Objektorientierte Systeme 1 - SWB2 & TIB2
Hausaufgabe 4

Die Aufgaben, die mit (Exkurs) markiert sind, beinhalten Stoff aus den Exkursen der Foliensätze zur Vorlesung und wurden daher evtl. in der Vorlesung nicht besprochen.

Aufgabe 1: Einfache Vererbung

Schreiben Sie ein Programm mit den folgenden Eigenschaften.

a) Die Klasse `Haushaltsgeraet` hat eine konstante Instanzvariable `name` vom Typ `string` und eine Instanzvariable `gewicht` vom Typ **`double`**. Sie geben die Bezeichnung bzw. das Gewicht eines Haushaltsgerätes an.

b) Die Klasse `Haushaltsgeraet` hat einen parametrisierten Konstruktor, um die Instanzvariablen zu initialisieren, sowie Instanzmethoden, um die Instanzvariablen auszulesen bzw. wo möglich zu setzen (`getXXX`, `setXXX`).

c) Die Klasse hat zusätzlich eine Instanzmethode `print()`, um die Instanzvariablen eines Objektes auszugeben. Die Ausgabe kann folgendermaßen aussehen:

```
Geraetename = Haushaltsgeraet Nr 1
Gewicht     = 7.1 kg
```

d) Leiten Sie von der Klasse `Haushaltsgeraet` die Klasse `Gefrierschrank` ab.

e) Die Klasse `Gefrierschrank` hat die Instanzvariable `volumen` vom Typ **`double`**. Schreiben Sie einen parametrisierten Konstruktor, der den Konstruktor der Klasse `Haushaltsgeraet` aufruft sowie passende `set`- und `get`-Methoden.

f) Schreiben Sie eine Methode `print()`, die alle Instanzvariablen eines Gefrierschranks ausgibt. Rufen Sie dazu die `print`-Methode der Klasse `Haushaltsgeraet` auf. Die Ausgabe könnte so aussehen:

```
Geraetename = Gefrierschrank Nr 1
Gewicht     = 21.5 kg
Volumen     = 101 L
```

g) Schreiben Sie eine Hauptfunktion `main`, in der Sie
   (i) ein Haushaltsgerät erstellen und dieses mit `print` auf der Konsole ausgeben,
   (ii) einen Gefrierschrank erstellen und diesen mit `print` auf der Konsole ausgeben,
   (iii) das Gewicht und das Volumen des Gefrierschranks ändern und dann das Objekt wieder auf der Konsole ausgeben.

Aufgabe 2: Vererbung mit Zugriffsmodifizierer (Exkurs)

Die Spezialisierung, die durch eine Klassenhierarchie dargestellt wird, zeichnet sich häufig dadurch aus, dass zu den Attributen der Basisklasse in der abgeleiteten Klasse weitere Attribute hinzukommen, weil die Objekte der abgeleiteten Klasse zusätzliche Eigenschaften besitzen. Das Beispiel vom Gefrierschrank, der als Eigenschaft zusätzlich zu den Eigenschaften eines Haushaltsgeräts ein Volumen hat, haben wir oben gesehen.

Häufig besteht die Spezialisierung aber auch darin, dass der sinnvolle Wertebereich einer geerbten Instanzvariable im Gegensatz zur Basisklasse eingeschränkt ist, es aber gar keine zusätzlichen Eigenschaften gibt. Ein Beispiel ist in der Klassenhierarchie grafischer Objekte zu finden:

Ein Rechteck ist ein Parallelogramm. Die Seiten eines Rechtecks stoßen in einem Winkel von 90 Grad aufeinander während der Winkel in einem Parallelogramm alle positiven Werte unter 180 Grad annehmen kann.

Programmieren Sie die Klasse `Parallelogramm` mit den folgenden Eigenschaften.

a) Die Instanzvariablen `a` und `b` stehen für die Seitenlängen und die Variable `winkel` für einen der Winkel im Parallelogramm. Alle drei Variablen sollen vom Typ **`double`** sein.
b) Für jede Instanzvariable gibt es `set`- und `get`-Methoden, um die Variablen zu setzen bzw. auszulesen.
c) Eine Methode `print()` gibt die Instanzvariablen aus, z.B. so:

```
Seite a = 10.00   Seite b = 20.00   Winkel = 45.00
```

Leiten Sie nun die Klasse `Rechteck` von der Klasse `Parallelogramm` ab. Statt des Zugriffsmodifizierers **`public`** nutzen Sie nun **`protected`**. Dadurch sind bei einem Rechteck die von `Parallelogramm` geerbten Elemente in der Klasse zugreifbar, außerhalb jedoch nicht.

Ergänzen Sie die Klassen, so dass das folgende Hauptprogramm fehlerfrei ausgeführt werden kann.

```
int main() {
    Parallelogramm p(10,20,45);
    Rechteck r(40,50);
    p.print();
    p.setA(15);
    p.setwinkel(50);
    r.print();
    r.setA(45);
    if (r.getWinkel() != 90.0) {
        cout << "Das ist doch kein Rechteck!" << endl;
    }
    r.print();
    return 0;
}
```

Aufgabe 3: Mehrfache Vererbung I - Mehrfaches Erbgut

In C++ ist es möglich, dass eine Klasse mehrere Basisklassen hat. Schreiben Sie ein Programm mit den folgenden Eigenschaften.

a) Die Klasse `Elektrogeraet` hat eine Instanzvariable `watt` vom Typ **`int`**. Sie gibt die Leistungsaufnahme des Geräts an.
b) Die Klasse `Elektrogeraet` hat einen parametrisierten Konstruktor, um die Instanzvariable zu initialisieren, sowie Instanzmethoden, um die Instanzvariable auszulesen bzw. zu setzen (`getWatt`, `setWatt`).
c) Die Klasse hat zusätzlich eine Instanzmethode `print()`, um die Instanzvariable auszugeben. Die Ausgabe kann folgendermaßen aussehen:

```
Leistungsaufnahme: 300 W
```

d) Leiten Sie von der Klasse `Elektrogeraet` die Klassen `Akkuschrauber` und `Akkubohrer` ab. Die Klassen haben keine weiteren Instanzvariablen aber jeweils einen parametrisierten Konstruktor, mit dem die Instanzvariable `watt` initialisiert werden kann.

e) Leiten Sie von den Klassen `Akkuschrauber` und `Akkubohrer` die Klasse `Akkubohrschrauber` ab.

f) Die Klasse `Akkubohrschrauber` hat keine zusätzlichen Instanzvariablen. Schreiben Sie aber einen parametrisierten Konstruktor, der die Konstruktoren der beiden Basisklassen aufruft.

g) Schreiben Sie eine Methode `print()`, die alle Instanzvariablen eines Akkubohrschraubers ausgibt. Rufen Sie dazu die `print`-Methode der Klasse `Elektrogeraet` auf. Die Ausgabe könnte so aussehen:

```
Leistungsaufnahme als Schrauber: 900 W
Leistungsaufnahme als Bohrer:    400 W
```

h) Testen Sie Ihre Klassen mit dem folgenden Hauptprogramm.

```
int main() {
    Akkubohrer b(300);
    b.print();
    Akkuschrauber s(500);
    s.print();
    Akkubohrschrauber bs(400,900);
    bs.print();
    return 0;
}
```

Aufgabe 4: Mehrfache Vererbung II - Virtuelle Basisklassen

Erweitern Sie Ihr Programm aus Aufgabe 1 um eine Klasse `Kuehlschrank`, welche die gleichen Eigenschaften wie die Klasse `Gefrierschrank` hat.

Leiten Sie dann die Klasse `Kombination` (Kuehl-Gefrierkombination) von `Kuehlschrank` und `Gefrierschrank` ab. Stellen Sie sicher, dass die Klasse `Kombination` die Eigenschaften von Klasse `Haushaltsgeraet` nur einmalig erbt.

Testen Sie Ihr Programm mit dem folgenden Hauptprogramm, das die gegebene Ausgabe erzeugen soll.

```
int main() {
    Kombination k("Kombination", 23.5, 25, 10);
    k.print();
    return 0;
}
```

Ausgabe:

```
Geraetename = Kombination
  Gewicht        = 23.5 kg
  Gefriervolumen = 25 L
  Kuehlvolumen   = 10 L
```

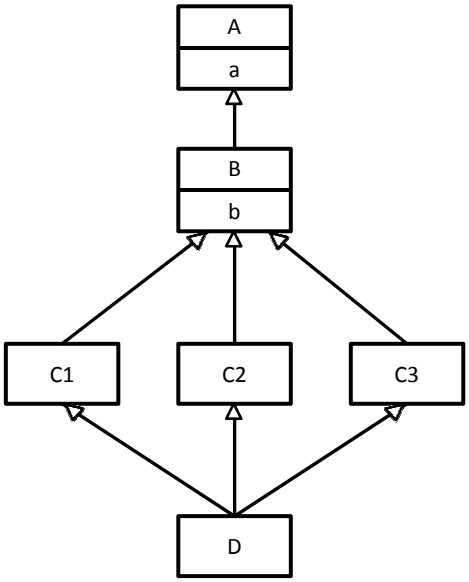
Aufgabe 5: Mehrfache Vererbung III - Anzahl der geerbten Instanzvariablen

Die Vererbungshierarchie in der Abbildung ist gegeben. Die Klasse `A` hat die Instanzvariable `a` und die Klasse `B` hat die Instanzvariable `b`. Die Klassen `C1`, `C2`, `C3` und `D` haben keine eigenen Instanzvariablen.

5.1 Geben Sie an, wie viele Instanzvariablen (eigene und geerbte) jede Klasse besitzt. Geben Sie an, welche Konstruktoren aufgerufen werden, wenn ein Objekt der Klasse `D` erzeugt wird.

5.2 Wählen Sie sich nun eine beliebige Menge der Vererbungsbeziehungen (Elternklasse - Kindklasse) in der Abbildung aus und nehmen Sie an, dass die jeweilige Elternklasse eine virtuelle Basisklasse ist. Beantworten Sie dann die Fragen aus 5.1.

5.3 Führen Sie Punkt 5.2 für jede mögliche Kombination von virtuellen Basisklassen in der Abbildung durch.



Aufgabe 6: Mehrfache Vererbung und Konstruktoren

Diese Aufgabe zeigt die Besonderheiten von Konstruktoren bei der Mehrfachvererbung. Kompilieren Sie das Programm und erklären Sie die Ausgabe. Sie werden sehen, dass eine Variable nicht initialisiert wird. Ändern Sie das Programm so ab, dass die Initialisierung vollständig durchgeführt wird.

```
#include <iostream>
using namespace std;

class A {
    int alpha;
public:
    A(){
        cout << "Defaultkonstruktor von Klasse A" << endl;
    };
    A(int alpha): alpha(alpha){
        cout << "Konstruktor von Klasse A" << endl;
    }
    void print(){
        cout << "A - print:" << endl;
        cout << "A::alpha= " << alpha << endl; }
};

class B : virtual public A {
    int beta;
public:
    B(int alpha, int beta):A(alpha), beta(beta){
        cout << "Konstruktor von Klasse B" << endl;
    }
    void print(){
        cout << "B - print:" << endl;
        A::print();
        cout << "B::beta= " << beta << endl; }
```

```
};

class C : virtual public A {
    int gamma;
public:
    C(int alpha, int gamma):A(alpha), gamma(gamma){
        cout << "Konstruktor von Klasse C" << endl;
    }
    void print() {
        cout << "C - print:" << endl;
        A::print();
        cout << "C::gamma= " << gamma << endl; }
};

class D : public B, public C {
    int delta;
public:
    D(int alpha, int beta, int gamma, int delta)
        : B(alpha, beta), C(alpha, gamma), delta(delta){
        cout << "Konstruktor von Klasse D" << endl;
    }
    void print() {
        cout << "D - print:" << endl;
        B::print();
        C::print();
        cout << "D::delta= " << delta << endl; }
};

void main() {
    cout << "Deklaration von a ..." << endl;
    A a(1);
    cout << "Deklaration von b ..." << endl;
    B b(2,3);
    cout << "Deklaration von c ..." << endl;
    C c(4,5);
    cout << "Deklaration von d ..." << endl;
    D d(11,12,13,14);
    A *v[4] = {&a, &b, &c, &d};
    cout << "Ausgabe von a.print():" << endl;
    a.print();
    cout << "Ausgabe von b.print():" << endl;
    b.print();
    cout << "Ausgabe von c.print():" << endl;
    c.print();
    cout << "Ausgabe von d.print():" << endl;
    d.print();
    cout << "Ausgabe von d.A::print():" << endl;
    d.A::print();
```

```
    cout << "Ausgabe mit erster Schleife" << endl;
    for (int i=0; i < 4; i++)
        v[i]->print();
    C *w[2] = {&c, &d};
    cout << "Ausgabe mit zweiter Schleife" << endl;
    for (int i=0; i < 2; i++)
        w[i]->print();
    cout << "Ausgabe mit Konvertierung" << endl;
    ((D*)w[1])->print();
}
```

Aufgabe 7: Klassenhierarchie für Fahrzeuge

Implementieren Sie eine sinnvolle Klassenhierarchie für die in den Abbildungen gezeigten Fahrzeuge. Führen Sie sinnvolle Instanzvariablen ein. Sinnvolle Instanzvariablen sind z.B. Länge, Breite, Höhe, Motorleistung, Höchstgeschwindigkeit, Tiefgang, Segelfläche, ... Nutzen Sie die Vorteile des objektorientierten Entwurfs und machen Sie dadurch die Klassenhierarchie effizient (d.h. Mehrfachprogrammierung wird unnötig).

Abbildung 1: Maybach Excelero

Abbildung 2: Motoryacht und Segelyacht mit Motor

Abbildung 3: VW Schwimmwagen Typ 166

Abbildung 4: Segelboot Optimist

Abbildung 5: Eissegler

Abbildung 6: Chevy Pick Up

Abbildung 7: Einrad

Abbildung 8: Tandem

Abbildung 9: Ruderboot

Abbildung 10: Draisine