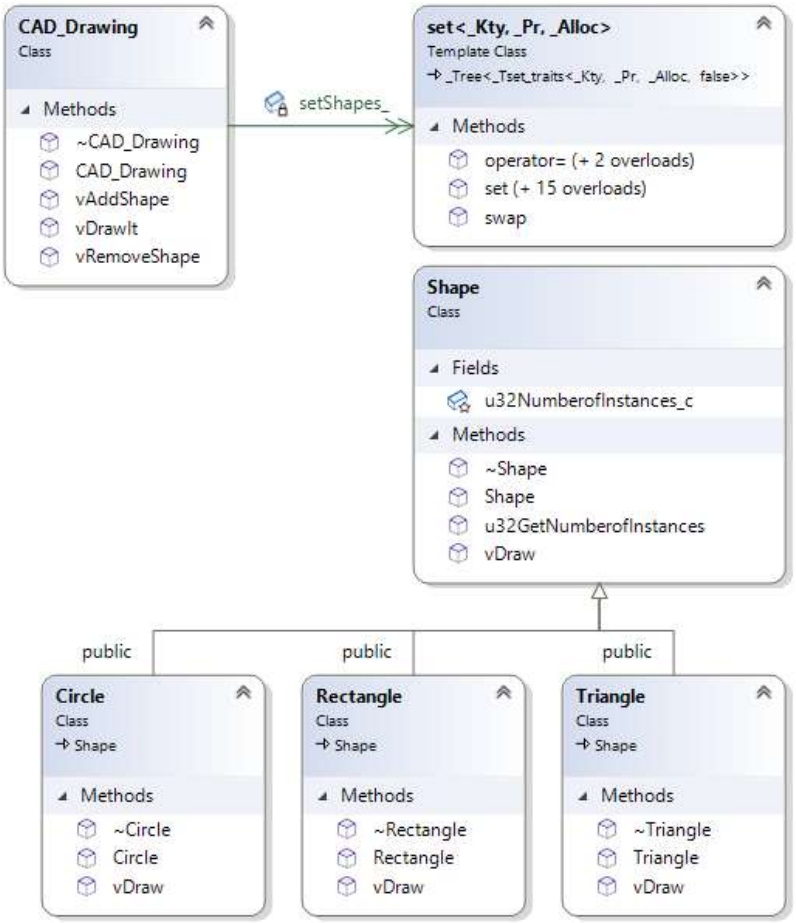
Übungen OOSWE/Progr. 2 (C++) – KE 6

Der C/C++-Coding Styleguide ist einzuhalten.
Folgende Einstellungen sind für Debug und Release (All Configurations) vorzunehmen:

Einstellung	**Wert**
Solution Platform	x86
Properties->Conf. Properties->C/C++->General->Warning Level	Level4 (/W4)
Properties->Conf. Properties->C/C++->General->Treat Warnings As Errors	Yes (/WX)
Properties->Conf. Properties->C/C++->General->SDL checks	Yes (/sdl)
Properties->Conf. Properties->C/C++->Code Generation->Basic Runtime Checks	Default
Properties->Conf. Properties->C/C++->Code Generation->Security Checks	Enable Security Checks (/GS)
Properties->Conf. Properties->C/C++->Language->C++ Language Standard	ISO C++ 20 Standard

Legen Sie sich eine Solution an, die alle Aufgaben als Projekte enthält.
Die geforderten Kommentare sind in Englisch zu hinterlegen.

Aufgabe 1: Dynamischer Polymorphismus
Es sollen Objekte für eine CAD-Zeichnung erstellt und verwaltet werden. Eine CAD-Zeichnung (Klasse CAD_Drawing) soll verschiedene Objekte enthalten (Triangle, Rectangle, Circle). Alle Klassen sollen sich im **namespace CAD** befinden.



1.1 Legen Sie ein neues Projekt an. Die Klassen Shape, Triangle, Rectangle und Circle sollen sich in separaten cpp- und h-Dateien befinden. Shape soll eine abstrakte Klasse mit einer pure virtual Funktion vDraw sowie ein protected **Klassenattribut** u32NumberofInstances_c enthalten. In deren Kon- und Destruktoren ist u32NumberofInstances_c zu in- oder dekrementieren. Die **Klassenmethode** u32GetNumberofInstances gibt den Wert der Anzahl der generierten Instanzen zurück. Die überschriebene vDraw-Methode soll nur einen Text ausgeben (z.B. „Draw Circle“).

1.2 Implementieren Sie die Klasse CAD_Drawing.
Die Klasse enthält als Collection ein **set**, in welchem die Zeiger auf die erstellten Shape-Objekte gespeichert werden. Ein **set** bietet gegenüber einer **list** oder einem **vector** die folgenden Vorteile:

- Die Elemente sind unique. Daher ist es nicht möglich, einen Objektzeiger ein zweites Mal im Set abzuspeichern.
- Das Hinzufügen (in vAddShape) und das Löschen eines Pointers (in vRemoveShape) geschieht nur über jeweils eine Methode von set (Frag Google).

```
#pragma once
#include <set>
#include "Shape.h"
#include "Triangle.h"
#include "Circle.h"
#include "Rectangle.h"

namespace CAD
{
   class CAD_Drawing
   {
   private:
      std::set<Shape*> setShapes_;

   public:
      CAD_Drawing(void);
      ~CAD_Drawing(void);
      void vAddShape(Shape*);
      void vRemoveShape(Shape*);
      void vDrawIt(void);
   };
}
```

Die vDrawIt-Methode soll die Draw-Funktion aller im Set referenzierten Objekte aufrufen. Hierzu bietet sich eine auto-ranged Schleife an. Dabei stellt it den Iterator dar!

```
void CAD_Drawing::vDrawIt(void)
{
   for (auto it : setShapes)
   {
      it->Draw();
   }
}
```

1.3 Rufen Sie in main eine Methode vTest_CAD_Drawing auf. Ebenso ist eine Überwachung auf Memory Leaks notwendig.

In vTest_CAD_Drawing sind verschiedene Shape-Objekte statisch und dynamisch zu instanziieren und der Objektzeiger ist an CAD_Drawing zu übergeben.

Ebenso können Zeiger wieder aus dem Set entfernt werden. Die Methode vDrawIt ist immer wieder aufzurufen.

Verifizieren Sie durch Tests:

- Stimmt nach dem Löschen von Objekten uiNumberofInstances_c?
- Ist es möglich einen Objektzeiger zweimal im Set von CAD_Drawing abzuspeichern?
- Gibt es Memory Leaks?
- Was passiert, wenn ein dynamisch generiertes Objekt gelöscht wird und sich der Zeiger noch im Set befindet?

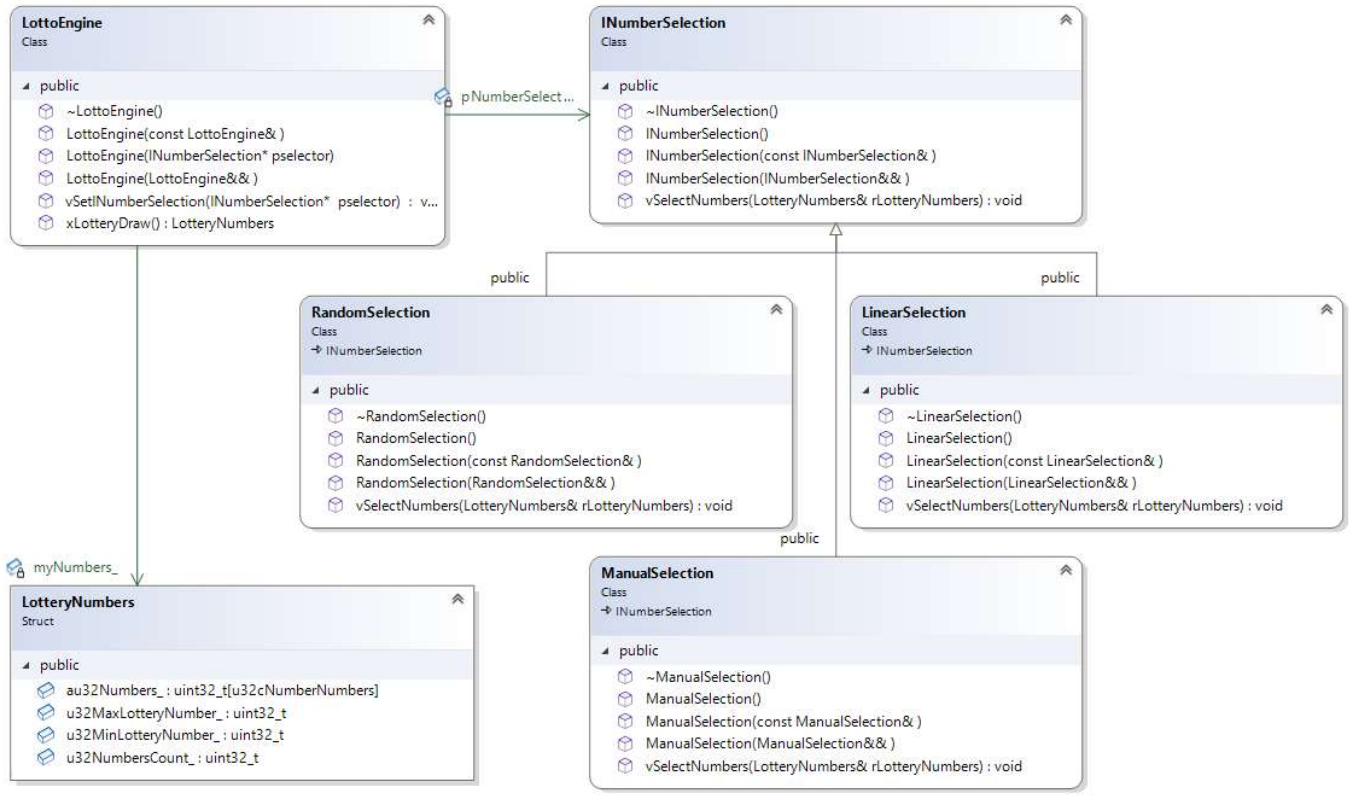
```
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#include <iostream>
#include "CAD_Drawing.h"

void vTest_CAD_Drawing(void);

int main(void)
{
   vTest_CAD_Drawing();

   _CrtDumpMemoryLeaks();
   return 0;
}
```

Aufgabe 2: Strategy-Pattern (Dynamischer Polymorphismus)
Es soll eine Klasse LottoEngine implementiert werden, welche eine Lottoziehung (xLotteryDraw) durchführt und sechs Lottozahlen bestimmt. Das „x“ steht für einen benutzerdefinierten Datentyp (hier die Struktur LotteryNumbers) Die folgenden Klassen und die Struktur befinden sich im **namespace Lotto**.



Die Ziehung der Lottozahlen (xLotteryDraw) soll dabei mittels dreier unterschiedlicher Algorithmen (…Selection) realisiert werden. Zentrales Element bildet das Interface **INumberSelection** welches eine pure virtual Funktion vSelectNumbers enthält. In C++ gibt es kein Schlüsselwort interface wie in Java oder C#. Ein Interface lässt sich in C++ als Klasse mit nur reinen virtuellen Methoden realisieren. Um eine solche Klasse als Interface zu markieren, wird dem Klassennamen ein „I“ vorangestellt.

2.1. Deklarieren und definieren Sie INumberSelection, LotteryNumbers, RandomSelection, ManualSelection und LinearSelection in einer h.- und einer .c-Datei (Einfachheit):

```
class INumberSelection
{
   public:
      INumberSelection() = default;
      INumberSelection(const INumberSelection&) = default;
      INumberSelection(INumberSelection&&) = default;
      virtual ~INumberSelection() = default;

      virtual void vSelectNumbers(LotteryNumbers& rLotteryNumbers) = 0;
   };
```

Die Funktion vSelectNumbers führt dabei die Ziehung durch, indem diese die übergebene Struktur (Referenz) rLotteryNumbers beschreibt. LotteryNumbers enthält ein Array mit u32cNumberNumbers (Konstante) Elementen.

```
const uint32_t u32cNumberNumbers = 6U;

   struct LotteryNumbers
   {
   public:
      uint32_t u32MinLotteryNumber_ = 1U;
      uint32_t u32MaxLotteryNumber_ = 49U;
      uint32_t u32NumbersCount_ = u32cNumberNumbers;
      uint32_t au32Numbers_[u32cNumberNumbers] = { 0 };
   };
```

Tipps zur Realisierung der virtuellen Funktion vSelectNumbers:

- RandomSelection: Achten Sie darauf, dass srand nur einmal aufgerufen wird. Es muss ebenso verhindert werden, dass eine Zahl zweimal gezogen wird. Hierzu könnte ein set verwendet werden. Die folgende Abfrage ergibt true, wenn u32Number noch nicht im set enthalten ist.
  `if (SetSelectedNumbers.find(u32Number) == SetSelectedNumbers.end()`
  Da ein set Elemente nur einmal enthält, ist die Verwendung der Funktion size noch einfacher.
- ManualSelection: Bei der manuellen Eingabe ist auch zu berücksichtigen, dass keine Zahl zweimal „gezogen" werden kann.
- LinearSelection: Hier sollen beginnend mit eins immer die folgenden Zahlen verwendet werden: Erste Ziehung: (1,2,3,4,5,6), zweite Ziehung (7,8,9,10,11,12) usw.
  Der Überlauf bei 49 ist zu berücksichtigen.

2.2 Realisieren Sie die Klasse LottoEngine in LottoEngine.h und LottoEngine.cpp.

```
class LottoEngine
{
private:
   INumberSelection* pNumberSelection_;    // selection strategy
   LotteryNumbers myNumbers_;

public:
   LottoEngine(INumberSelection* pselector);
   LottoEngine(const LottoEngine&) = delete;
   LottoEngine(LottoEngine&&) = delete;
   ~LottoEngine() = default;

   void vSetINumberSelection(INumberSelection* pselector);
   LotteryNumbers xDrawingLotteryNumbers(void);
   };
```

Mittels Constructor Injection oder einer Set-Funktion kann der Zeiger pNumberSelection_ auf ein Objekt gesetzt werden, welches das Interface INumberSelection realisiert.

2.3 Instanziieren Sie in main statisch drei Selection-Objekte (RandomSelection, ManuaSelection und LinearSelection). Instanziieren Sie **eine** LottoEngine, welche die Adresse des RandomSelection-Objektes erhält. Führen Sie mehrere Lottoziehungen (DrawingLotteryNumbers) durch. Das Ergebnis einer Lottoziehung sollten Sie in einer lokalen LotteryNumbers-Struktur speichern.

Realisieren Sie noch in Main.cpp eine statische Funktion, welche die LottersNumbers-Struktur ausgibt.

```
static void vPrintLotteryNumbers(Lotto::LotteryNumbers ln);
```

Weisen Sie danach der LottoEngine die anderen Selection-Objekte zu und führen Sie weitere Ziehungen durch. Verifizieren Sie anhand von vPrintLotteryNumbers die Korrektheit der Implementierung.

Zusammenfassend:
Die LottoEngine hat keine Informationen darüber, welche Strategie bei der Lottoziehung verwendet wird. Der Algorithmus ist über eine Setter-Funktion austauschbar.

Am Ende der Lehrveranstaltung werden Sie lernen, dass es sogenannte **Design Patterns** gibt. Dies sind vorgegebene Schablonen, welche eine objektorientierte Lösung für gängige Problemstellungen liefern. In dieser Aufgabe haben Sie Ihr erstes Design Pattern - das **Strategy-Pattern** - realisiert.