technische universität dortmund

fakultät für informatik

Jurij Kuzmic
Horst Schirmeier
Jonas Ellert

Dortmund, den 28. Januar 2021

Übung zur Vorlesung EidP (WS 2020/21)

Blatt 10

Block grün

Es können 4 Punkte erreicht werden.

Abgabedatum: 4. Februar 2021, 23:59 Uhr

Hinweise

- Bitte beachten Sie aktuelle Hinweise unter:

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep2021uebung/`

- Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
- Verwenden Sie **keine zusätzlichen Bibliotheken** zur Lösung der Aufgaben.
- Für die Programmieraufgaben kopieren Sie immer die Ausgabe als Block-Kommentar an das Ende der Datei, welche das jeweilige Hauptprogramm enthält.
- Sie sollten während der Entwicklung Ihrer Programme diese unbedingt regelmäßig – und insbesondere noch einmal vor der Abgabe – **compilieren und ausführen**.

Aufgaben

Aufgabe 1: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei `Aufgabe_10_1.txt` an.

a) Was versteht man unter Polymorphie? (0.2 Punkte)

b) Was gilt in Bezug auf Attribute und Methoden, wenn `A` eine Oberklasse von `B` ist? (0.2 Punkte)

c) Wie werden virtuelle Methoden gekennzeichnet und wann werden sie gebunden? (0.2 Punkte)

d) Wie werden rein virtuelle Methoden deklariert? (0.1 Punkte)

e) Wann nennt man eine Klasse abstrakt? (0.1 Punkte)

f) Wozu werden abstrakte Klassen eingesetzt? (0.2 Punkte)

Aufgabe 2: Vererbung (3 Punkte)

Legen Sie für diese Aufgabe die Dateien `Aufgabe_10_2.h`, `Aufgabe_10_2.cpp` und `Aufgabe_10_2_test.cpp` an.

a) Erstellen Sie ein Programm für die Speicherung von Fahrzeugdaten. Für die unterschiedlichen Fahrzeugarten (PKWs, Motorräder und LKWs) sind jeweils unterschiedliche Daten zu erfassen:

1. Für alle Fahrzeugtypen: Kennzeichen, Erstzulassung [Jahr], Hubraum [ccm]
2. PKWs, zusätzlich: Leistung [kW], Schadstoffklasse [Euro 1-6]
3. Motorräder, zusätzlich: Beiwagen [Ja/Nein]
4. LKWs, zusätzlich: Achsen [Anzahl], Zuladung [t]

Erstellen Sie geeignete `C++`-Klassen, um die oben aufgelisteten Daten **ohne Redundanz** abzuspeichern. Wählen Sie sinnvolle Datentypen für die jeweiligen Attribute. Beachten Sie dabei, dass von der Basisklasse **keine Objekte** erzeugt werden sollen (abstrakte Basisklasse).

(1 Punkt)

b) Erweitern Sie Ihr Programm um die Delegation an die Basisklassenkonstruktoren und die Initialisierungsliste, wo immer dies möglich ist. Schreiben Sie zusätzlich eine Methode `void print()` zur Ausgabe der gespeicherten Daten für die jeweiligen Klassen. Jede dieser `print()`-Methoden muss in der jeweiligen Klasse überschrieben werden. Diese gibt dann nur die in der Klasse gespeicherten Attribute aus. (1 Punkt)

c) Schreiben Sie ein Hauptprogramm `Aufgabe_10_2_test.cpp` und legen Sie pro Fahrzeug jeweils zwei Testobjekte an. Geben Sie anschließend die gespeicherten Daten mit `print()` aus. Folgendes Beispiel muss nach Ihrer Implementierung möglich sein:

```
1 PKW vw("MK - JK 1111", 2006, 1980, 130, 2);
2 Fahrzeug *seat = new PKW(" K - KJ 1284", 2014, 1990, 150, 5);
3 vw.print();
4 seat->print();
```

(1 Punkt)

Hinweis: Unterteilen Sie die Deklaration und die Definition der jeweiligen Klassen in die Dateien `Aufgabe_10_2.h` und `Aufgabe_10_2.cpp`. Handeln Sie ebenfalls nach dem „*Information Hiding*“-Prinzip.

Präsenzaufgabe 3: Ausnahmebehandlung (0 Punkte)

In dem unten angegebenen Programm wird in der Funktion `Formel` die Summe zweier Zahlen x und y durch ihre Quadratwurzel geteilt. Folglich darf die Summe weder Null noch negativ sein, da dann aus unterschiedlichen Gründen Fehler auftreten können. Falls ein Fehler auftritt, enthält die Variable `fehler` einen positiven Fehlercode, der nach jedem Aufruf abgeprüft werden muss.

```
1  /*** Aufgabe_10_3.cpp ***/
2  #include <iostream>
3  #include <cmath>
4  using namespace std;
5
6  double Formel(double x, double y, int &fehler) {
7    double sum = x + y;
8    if (sum < 0) {
9      fehler = 1;
10   } else if (sum == 0) {
11     fehler = 2;
12   } else {
13     fehler = 0;
14     return sum / sqrt(sum);
15   }
16   return 0;
17 }
18
19 int main() {
20   int fehler;
21   double x[3] = { -2, -4, 0 };
22   double y[5] = { 1, 2, 0, 3, 1 };
23   for (unsigned int i = 0; i < 3; ++i) {
24     for (unsigned int j = 0; j < 5; ++j){
25       double z = Formel(x[i], y[j], fehler);
26       switch (fehler) {
27       case 1:
28         cerr << "negative Summe" << endl; break;
29       case 2:
30         cerr << "Division durch Null" << endl; break;
31       default:
32         cout << z << endl;
33       }
34     }
35   }
36   return 0;
37 }
38 /*** Ende Aufgabe_10_3.cpp ***/
```

Schreiben Sie die Funktion und das Hauptprogramm um, indem Sie den Mechanismus der C++-Ausnahmebehandlung mit `try`, `throw` und `catch` verwenden. Die Ausgabe des Programms soll sich nicht ändern. Verwenden Sie für jede Fehlerart eine eigene Fehlerklasse und einen eigenen `catch`-Handler.