LANDESBERUFSSCHULE 4 SALZBURG

Informatik

Polymorphismus

LBS 4

Dieses Skript dient als zusätzliche Lernunterlage für Informatik

Inhalt

Polymorphismus

Polymorphie bedeutet „Vielgestaltigkeit“. Der Begriff Polymorphie ist eng mit der Vererbung verbunden. Das Konzept bezeichnet die Tatsache, dass eine Methode in verschiedenen abgeleiteten Klassen einer Hierarchie (Vererbung) unterschiedlich implementiert sein kann. Damit werden verschiedene Objekte gleichbehandelt, haben aber andere Funktionen.

Methoden die in mehreren Klassen benötigt werden, werden in der Basisklasse deklariert. In den Abgeleiteten Klassen werden diese neu Deklariert und Definiert.

Polymorphismus beschreibt die dynamische Bindung einer Methode. Dabei wird zur Laufzeit entschieden, welche Methode verwendet wird.

Frühe Bindung – späte Bindung

Bei der „frühen Bindung“ weiß der Compiler zur Kompilierzeit, welche Methode ausgeführt oder verwendet werden muss.

Bei der „späten Bindung“ weiß der Compiler nicht welche Methode(n) ausgeführt werden sollen. Es wird zur Laufzeit entschieden, welcher Code ausgeführt wird.

Schlüsselwort: virtual

Werden Methoden in der Basislasse mit „virtual“ deklariert, dann ist die Methode automatisch virtual in der Subklasse. Das Schlüsselwort virtual stellt sicher, dass beim Aufruf auch wirklich die überschriebenen Methoden aufgerufen werden und nicht die der Basisklasse. Wenn man erzwingen will, dass in jeder Subklasse die Definition der virtuellen Methode erfolgen muss, dann verwendet man „rein virtuelle Methoden“. Rein virtuelle Methoden werden in C++ auf „0“ gesetzt.

Beispiel: C++

```
class Auto{
public:
virtual void marke() = 0;
virtual ~Auto();
};
```

Durch das „0“ setzen der Methode kann die Basisklasse nicht mehr initialisiert werden. Man spricht dann von „abstrakten Klassen“.

Methoden in den Subklassen müssen die gleiche Signatur, wie in der Basisklasse besitzen.

Jede Klasse mit einer virtuellen Methode muss auch einen virtuellen Destruktor besitzen. Allgemein soll der Destruktor immer **virtua**l gesetzt sein.

Abstrakte Klassen:

Abstrakte Klassen – Klassen die nicht instanziiert werden können – machen nur Sinn, wenn man eine Vorlage – ähnlich einem Interface – bereitstellen will, aber das Verhalten in jeder Subklasse anders implementiert wird.

In der abstrakten Klasse können auch Methoden implementiert werden, welche in der Vererbungshierarchie verwendet wird.

Wichtig:

Abstrakte Klassen besitzen mindestens eine „rein virtuelle Methode“
„rein virtuelle Methoden“ müssen in der Subklasse definiert werden

Basisklassenzeiger:

Basisklassenzeiger können mit Objekten der Subklassen instanziiert werden, dagegen dürfen Subklassenzeiger nicht mit Basisklassen instanziiert werden.

Werden Objekte über Basisklassenzeiger verwaltet, dann weist das Schlüsselwort „*virtual“* den Compiler an, nur die Methode des betreffenden Objektes auszuführen. Statisch erzeugte Objekte verfügen nur über die eine Methode. (frühe Bindung)

Verwendung finden Basisklassenzeiger bei dynamischer Reservierung von Speicher. So kann zur Laufzeit entschieden werden, welches Objekt erzeugt werden soll.

Beispiel:

```
class Auto{
public:
virtual void brand()=0;
~Auto();
};
//Definition Auto
void Auto::brand(){};
Auto::~Auto(){};
```

Die abgeleitete Klasse:

```
class BMW : public Auto{
public:
void brand(); //Überladen der Methode
~BMW();
};
//Definition BMW
void BMW::brand(){
cout<< "BMW" << endl;
throw 20;
}
BMW::~BMW(){};
```

```
class VW : public Auto{
public:
void brand();//überladen der Methode
~VW();
};
//Definition VW
void VW::brand(){
cout<< VW << endl;
}
VW::~VW(){};
```

Beispiel C++:

```
#include "Auto.h"
#include "BMW.h"
#include "VW.h"
int main()
{
   Auto *car[2]; //Basisklassen Array

   car[0] = new BMW;
   car[1] = new VW;
   car[1]->brand();
return 0;
```