Polymorphie

Florian Adamsky, B. Sc. (PhD cand.)

florian.adamsky@iem.thm.de
http://florian.adamsky.it/

Softwareentwicklung im WS 2014/15

TECHNISCHE HOCHSCHULE MITTELHESSEN

Outline

1 Einführung

2 Operator Overloading

3 Method Overloading

4 Virtuelle Methoden

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Inhaltsverzeichnis

1 Einführung

2 Operator Overloading

3 Method Overloading

4 Virtuelle Methoden

TECHNISCHE HOCHSCHULE MITTELHESSEN

Was ist Polymorphie?

- Polymorphie/Polymorphismus (griechisch)

  polys "*viele, mehrere*"
  morphē "*Form, Gestalt*"

- Objekt verhält sich unterschiedlich in unterschiedlichen Kontexten
- drei wesentliche Formen:
  - Operator Overloading
  - Method Overloading
  - Virtuelle Methoden

THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Was ist Polymorphie?

- Polymorphie/Polymorphismus (griechisch)

  polys "*viele, mehrere*"
  morphē "*Form, Gestalt*"

- Objekt verhält sich unterschiedlich in unterschiedlichen Kontexten
- drei wesentliche Formen:
  - Operator Overloading
  - Method Overloading
  - Virtuelle Methoden

  } statisches Binden (Compilezeit) — Operator Overloading, Method Overloading

Was ist Polymorphie?

- Polymorphie/Polymorphismus (griechisch)

  polys "*viele, mehrere*"
  morphē "*Form, Gestalt*"

- Objekt verhält sich unterschiedlich in unterschiedlichen Kontexten
- drei wesentliche Formen:
  - Operator Overloading
  - Method Overloading

  } statisches Binden (Compilezeit)
  - Virtuelle Methoden } dynamisches Binden (Laufzeit)

Inhaltsverzeichnis

1 Einführung

2 Operator Overloading

3 Method Overloading

4 Virtuelle Methoden

TECHNISCHE HOCHSCHULE MITTELHESSEN

Sie kennen Operator Overloading schon!

Example (Integer Addition)

```
int x = 23 + 42;
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Sie kennen Operator Overloading schon!

Example (Integer Addition)

```
int x = 23 + 42;
```

Example (Gleichkommazahl Addition)

```
double y = 23.42 + 2.0;
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Sie kennen Operator Overloading schon!

Example (Integer Addition)

```
int x = 23 + 42;
```

Example (Gleichkommazahl Addition)

```
double y = 23.42 + 2.0;
```

Example (String-Verbindung)

```
string str = "foo" + "bar";
```

Syntactic Sugar

- Operator Overloading bietet keine neue Funktionalität, sondern lediglich *Syntactic Sugar*
- Man könnte auch alles mit Methoden lösen

Example (String Concatenation mit Methoden)

```
string str1 = "foo", str2 = "bar", str3 = "test";
string str4 = str1.join(str2.join(str3));    // schwer zu lesen
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Syntactic Sugar

- Operator Overloading bietet keine neue Funktionalität, sondern lediglich *Syntactic Sugar*
- Man könnte auch alles mit Methoden lösen

Example (String Concatenation mit Methoden)

```
string str1 = "foo", str2 = "bar", str3 = "test";
string str4 = str1.join(str2.join(str3));     // schwer zu lesen
```

Example (String Concatenation mit Operator Overloading)

```
string str1 = "foo", str2 = "bar", str3 = "test";
string str4 = str1 + str2 + str3;             // leichter zu lesen
```


TECHNISCHE HOCHSCHULE MITTELHESSEN

Welche Operatoren?

Example (Operatoren die überladen werden können)

```
new        +   %   ~   >    /=   |=    <<=   >=   --    ()
delete     -   ^   !   +=   %=   <<    ==    &&   ,     []
new[]      *   &   =   -=   ^=   >>    !=    ||   ->*
delete[]   /   |   <   *=   &=   >>=   <=    ++   ->
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Syntax

```
obj1 + obj2
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

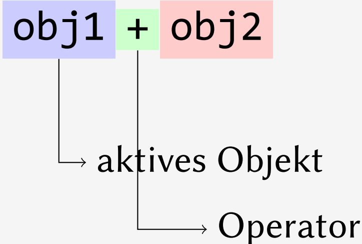
Syntax

`obj1 + obj2`

obj1 → aktives Objekt

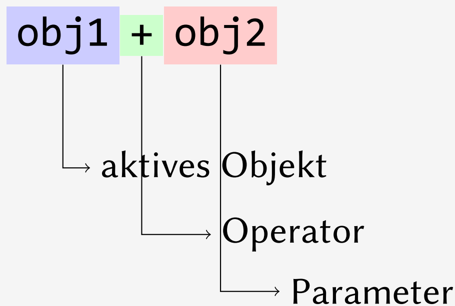
THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Syntax



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Syntax



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Schlüsselwort operator

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+ (const Bruch& foo);

private:
   int zaehler;
   int nenner;
};
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Schlüsselwort operator

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+ (const Bruch& foo);

private:
   int zaehler;
   int nenner;
};
```

Rückgabewert: Neues Bruch-Objekt

Schlüsselwort operator

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+ (const Bruch& foo);

private:
   int zaehler;
   int nenner;
};
```

Operator + soll überladen werden

TECHNISCHE HOCHSCHULE MITTELHESSEN

Schlüsselwort operator

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+ (const Bruch& foo);

private:
   int zaehler;
   int nenner;
};
```

Parameter: am besten als const Referenz

THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Inhaltsverzeichnis

1 Einführung

2 Operator Overloading

3 Method Overloading

4 Virtuelle Methoden

TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
```

Example (Signaturen)

THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
```

Example (Signaturen)

```
add(char, double)
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
int add(int bla);
```

Example (Signaturen)

```
add(char, double)
```

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
int add(int bla);
```

Example (Signaturen)

```
add(char, double)
add(int)
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
int add(int bla);
void add(char buch, int zahl);
```

Example (Signaturen)

```
add(char, double)
add(int)
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Signaturen

- Methoden werden über Signaturen identifiziert
- Signature setzt sich aus Name und Parameter zusammen
- Implementierung unterschiedlicher Methoden mit gleichem Namen

Example (Methoden)

```
void add(char buch, double zahl);
int add(int bla);
void add(char buch, int zahl);
```

Example (Signaturen)

```
add(char, double)
add(int)
add(char, int)
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Method-Overloading Beispiel

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+(const Bruch& summand);
    Bruch operator+(const int& summand);
    Bruch operator+(const double& summand);

private:
   int zaehler;
   int nenner;
};
```

Method-Overloading Beispiel

Example (Bruch-Beispiel)

```
class Bruch {
public:
    Bruch(int z, int n);
    Bruch operator+(const Bruch& summand);
    Bruch operator+(const int& summand);
    Bruch operator+(const double& summand);

private:
   int zaehler;
   int nenner;
};
```

Gleicher Methodennamen, unterschiedlich Methode, unterschiedlicher Parameter

TECHNISCHE HOCHSCHULE MITTELHESSEN

Sich absetzen von den Eltern

- Oberklasse vererbt der Unterklasse alle ihre `public` Methoden
- Unterklasse erbt eine Methode, will aber darin etwas anderes tun, muss man diese *überschreiben* (overloading)

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Method Overloading bei Vererbung

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Method Overloading bei Vererbung

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Überschreibe die Methoden `singen()` von der Elternklasse

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Method Overloading bei Vererbung

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Überschreibe die Methoden `singen()` von der Elternklasse

Example (Main)

```
int main(void) {
    Eltern eltern;
    Kind   kind;

    eltern.singen();
    kind.singen();
}
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Method Overloading bei Vererbung

Example (Klassen)

```
class Eltern {
public:
    void singen() {

    }
};

class K
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
int main(void) {
    Eltern eltern;
    Kind   kind;
```

Ausgabe

```
$ ./programm
Eltern: mimimi
Kind: lalala
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Probleme bei Method Overloading

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

TECHNISCHE HOCHSCHULE MITTELHESSEN

Probleme bei Method Overloading

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
void FamilienSingen(?????) {
    fam->singen();
}

int main(void) {
    Eltern eltern;
    Kind   kind;

    FamilienSingen(&eltern);
    FamilienSingen(&kind);
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Probleme bei Method Overloading

Example (Klassen)

```
class Eltern {
public:
   void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
void FamilienSingen(Eltern* fam) {
    fam->singen();
}

int main(void) {
    Eltern eltern;
    Kind   kind;

    FamilienSingen(&eltern);
    FamilienSingen(&kind);
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Probleme bei Method Overloading

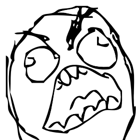
Example (Klassen)

```
class Eltern {
public:
    void
    
    }
};

class K
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
void FamilienSingen(Eltern* fam) {
    fam->singen();

    FamilienSingen(&Kind);
}
```

Ausgabe




```
$ ./programm
Eltern: mimimi
Eltern: mimimi
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Inhaltsverzeichnis

1 Einführung

2 Operator Overloading

3 Method Overloading

4 Virtuelle Methoden

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Lösung des Problems: virtuelle Methoden

Example (Klassen)

```
class Eltern {
public:
   virtual void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Lösung des Problems: virtuelle Methoden

Example (Klassen)

```
class Eltern {
public:
   virtual void singen() {
       cout << "Eltern: mimimi";
   }
};

class Kind : public Eltern {
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
void FamilienSingen(Eltern* fam) {
    fam->singen();
}

int main(void) {
    Eltern eltern;
    Kind   kind;

    FamilienSingen(&eltern);
    FamilienSingen(&kind);
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Lösung des Problems: virtuelle Methoden

Example (Klassen)

```
class Eltern {
public:
    virt
    }
};

class K
public:
    void singen() {
        cout << "Kind: lalala";
    }
};
```

Example (Main)

```
void FamilienSingen(Eltern* fam) {
    fam->singen();
    FamilienSingen(&kind);
}
```

Ausgabe

```
$ ./programm
Eltern: mimimi
Kind: lalala
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Alle in einen Sack stecken

Example (Beispiel)

```
int main(void) {
    Eltern sack[2];

    sack[0] = Eltern();
    sack[1] = Kind();

    for (int i = 0; i < 2; i++) {
        sack[0]->singen();
    }
}
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Alle in einen Sack stecken

Example (Beispiel)

```
int main(void) {
    Eltern sack[2];

    sack[0] = Eltern();
    sack[1] = Kind();

    for (int i = 0; i < 2; i++) {
        sack[0]->singen();
    }
}
```

Ausgabe

```
./programm
Eltern: mimimi
Kind: lalala
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Arten von virtuellen Methoden

Definition (Virtuelle Methode)

Die Oberklasse gibt Funktionalität vor; die Unterklasse kann diese jedoch bei Bedarf überschreiben.

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Arten von virtuellen Methoden

Definition (Virtuelle Methode)

Die Oberklasse gibt Funktionalität vor; die Unterklasse kann diese jedoch bei Bedarf überschreiben.

Definition (Rein virtuelle Methode)

Die Oberklasse gibt keine Funktionalität vor. Die Unterklasse muss die Methode überschreiben. Dadurch kann von der Oberklasse kein Objekt mehr erzeugt werden. Man spricht hier auch von einer abstrakten Klasse.

```
virtual void methode(int x) = 0;
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Wie funktioniert Vererbung und Polymorphie?

- Compiler erzeugt für jede Klasse eine Tabelle
- virtual method table (VMT) oder virtual function table (VFT)
- Konzept dass dahinter steckt nennt sich: Dispatch Table

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

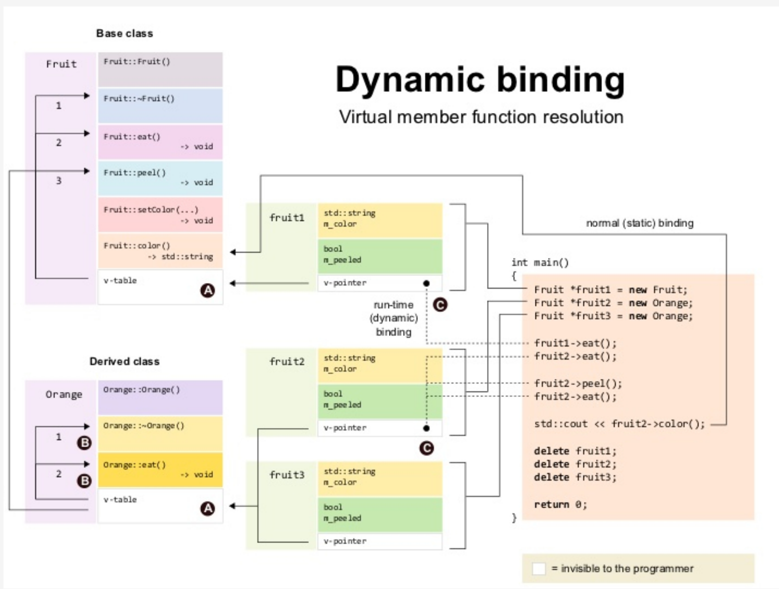


Abbildung: Quelle: Blog: The Syntactic Sugar

THM TECHNISCHE HOCHSCHULE MITTELHESSEN