sieht, dass auch jeder Student einige Tanzfiguren tanzen kann.
Welche Klassen kannst du identifizieren? (7 Punkte)
Tipp: Falls du eine Klasse hast, wo die Mehrzahl gleich der Einzahl ist (Lehrer), dann kannst du dem Klassennamen einfach das Wort -Kollektion anhängen.

b)



Definiere eine Klasse *Person* mit den Datenelementen, die es braucht um den Vornamen und den Nachnamen zu speichern. Die Klasse soll nur einen Konstruktor mit Argumenten (Vorname, Nachname) haben. Definiere den Konstruktor inline (2 Punkte)
Definiere nun darunter einen Datentypen *Personen* als Kollektion von Person, indem du eine Container-Klasse aus der STL (Standard Template Library) verwendest. (1 Punkt)
Vergiss nicht die nötigen #includes!
(Total 3 Punkte)

```
#include <vector>
#include <string>


class Person
{
    public:
        Person(const std::string& vname,
               const std::string& nname)
        :_vname(vname), _nname(nname)
        {
        }

    private:
        std::string _vname;
        std::string _nname;
};

typedef std::vector<Person> Personen;
```

2. Windows-API

a) Events!
Win-32 Events können verwendet werden um einem anderen Thread ein Signal zu geben. Du weißt bestimmt noch wie man ein Event erzeugt und wieder schliesst/zerstört.
Zum Setzen des Signals gibt es die Win-32 Funktion *SetEvent* und zum zurücksetzen die Funktion *ResetEvent*.
Schreibe eine Klasse *Event*. Diese Klasse soll einen Konstruktor und einen Destruktor haben. Der Konstruktor soll ein Win-32 Event erzeugen, der Destruktor dieses wieder zerstören.
Schreibe auch die zwei Methoden *set* und *reset*, die jeweils das Event setzen oder zurücksetzen.
Definiere alle Methoden inline. (5 Punkte)

```
#include <windows.h>

class Event
{
    public:
        Event()
        {
            _h = CreateEvent(0, FALSE, FALSE, 0);
        }
        ~Event()
        {
            CloseHandle(_h);
        }
        void set()
        {
            SetEvent(_h);
        }
        void reset()
        {
            ResetEvent(_h);
        }
    private:
        HANDLE _h;
};
```

```
Wie heisst die Win-32 Funktion, mit der man einen Thread
erzeugt?
Nur Funktionsnamen hinschreiben (1 Punkt)



CreateThread
```

b) Was musst du „includieren“ (#include), um Win-32 Funktionen und Datentypen zu verwenden? (1 Punkt)
Schreibe die vollständige #include-Zeile hin.

#include <windows.h>

c) Wie nennt man die Art von einer Datei oder einem seriellen Port zu lesen, bei der man den Lesebefehl zwar absetzt, aber **nicht** auf das Ergebnis wartet? (Nur eine Lösung) (1 Punkt)

- o isochron
- o synchron
- o **asynchron**
- o FILE_FLAG_OVERLAPPED

d) Wie heisst die Datei, die du mit CreateFile öffnen musst, um auf den seriellen Port 1 zuzugreifen? (1 Punkt)

```
HANDLE h = CreateFile(??????,


COM1
```

e) Wie heisst die Struktur, die du beim nicht-synchronen Lesen oder Schreiben der ReadFile oder WriteFile Funktion als letztes Argument übrgeben musst (nur Name der Struktur hinschreiben!)
(1 Punkt)

OVERLAPPED

3. Objektorientierte Programmierung

a) Templates.
Schreibe eine Funktion *WriteStringLine*. Diese Funktion nimmt als Argument einen string und gibt diesen auf der Konsole aus. Zusätzlich gibt die Funktion ein *std::endl* aus, so dass die Zeile abgeschlossen wird (WriteLine heisst: SchreibeZeile). Vergiss nicht die nötigen includes. (2 Punkte)
Schreibe nun eine Template-Funktion *WriteLine*. Diese soll für frei bestimmbare Datentypen funktionieren und so wie die *WriteStringLine* das Argument ausgeben und ein std::endl hinzufügen. (2 Punkte)
Schwierige Zusatzfrage: Welche Eigenschaft muss der Datentyp, für den man die *WriteLine*-Funktion verwendet haben? (1 Punkt)
(Total 5 Punkte)

```
void WriteStringLine(const std::string& s)
{
    std::cout << s << std::endl;
}


template <class T>
void WriteLine(const T& t)
{
    std::cout << t << std::endl;
}


Es muss der operator << für ostream definiert sein.
```

```
Objektorientierte Design-Patterns kennen wir als:
(nur eine Lösung) (1 Punkt)
```

- o Stoffmuster
- o Programmiertricks um weniger Code zu schreiben
- o Besondere Anwendung von Templates
- o **Entwurfsmuster um wiederkehrende Probleme zu lösen**

b) Schreibe eine Klasse *Salsaschule* und wende hierfür das **Singleton**-Pattern an. Es soll also nur ein Objekt der Klasse *Salsaschule* geben. Die Klasse soll nur eine Methode *lassDiePuppenTanzen* haben, die nur „Tanz Baby“ auf der Konsole ausgibt.
Trenne die Klassendeklaration von der Implementation. Schreibe also zuerst den Teil, der in die Header-Datei gehört und den Teil, der in die Quellcode-Datei (.cpp) gehört darunter. Vergiss nicht die nötigen includes. (4 Punkte)
(Auf der nächsten Seite ist auch Platz).

```
class Salsaschule
{
    public:
        static Salsaschule& get();
        void lassDiePuppenTanzen();
    private:
        Salsaschule();
};
```

```
#include <iostream>
#include „Salsaschule.h“

Salsaschule::Salsaschule()
{
}

void Salsaschule::lassDiePuppenTanzen
{
    std::cout << „Tanz Baby“ << std::endl;
}


Salsaschule& Salsaschule::get()
{
    static Salsaschule s;
    return s;
}
```

4. Programmverständnis

a) Was gibt das folgende kleine Programm aus?
Beachte die Berechnung genau und achte auch darauf, wie die Werte ausgegeben werden (Trennzeichen). (3 Punkte)

```
#include <vector>
#include <iostream>

typedef std::vector<double> Werte;

int main()
{
    Werte werte;

    for(int i = 0; i < 6; ++i)
    {
        double wert = (double)i + (double)i/10.0;
        werte.push_back(wert);
    }

    Werte::iterator it = werte.begin();
    Werte::iterator end = werte.end();

    while(it != end)
    {
        std::cout << *it << ";";
        ++it;
    }

    return 0;
}
```

```
0.0;1.1;2.2;3.3;4.4;5.5;
```

b) Was hat die Variable „bbb“ am Ende dieser Aufrufe für einen Wert? (1 Punkt)

```
    bool bbb = false;
    bbb = (34 < 45);
```

```
true
```

Was gibt folgendes Programm aus? (Genau, also auch endl beachten)
Lass Dich nicht zu stark verwirren, es gibt zwei kleine Stolpersteine.
Beachte genau, wie der string in der ersten Schleife zusammengesetzt wird. Lass dich nicht durch unnötige Aufrufe verwirren. (3 Punkte)

```
#include <string>
#include <sstream>
#include <iostream>
#include <list>

typedef std::list<std::string> StringList;

int main()
{
    const char* Texte[3] = { "Eins", "Zwei", "Drei" };
    StringList stringListe;
    for(int i = 0; i < 3; ++i)
    {
        std::stringstream stream;
        stream << i << ":" << Texte[i];
        stringListe.push_back(stream.str());
    }
    stringListe.sort();
    StringList::iterator it = stringListe.begin();
    StringList::iterator end = stringListe.end();
    for( ; it != end; ++it)
    {
        std::cout << *it << std::endl;
    }
    return 0;
}
```

```
0 Eins
1 Zwei
2 Drei
```

c) Schwierigere Aufgabe mit überladenen Operatoren.
Was gibt folgendes Programm aus? (3 Punkte)
Finde heraus welche Operatoren aufgerufen werden.

```
#include <iostream>

class Clever
{
public:
    operator bool()
    {
        std::cout << "Hi ";
        return true;
    }
    operator long()
    {
        std::cout << "Ho ";
        return 1L;
    }
    bool operator==(long arg)
    {
        std::cout << "Ha ";
        return false;
    }
};

int main()
{
    Clever clever;
    if(clever)
    {
        std::cout << "Jupii" << std::endl;
    }
    if(clever == 1L)
    {
        std::cout << "Jipii" << std::endl;
    }
    if(1L == (long)clever)
    {
        std::cout << "Yeah" << std::endl;
    }
    return 0;
}
```

```
Hi Jupii
Ha Ho Yeah
```

5. Weitere Fragen

a) Mit welcher Klasse aus der STL lassen sich Zahlen in strings umwandeln und umgekehrt? (1 Punkt)
Was muss man dafür includieren? (1 Punkt) (Total 2 Punkte)

stringstream
#include <sstream>

b) Was verwendest du bei Programmen um unerwartete Ereignisse und Ausnahmen während der Laufzeit zu behandeln?
(1 Punkt) (Nur eine Lösung)

- o assert oder _ASSERT
- o **C++ Exceptions (try-catch)**
- o bool - Rückgabewerte
- o Gar nichts, denn unerwartetes interessiert mich nicht

c) Ein iterator entspricht vereinfacht gesprochen einem…
(1 Punkt) (Nur eine Lösung)

- o **Zeiger auf den Datentypen, der sich im entsprechenden Container befindet.**
- o Pfeil im Köcher des C++ Pfeilbogenschützen (Programmierer)
- o Objekt des Datentypen, der sich im entsprechenden Container befindet.
- o Referenz auf den Datentypen, der sich im entsprechenden Container befindet.

d) Mit welcher Klasse aus der STL lassen sich Dictionaries (z.B. English - Deutsch) leicht implementieren? (1 Punkt)
Was muss man dafür includieren? (1 Punkt) (Total 2 Punkte)

map
#include <map>

e) Womit verhindere ich, dass von einer Klasse beliebig viele Objekte erzeugt werden. Was muss diese Klasse haben, damit man nicht einfach Objekte erzeugen kann? (1 Punkt)

private Konstruktor

f) Bei der Analyse von einem Problem hast du das Gefühl, dass dieses Design-Problem häufig vorkommt. Du kannst nicht glauben, dass noch nie jemand dieses Problem hatte und du bist davon überzeugt, dass es eine allgemeine Lösung, ein Muster dafür geben muss.
Wo würdest du zuerst nachsehen?
(1 Punkt) (Nur eine Lösung)

- o C++ Programmier-Referenz, Nachschlagewerk
- o **Entwurfsmuster. Elemente wiederverwendbarer Objektorientierter Software (Design Patterns)**
- o Salsa für Anfänger (Überzeuge deine Tanzpartnerin)
- o Im C++ Unterrichtsstoff von M. Burri

g) In welcher Funktion einer Klasse würdest du Aufräumarbeiten wie Speicher freigeben, HANDLEs schliessen und Zähler zurücksetzen erledigen? (1 Punkt)

Destruktor

h) Willst Du fehlerfreie Programme schreiben? (1 Punkt)

Ja