5. Semester, 1. Prüfung

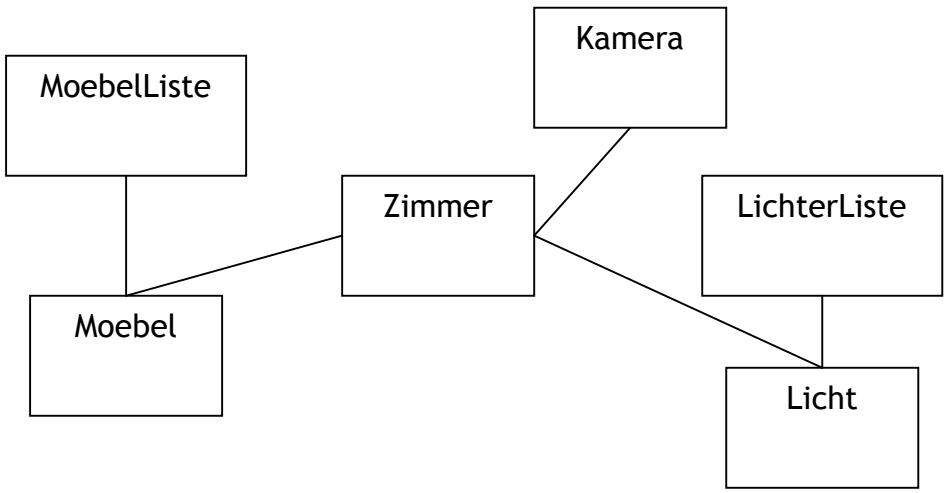
Name	

- Die gesamte Prüfung bezieht sich auf die Programmierung in C++!
- Prüfungsdauer: 90 Minuten
- Mit Kugelschreiber oder Tinte schreiben
- Lösungen können direkt auf die Aufgabenblätter 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		
Exceptions		
Programmverständnis		
Weitere Fragen		
Total		

1. Analyse und Design

a) Zeichne für folgendes Problemstellung die möglichen Klassen auf (als einfaches Klassendiagramm) und verdeutliche ihre Abhängigkeiten mit Linien untereinander. Du musst keine Methoden, Funktionen oder Datenelemente definieren.
Es soll ein Zeichnungsprogramm für ein Möbelverkaufsgeschäft geschrieben werden, mit dem man Möbel in einem Zimmer plazieren kann. Der Benutzer hat die Möglichkeit einen Raum mit Abmessungen und Farben zu definieren. Danach hat er die Möglichkeit Möbel aus einer Liste auszuwählen und diese in das Zimmer zu „stellen“ (zum Beispiel mit ‚Drag and Drop’). Schlussendlich kann er auch eine Kamera und mehrere Lichtquellen in den Raum stellen, um sich danach eine 3-D Ansicht berechnen zu lassen.
Definiere auch Klassen für Listen oder andere Sammlungen von Objekten. (6 Punkte)



b) Für eine Problemstellung wird eine Klasse *Gast* definiert, die verwendet wird um Gäste in einem Hotel zu abstrahieren. Diese Gäste kommen und gehen zu beliebigen Zeiten. Es soll nun eine Container-Klasse gewählt werden, mit der diese Gäste in einem Buchungssystem verwaltet werden können. Welche Container-Klasse aus der STL verwendest du? (1 Punkt)
Definiere einen neuen Datentypen mit typedef, mit dem Namen *Gaeste* und dem gewählten Container für Elemente der Klasse *Gast*. (2 Punkte).
Erzeuge zuletzt noch eine Variable dieses neuen Datentypen (1 Punkt).
Vergiss nicht die notwendigen #include-Anweisungen und die namespace-Spezifizierer. (Total 4 Punkte)

```
kommen und gehen zu beliebigen Zeiten, keine Echtzeit -> Liste
also std::list

#include <list>

typedef std::list<Gast> Gaeste;


Gaeste meineGaeste;
```

2. Exceptions

a) Schau Dir folgende Klasse *File* an. Diese Klasse wirft Exceptions wenn etwas schief geht. Schreibe ein kleines Programm, das ein File-Objekt erzeugt, mit *getChar* ein char liest, dieses ändert (z.B. ++), wieder mit *putChar* schreibt und das File-Objekt mit *close* schliesst. Verwende in diesem Programm Exception-Handling und mache etwas sinnvolles mit den „gefangenen“ Fehlern. (6 Punkte)

```
#include <string>
#include <fstream>

using namespace std;

class File
{
    public:
        File(const char* filename)
        {
            _stream.open(filename);
            if(!_stream.is_open())
            {
                throw string("Datei öffnen misslungen");
            }
        }

        ~File()
        {
            if(_stream.is_open())
            {
                close();
            }
        }

        char getChar()
        {
            if(!_stream.is_open())
            {
                throw string("Datei ist nicht geöffnet");
            }

            char c = 0;
            _stream >> c;
            return c;
        }

        void putChar(char c)
        {
            if(!_stream.is_open())
            {
                throw string("Datei ist nicht geöffnet");
            }
            _stream << c;
        }

        void close()
        {
            _stream.close();
        }

    private:
        fstream _stream;
};

int main()
{
       //.......... ab hier Deinen Code einfügen
```

```
int main()
{
    try
    {
        File einFile(„test.txt“);
        char data = einFile.getChar();
        data++;
        einFile.putChar(data);
        einFile.close();
    }
    catch(std::string& error)
    {
        cout << „Ein Fehler ist aufgetreten : „;
        cout << error << endl;
    }
    return 0;
}
```

3. Programmverständnis

a) Was gibt folgendes Programm aus? Das grosse ‚A' hat den ASCII-Wert 65! (2 Punkte)

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

typedef std::vector<char> Chars;

int main()
{
    Chars someChars;
    for(int i = 65; i < 75; ++i)
    {
        someChars.push_back(i);
    }

    Chars::iterator it = someChars.begin();
    Chars::iterator end = someChars.end();

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

    return 0;
}
```

```
ABCDEFGHIJ
```

b) Gegeben ist die Klasse *Element* und der Typ *Elements*. Schreibe eine Funktion *PrintElements*, die ein *Elements*-Objekt als Argument übernimmt und für alle Elemente in der Liste die Methode *print* aufruft. Achte auch auf effizienten Code! (6 Punkte)

```
#include <list>
#include <iostream>

class Element
{
public:
    void print();
};

typedef std::list<Element> Elements;

// Funktion PrintElements
void PrintElements(Elements& elements)
{
    Elements::iterator it = elements.begin();
    Elements::iterator end = elements.end();

    for( ; it != end; ++it)
    {
        it->print();
    }
}
```

4. Weitere Fragen

a) Schreibe eine Funktion *maxInt*, die als Argument zwei integer-Werte nimmt und als Rückgabewert den grösseren der beiden zurückliefert. (2 Punkte)

```
int maxInt(int a, int b)
{
    if(a > b)
    {
        return a;
    }
    else
    {
        return b;
    }
}
```

b) Schreibe nun eine template-Funktion *max*, die man für beliebige Datentypen verwenden kann. (4 Punkte)

```
template<class Typ>
Typ max(Typ a, Typ b)
{
    if(a > b)
    {
        return a;
    }
    else
    {
        return b;
    }
}
```

c) Definiere eine Aufzählung (*enum)*, in der alle Wochentage einen Wert haben. (2 Punkte)

```
enum Wochentage
{
    Montag,
    Dienstag,
    Mittwoch,
    Donnerstag,
    Freitag,
    Samstag,
    Sonntag
};
```

d) Wie heisst die Anweisung, mit der man ein Programm in der „Debug"-Version zum Abstürzen bringen kann? (1 Punkt)

```
assert oder _ASSERT
```

e) Willst Du fehlerfreie Programme schreiben? (1 Punkt) **JA**