Klausur 1-B 2022

Aufgaben

1. **Solution und Projekt unter Visual Studio anlegen:**

    a. Erstellen Sie unter Visual Studio eine Solution mit dem Namen **OOP_KL_Nachname.Vorname**

    b. Erstellen sie zwei Projekte in dieser Solution, **Grundsl** und **Diverses**.

    c. Trennen Sie Ihre Klassen sinvoll in .cpp und .h Files.

2. Grundsl

a. Erstellen Sie eine Klassenstruktur zur Verwaltung von Lebensmittel. Die Elternklasse soll hierbei **Produce** mit den Membern ***price*** (*double*) und ***vitamins*** (*vector<int>*) sein, der Zugriff auf diese Member soll nur **geregelt** möglich sein. Von dieser Klasse sollen die Klassen **Fruit** und **Vegetables** erben. **(20P)**

b. Erweitern Sie nun die **Produce** Klasse insofern, das Lebensmittel mit dem **+ operator** addiert werden können. Hierbei wird ***price*** addiert und die ***vitamins*** Vektoren zusammengeführt. **(10P)**

Produce (10.0, [1,2,3]) + Fruit (5.5, [4,5]) = Produce (15.5, [1,2,3,4,5])

c. Ermöglichen Sie die Zuweisung **ODER** CopyKonstruktion unterschiedlicher Produce Objekte und derer Ableitungen. Bei der Zuweisung/CopyKonstruktion soll **OBJECT ASSIGN/*OBJECT COPY*** auf die Konsole ausgegeben werden. Ermöglichen Sie den folgenden Code **(10P)**:

```
Produce p1{ 10, std::vector<int>{1,2,3} };
Fruit f1{ 5, std::vector<int>{1,2,3,4,5} };
Vegetables v1{ 10, std::vector<int>{7,8,9} };
Produce p2 = f1 + p1;
Fruit f2 = f1 + v1;
Vegetables v2 = p1 + f2;
```

d. Erstellen Sie die Funktion **GetNutrition** welche spät gebunden den Nährstoffwert unterschiedlicher Lebensmittel auf die Konsole ausgibt. In der **Produce** Klasse soll dieser Wert ***gleich*** der Anzahl der ***vitamins*** Elemente sein, in der Klasse **Fruit** die ***halbe*** Anzahl der Elemente und in der Klasse **Vegetables** die ***doppelte*** Anzahl an Elementen. **(15P)**

Produce (10.0, [1,2,3]).getNutritionValue() → "Produce nutrition: 3"
Fruit (5.5, [4,5,6]).getNutritionValue() → "Fruit nutrition: 1"
Vegetable (15.5, [1,2,3,4,5]).getNutritionValue() → "Vegetables nutrition: 10"

e. Erstellen Sie einen **Templatecontainer** welcher alle Objekte der erstellten Klassenstruktur verwalten kann und befüllen Sie diesen mit 10 zufälligen Objekten (**Produce** und **Kindklassen**). Iterrieren Sie über diesen Container und rufen Sie die Funktion **GetNutrition()** jedes Objektes auf. **(5P)**

Erstellen einer Zufallszahl (0-2) in Cpp (random inkludieren!):

```
std::random_device dev;
std::mt19937 rng(dev());
std::uniform_int_distribution<std::mt19937::result_type> dist3(0, 2);
std::cout << dist3(rng) << std::endl;
```

```
for (int i = 0; i < 10; i++) {
   if(dist3(rng) == 0) {
          // TODO: Add produce elem
   }
   else if(dist3(rng) == 1) {
          // TODO: Add child1 elem
   }
   else {
          // TODO: Add child2 elem
   }
}
```

f. Testen Sie den Code ausreichend indem Sie jede implementierte Funktion mindestens einmal verwenden. **(5P)**

3. Diverses

a. Lambda und Algorithm

Für dieses Beispiel muss eine Funktion der **algorithm** Bibliothek verwendet werden: https://en.cppreference.com/w/cpp/algorithm. Finden sie alle **großgeschriebenen Buchstaben chars (a-z)** eines übergebenen **Strings** und wandeln Sie diese in kleingeschriebene Buchstaben über ein **Lambda** um. Geben Sie schließlich alle Elemente des Containers ***außerhalb*** des **Lambdas** und der **algorithm** Funktion aus. **(15P)**

"a3C2fE" → "a3c2fe"

a. Erstellen Sie ein Interface (abstrakte Klasse!) **myInterface** welches die Funktion **GetElement** mit einem ***vector<beliebigerTyp>*** als **Übergabeparameter** und einen ***beliebigen Typ*** als **Rückgabewert** für sämtliche Kindklassen zwingend vorschreibt. Der ***beliebige*** Typ soll für das gesamte Interface gelten. Als Test, erstellen Sie eine Kindklasse und rufen Sie die implementierte Funktion auf. **(10P)**

```
class test : myInterface<int> {
public:
	// ...
};
```

b. Erstellen Sie eine von std::exception erbende **Exception** (myException) welche beim Aufruf der **what()** Funktion *"Exception occured: Sum of Doubles to high"* zurückgibt. Erzeugen Sie ein **double array** und werfen und behandeln Sie eine solche **Exception** wenn die Summe der double Werte des arrays **>=50** ist. **(10P)**

Nachbearbeitung

- Klicken Sie unter Visual Studio mit der **rechten Maustaste im Solution Explorer** auf den Solutionname (erste Zeile).
  Wählen Sie den Menüpunkt **Clean Solution** aus. Damit sollen (fast) alle Compilate und temporären Dateien gelöscht sein.
- Löschen Sie das .vs Verzeichnis (Achtung: hidden folder!) im Verzeichnis der Solution, bevor Sie das **zip**-File der Solution erstellen.
- Geben Sie das **zip** File in der bereitgestellten **Moodle** Abgabe ab.