Version C++ 23

Übungsblatt 3

Objektorientierung

Aufgabe 1 - Operator Überladung

Gegeben sei folgende Vektorklasse:

```
class Vektor3d
{
  public:
  Vektor3d(double _x, double _y, double _z): x{_x}, y{_y}, z{_z}{}
  Vektor3d() = default;

  private:
  double x {};
  double y {};
  double z {};

};
```

Erweitern Sie diese Klasse so, dass folgender Quelltext ausführbar wird:

```
Vektor3d vec1 {3.1, 1.5, 0.2};
Vektor3d vec2 {1.0, 0.5, 0.2};

if(vec1 == vec2)
  std::println("Die Vektoren sind gleich");
else
  std::println("Die Vektoren sind ungleich");

// Ausgabe: "Die Vektoren sind ungleich"
```

Aufgabe 2 - Identität von Objekten

1. Sie haben zwei Objekte instanziiert: **realElvis** und **fakeElvis**. Die Objekte **realElvis** und **fakeElvis** haben den gleichen Namen (siehe Abbildung 1).

   Wie können Sie in C++ dennoch herausfinden, dass beide Personen unterschiedliche Objekte sind, also eine andere Identität haben? Schreiben Sie ein Stück C++ Quelltext, welches diese Überprüfung durchführt.

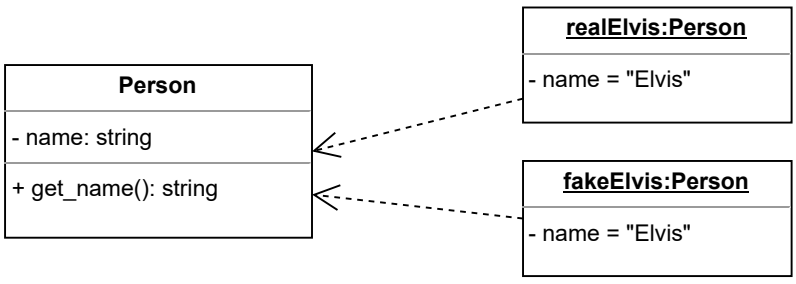

Abbildung 1: Exemplare von zwei Personen.

2. Erweitern Sie die Klasse Person so, dass diese Klasse eine Methode mit folgender Deklaration erhält:

```
bool ist(const Person& person);
```

Diese Methode gibt *true* zurück, wenn es sich bei der anderen Person um dieselbe Person handelt. "Implementieren" Sie die Methode.

Aufgabe 3 - Polymorphie & Vererbung

1. Analysieren Sie den Quelltext in Listing 1. Beantworten Sie hierzu folgende Fragen:
   a) Welche Ausgabe liefert der Quelltext?
   b) Welche Form von Polymorphie findet hier statt? dynamische Polymorphie
   c) Erstellen Sie ein UML Diagramm auf Basis des Quelltextes. Nö

2. Wenn Sie diesen Quelltext um ein weiteres Fortbewegungsmittel erweitern möchten, so dass die Main-Funktion diesen Quelltext ausführen kann:

```
int main()
{

  Flugzeug f;
  Auto a;
  Longboard l;
  Fahrrad r;

  Fortbewegung::sich_fortbewegen(f);
  Fortbewegung::sich_fortbewegen(a);
  Fortbewegung::sich_fortbewegen(l);
  Fortbewegung::sich_fortbewegen(r);

  return 0;
}
```

muss der Quelltext aus Listing 1 an mehreren Stellen angepasst werden. Diese Vorgehensweise ist in der Softwareentwicklung nicht optimal, da sich leicht Fehler einschleichen können.

a) Überarbeiten Sie den Quelltext aus Listing 1 so, dass bei Erweiterung um ein neues Fortbewegungsmittel lediglich eine neue Klasse eines Fortbewegungsmittels (z.B. Fahrrad) hinzugefügt werden muss.

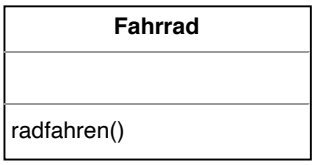

Abbildung 2: Die Klasse Fahrrad

b) Überarbeiten Sie auch das UML-Diagramm. Erweitern Sie dieses um beliebige Fortbewegungsmittel.

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

class Flugzeug
{
  public:
  void fliegen(){
    std::println("Ich fliege!");
  }
};

class Auto
{
  public:
  void fahren() {
    std::println("Ich fahre!");
  }
};

class Longboard
{
  public:
  void rollen() {
    std::println("Ich rolle!!");
  }
};

class Fortbewegung
{
  public:
  static void sich_fortbewegen(Flugzeug &f){
    f.fliegen();
  }

  static void sich_fortbewegen(Auto &a){
    a.fahren();
  }

  static void sich_fortbewegen(Longboard &l){
    l.rollen();
  }
};

int main()
{

  Flugzeug f;
  Auto a;
  Longboard l;

  Fortbewegung::sich_fortbewegen(f);    Ich fliege!
  Fortbewegung::sich_fortbewegen(a);    Ich fahre!
  Fortbewegung::sich_fortbewegen(l);    Ich rolle!!

  return 0;
}
```

Listing 1: Fortbewegung. **Hinweis:** Bei der Methode *static void sich_fortbewegen(...)* handelt es sich um eine statische Methode. Statische Methoden verändern keine Daten einer Klasseninstanz und können somit aufgerufen werden, ohne dass eine Instanz (Objekt) der Klasse existiert. Ein Aufruf erfolgt mit Hilfe des Scope-Operators :: und vorangestelltem Klassennamen.