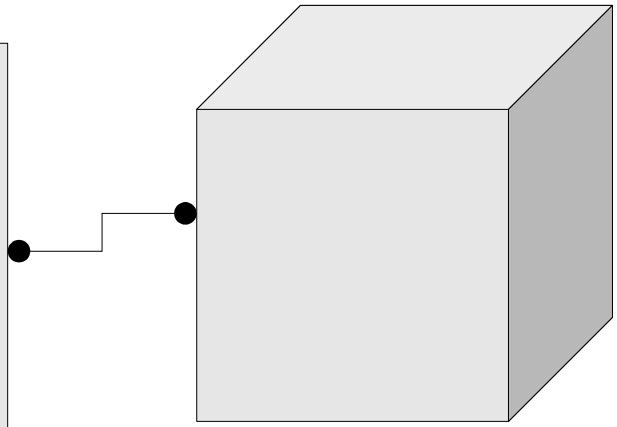
C++ - Objektorientierte Programmierung

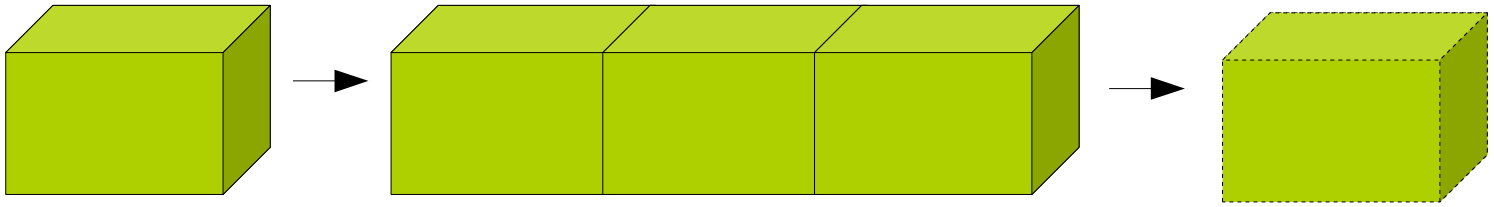
Klassen und Objekte

… hat eine Kantenlänge
… hat eine Füllfarbe

Kantenlänge setzen
Füllfarbe lesen
Volumen berechnen



Klasse „Warteschlange“

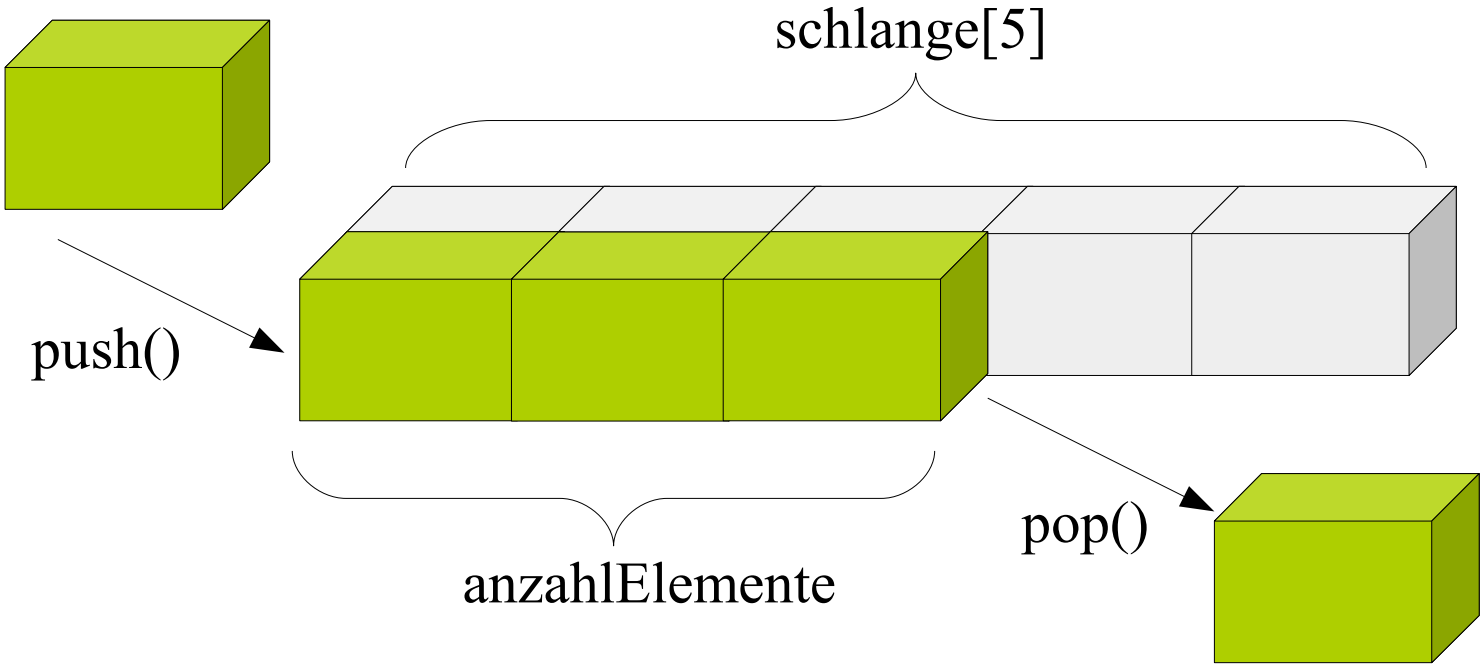


- Attribute: Wie viele Personen stehen in der Schlange (anzahlElemente)? Die Warteschlange (schlange[5]). Die Attribute bilden die Eigenschaften einer Objektgruppe ab. Mit Hilfe der Attribut-Werte wird eine Warteschlange beschrieben.
- Mit Hilfe der Methode pop() wird das Element am Kopf der Warteschlange entfernt. Die Methode push() fügt ein Element am Ende der Warteschlange ein.

Klasse

- Abstraktion von Dingen aus der realen Welt.
- Allgemeine Beschreibung von einem Objekt. Welche Eigenschaften hat ein Objekt? Wie verhält sich das Objekt?
- Bauplan für ein bestimmtes Ding.
- Vorlage für die Erzeugung eines Objektes.

Beispiel „Klasse“



Leibniz Universität Hannover

Aufbau einer Klasse in C++

```
class clsQueue {

public:
    void pop();
    void push(int);
    int peek();
    bool isEmpty();
    bool isFull();
    void writeSchlange();

private:
    int schlange[5];
    int anzahlElemente = 0;

    void setArrayNull();
};
```

```
class clsQueue {

    public:
        Methoden();

    private:
        attribut;
        attribut;
        Methoden();

}
```

Beispiele/cppOOP_002_...

Deklaration einer Klasse

- Welche Attribute werden für die Abbildung des Objektes benötigt?
- Wie sieht die Schnittstelle aus? Welche Methoden werden benötigt, um die gewünschten Änderungen an den Attributen durchzuführen? Welche Informationen über das Objekt müssen nach außen gegeben werden?
- Wie werden die Objekte erzeugt und zerstört?

… in C++

```
class clsQueue
{

};
```

- Beginn mit dem Schlüsselwort class.
- Dem Schlüsselwort folgt der Name der Klasse. Die Bezeichnung ist eindeutig.
- Der Rumpf der Klasse wird durch die geschweiften Klammern begrenzt.
- Jede Deklaration einer Klasse endet mit einem Semikolon.

… in einer Header-Datei (...h)

```
class clsQueue {

    public:
        clsQueue();
       ~clsQueue
        Methoden-Prototyp();
        Methoden-Prototyp();

    private:
        attribut;
        attribut;
        Methoden_Prototyp();

}
```

- Deklaration der Klasse.
- Attribute werden deklariert
- In der Klasse werden die Prototypen der Methoden beschrieben.
- Prototypen für Konstruktoren zum Erzeugen und Destruktoren zum Zerstören von Objekten werden geschrieben.

Implementierungsdatei (...cpp)

- Wie werden die Methoden, Konstruktoren und Destruktoren implementiert?
- Welche Funktionalitäten beschreiben die Methoden.
- Initialisierung von statischen Variablen.
- Mit Hilfe der Anweisung `#include` wird die zu implementierende Klasse eingebunden wird.

… in NetBeans

- Rechter Mausklick auf den Projektnamen im Projekt-Explorer. *New* – *C++-Class*.
- Oder: *File* – *New File*. Auswahl der Kategorie *C++* und den Dateityp *C++-Class*.
- In beiden Fällen wird automatisch für die Klasse eine Header-Datei mit der Endung „.h“ und eine Codedatei mit der Endung „.cpp“ angelegt.

Hinweise

- Der Name der Header-Datei und der Implementierungsdatei sollten sich nur in der Datei-Endung unterscheiden.
- Klassen werden in der cpp-Datei `main` als Datentypen genutzt. Sie sollten dort aber nicht deklariert werden.
- Klassen können nicht in Funktionen oder Anweisungsblöcken deklariert werden.
- Klassen können nur als Attribute in anderen Klassen genutzt werden.

Bezeichner

- Namen für Klassen, Funktionen / Methoden, Konstanten, Variablen / Attribute etc.
- Die Namen sind in ihrem Gültigkeitsbereich eindeutig.
- Schlüsselwörter der Programmiersprache sind als benutzerdefinierte Namen nicht erlaubt.
- Unterscheidung zwischen Groß- und Kleinschreibung. „clsQueue“ und „ClsQueue“ sind Namen für zwei verschiedene Klassen.

Erlaubte Zeichen

- Buchstaben A...Z und a...z.
- Zahlen 0...9.
- Der Unterstrich.

Konventionen

- Benutzerdefinierte Namen beginnen mit einem Buchstaben.
- Namen, die mit einem Unterstrich beginnen, werden für Bezeichnungen aus dem Standard von C++, den zugelassenen Bibliotheken und Makronamen genutzt.
- Leerzeichen werden durch Unterstriche ersetzt. Zum Beispiel celsius_To_Fahrenheit.
- Namen, die nur aus Großbuchstaben und Unterstrichen bestehen, sind Makros oder Konstanten vorbehalten.

Konventionen für Klassennamen

- Klassennamen sollten mit einem Großbuchstaben beginnen.
- Häufig haben Klassennamen das Präfix „C“, um sie von anderen Elementen, wie zum Beispiel „Strukturen“, in C++ zu unterscheiden.

Weitere Hinweise

- Dinge aus der realen Welt und die beschreibende Klasse in C++ sollten den gleichen Namen nutzen.
- Jedes Teilwort eines Klassennamen beginnt mit mit einem Großbuchstaben.
- Es wird ein Sprachraum genutzt.

… in einer Projektbeschreibung

- Klassen entsprechen Substantiven in einer Projektbeschreibung.
- Redundante oder für die Aufgabenstellung unnötige Klassen werden nicht implementiert.

Beispiele

- Das *Konto* wird für den *Inhaber* xyz von der *Bank* a eröffnet.
- Das *Auto* HI hat den *Besitzer* xyz.
- Mitarbeiter *Müller* arbeitet an dem *Projekt* „Ausschreibung“.
- Die *Bestellung* wird über einen *Laserdrucker* xyz ausgedruckt.
- *Außendienstmitarbeiter* Meier betreut *Kunden* im *Bezirk* Unterweser.

Attribute (Member, Instanzvariablen)

- Beschreibung eines Gegenstandes, einer Person, etc.
- Allgemeingültige Beschreibung für einen bestimmten Objekttyp.
- Jedes Objekt einer Klasse hat die gleichen Attribute.
- Jedes Objekt einer Klasse unterscheidet sich aber in mindestens einem Attribut-Wert von allen anderen Objekten.
- Für jedes Objekt werden die Attribute im Speicher abgelegt.

… in C++

```
class clsQueue {

public:

private:
    int schlange[5];
    int anzahlElemente = 0;

};
```

Deklaration der Attribute

int	anzahlElemente	;
Datentyp	name	;

- Der Datentyp wird in Abhängigkeit des zu speichernden Wertes festgelegt.
- Der Name des Attributs ist frei wählbar.
- Attribute werden am Anfang oder Ende der Klasse deklariert.
- Jede Deklaration eines Attributs endet mit dem Semikolon.

… in der Projektbeschreibung

- Attribute entsprechen meist der Beschreibungen eines Objekts.
- Attribute sind Substantive, die einen Rückbezug auf ein anderes Substantiv besitzen.

Beispiele

- Das Auto ist *rot (Farbe)* und hat einen *KM-Stand* von 130.000.
- Die Flächen des Würfels sind *rot* gestrichen, die Kanten sind *schwarz*. Jede Kante hat eine Länge von *5 cm*.
- Die *Menge* eines Artikels im Lager wird erfasst.
- Das *Ausleihdatum* des Buches wurde überschritten.

Speicherung von boolschen Werte

bool	lichtAn	;

- Wahrheitswerte.
- Ist der Ausdruck wahr?
- true. Wahr. Strom / Licht ist eingeschaltet. Nicht null.
- false. Falsch. Strom / Licht ist ausgeschaltet. Null.

Leibniz Universität Hannover

Speicherung von Ganzzahlen

short	ganzzahl	;
int	ganzzahl	;
long	ganzzahl	;
long long	ganzzahl	;

Speicherung von Gleitkommazahlen

float	wert	;
double	wert	;
long double	wert	;

Speicherung von ASCII-Zeichen

char	zeichen	;

zeichen	=	'A'	;	Character
zeichen	=	65	;	Dezimal
zeichen	=	'\u0041'	;	Unicode

Methoden

- Wie verhält sich das Objekt?
- Lesen und modifizieren von Attribut-Werten.
- Zugriff von außen auf die Attribute.
- Methoden werden für alle Objekte einer Klasse gemeinsam im Speicher abgelegt.

… in der Projektbeschreibung

- Verben.
- Beschreibung einer Zustandsänderung / Änderung.
- Lesen von Werten.

… in der Projektbeschreibung

- Der Würfel wird um 90° Grad *gedreht*.
- Der Inhaber *hebt* x Euro vom Girokonto *ab*.
- Die Straßenlaterne wird ab einer Helligkeit von x Prozent *abgeschaltet*.
- Der Barcode der Ware … wird *eingelesen*.
- Die Wegstrecke zwischen dem Punkt x und y wird *gemessen* und *ausgegeben*.

… in einer Klasse definieren (inline)

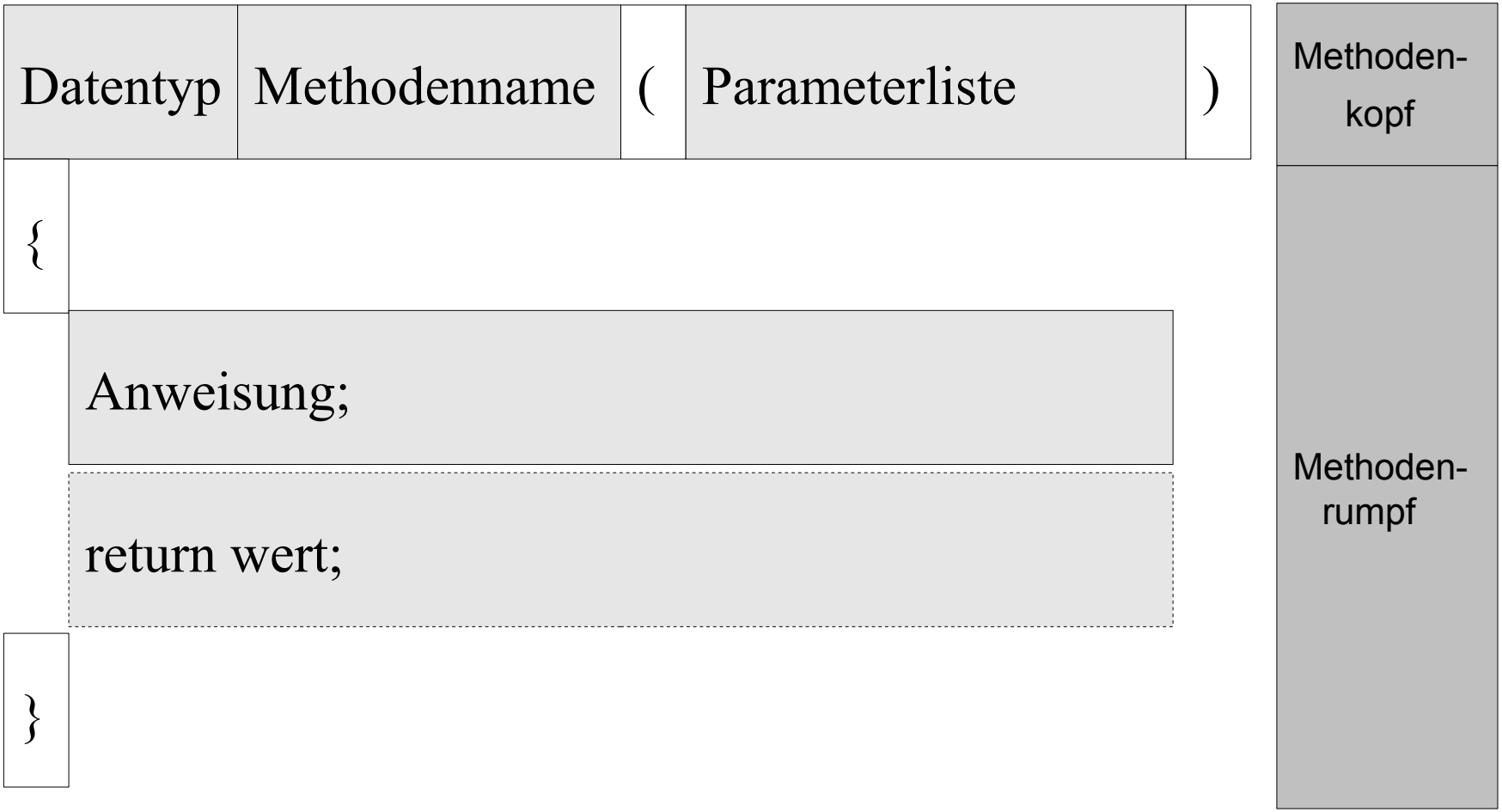
```
class clsQueue {

public:

    bool isFull(){
            int maxElemente;

            maxElemente = sizeof(schlange) / sizeof(int);
            return (anzahlElemente == maxElemente);
    }

private:
};
```

Methodenkopf und – rumpf

bool isFull()	Methodenkopf
{ int maxElemente; maxElemente = sizeof(schlange) / sizeof(int); return (anzahlElemente == maxElemente); }	Methodenrumpf

Leibniz Universität Hannover

Aufbau einer Methode



Hinweise

- Die Methode muss innerhalb der Klasse deklariert werden.
- Methoden, die einen sehr großen Rumpf haben, sollten immer außerhalb der Klasse definiert werden.
- Für einen lesenden Zugriff auf die Attribute oder bei einen sehr kleinen Rumpf kann eine Methode auch in einer Klasse definiert werden. Methoden, die in einer Klasse deklariert und definiert werden, verhalten sich wie inline-Funktionen.

Methodenkopf

Datentyp	methodename	(	Parameterliste	)

- Die Methode ist vom Datentyp .....
- Die Methode wird mit Hilfe des Namens … aufgerufen. Der Name ist eindeutig.
- In den runden Klammern folgt dem Methodennamen eine Parameterliste. Die Parameterliste kann leer sein.

Methodenrumpf

- Beginn und Ende mit den geschweiften Klammern.
- Kapselung von Anweisungen.
- In einem Methodenrumpf kann eine andere Methode aufgerufen, aber nicht definiert werden.

Deklaration als Prototypen in einer Header-Datei

Beispiele/cppOOP_002_...

```
class clsQueue {

public:
    void pop();
    void push(int);
    int peek();
    bool isEmpty();
    bool isFull();
    void writeSchlange();

private:

    void setArrayNull();
};
```

Prototypen von Methoden

- Der Methodenkopf wird als Prototyp für eine Methode in einer Klasse genutzt.
- Die Deklaration und Definition von Methoden werden getrennt.
- Definition einer Schnittstelle nach außen.
- Irgendwo in diesem Programm existiert eine Methode ...

Prototypen

datentyp	methodenName	(			)	;
void	methodenName	(			)	;

datentyp	methodenName	(	typ	,	typ	)	;
void	methodenName	(	typ	,	typ	)	;

Hinweise

- Prototypen von Methoden werden innerhalb einer Klasse geschrieben. Häufig wird die Klasse in einer Header-Datei deklariert.
- Prototypen enden mit einem Semikolon.

Prototyp

Datentyp	methodename	(	Parameterliste	)

- Was für ein Typ von Wert wird an den Aufrufer zurückgegeben?
- In welcher Klasse ist die Methode deklariert?
- Wie heißt die Methode?
- Welche Startwerte müssen der Methode übergeben werden?

Konventionen für Methoden-Bezeichner

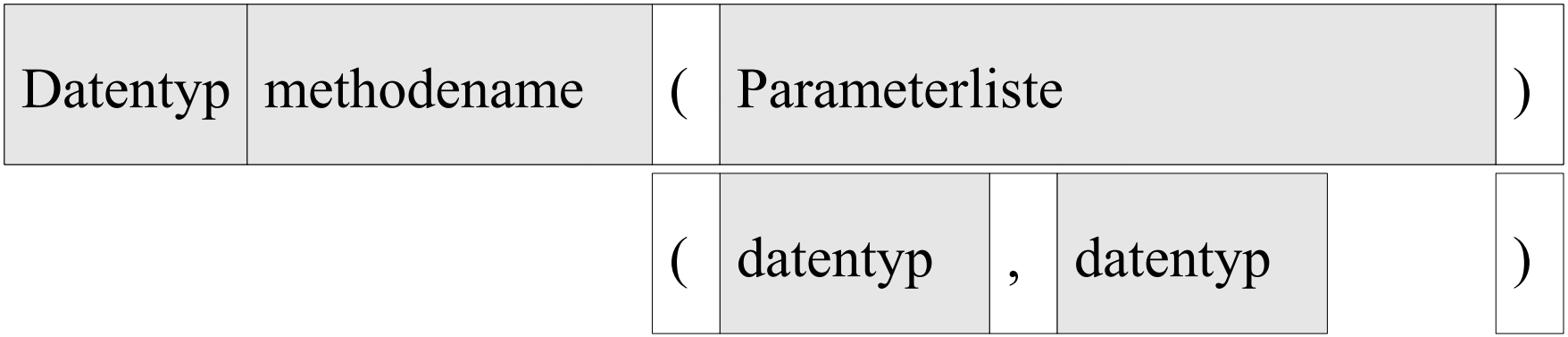
- Methoden, die mit „is“ beginnen, liefern einen boolschen Wert zurück. Zum Beispiel „is_Equal“ überprüft, ob die zwei übergebenen Parameter gleich sind.
- Methoden, die mit „set“ beginnen, setzen eine Variable.
- Methoden, die mit „get“ beginnen, liefern einen beliebigen Wert zurück.

Parameterliste im Methodenkopf

Datentyp	methodename	(	Parameterliste	)

- Beginn und Ende mit Hilfe der runden Klammern.
- Eine Parameterliste kann leer sein.

Parameter in der Parameterliste



- Die Parameter werden durch Kommata in der Parameterliste getrennt.
- Eine Parameterliste kann beliebig viele Parameter haben.
- Für jeden Parameter wird der Datentyp, aber nicht der Name des Parameters angegeben.

Methoden, die keinen Wert zurückliefern

void	methodenName	(	parameterliste	)	;

- Methoden vom Typ void.
- Diese Methoden liefern keinen Wert an den Aufrufer zurück.

Nutzung

- Modifizierung von Attribut-Werten ohne den Aufrufer, über den neuen Wert oder den Status zu informieren.
- Attribut-Werte werden am Bildschirm angezeigt, in eine Datei geschrieben etc.

Methoden, die einen Wert zurückliefern

datentyp	methodenName	(	parameterliste	)	;

- Methoden vom Datentyp …
- Methoden können von jeden beliebigen Typ sein.
- Mit Hilfe der Anweisung `return` wird ein Wert an den Aufrufer zurückgegeben.

Nutzung

- Lesender Zugriff auf Attribute.
- Neu-Berechnung von Attribut-Werten mit Hilfe der übergebenen Parameter.
- Übergabe des Status des aktuellen Objektes.

Definition von Methoden in einer cpp-Datei

```
#include "clsQueue.h"

void clsQueue::pop(){
    if(this->isEmpty() == false){
        schlange[anzahlElemente - 1] = 0;
        anzahlElemente--;
    }
}

int clsQueue::peek(){
    return(schlange[anzahlElemente - 1]);
}
```

Beispiele/cppOOP_002_...

Methodenkopf und – rumpf

int clsQueue::peek()	Methodenkopf

{
return(schlange[anzahlElemente - 1]);
}
	Methodenrumpf

Methodenkopf

int	clsQueue	::	peek	(		)
Datentyp	Klasse	::	methodename	(	Parameterliste	)

- Was für ein Typ von Wert wird an den Aufrufer zurückgegeben?
- In welcher Klasse ist die Methode deklariert?
- Wie heißt die Methode?
- Welche Startwerte müssen der Methode übergeben werden?

Parameterliste im Methodenkopf

Datentyp	Klasse	::	methodename	(	Parameterliste	)

- Beginn und Ende mit Hilfe der runden Klammern.
- Eine Parameterliste kann leer sein.

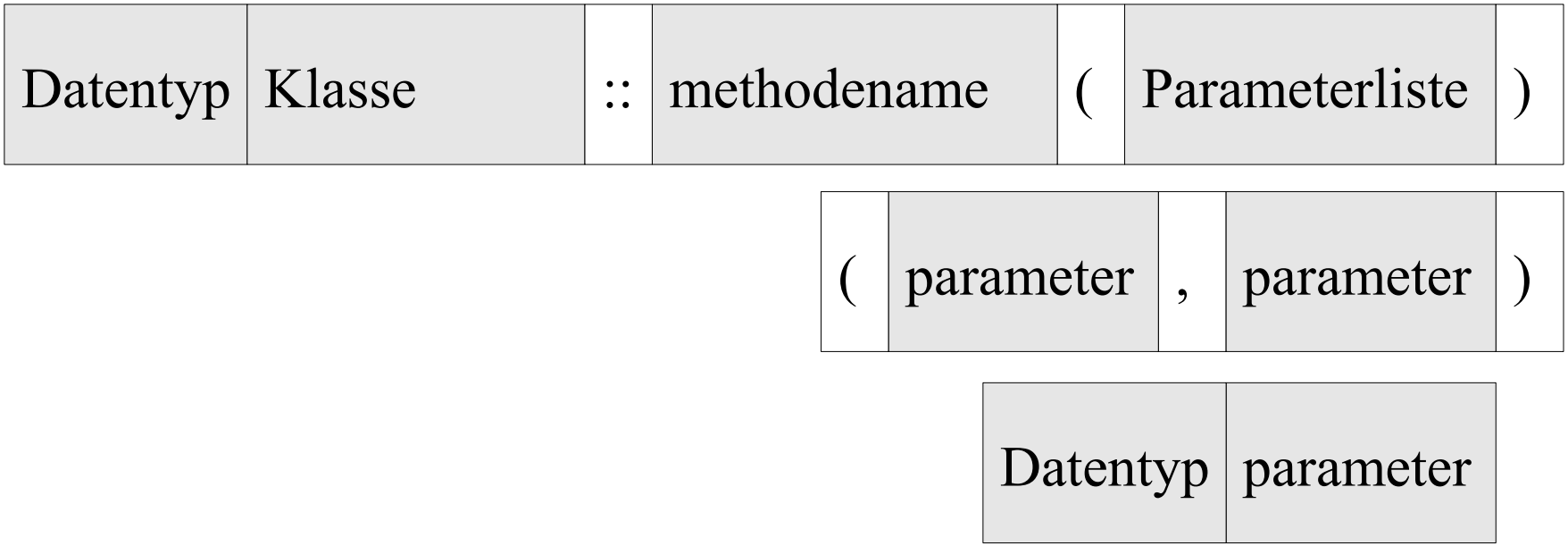
Parameter in der Parameterliste

Datentyp	Klasse	::	methodename	(	Parameterliste	)

(	parameter	,	parameter	)

- Die Parameter werden durch Kommata in der Parameterliste getrennt.
- Eine Parameterliste kann beliebig viele Parameter haben.

Parameter



- Jeder Parameter setzt sich aus einem Datentyp und den Namen des Parameters zusammen.
- Der Name des Parameters ist eindeutig.

In welcher Klasse ist die Methode deklariert?

Datentyp	Klasse	::	methodename	(	Parameterliste	)

- Der Bereichsoperator (Scope-Operator) ordnet einer Methode (rechts) einen Klassennamen (links) zu.
- Der Operator besteht aus zwei direkt aufeinander folgenden Doppelpunkten.
- Der Bereichsoperator identifiziert eindeutig eine Klasse, in der die Methode links vom Operator deklariert ist.
- Klasse::Methode wird als qualifizierter Name bezeichnet.

Methoden, die einen Wert zurückliefern

datentyp	methodenName	(	parameterliste	)	;

- Methoden vom Datentyp …
- Methoden können von jeden beliebigen Typ sein.
- Mit Hilfe der Anweisung `return` wird ein Wert an den Aufrufer zurückgegeben.

Return-Anweisung im Methodenrumpf

```
int clsQueue::peek(){
    return(schlange[anzahlElemente - 1]);
}
```

- Das Schlüsselwort return liefert einen Wert an den Aufrufer zurück.
- Der Wert entspricht dem Datentyp der Methode oder kann in diesen konvertiert werden.
- Der Wert kann mit Hilfe eines Ausdrucks berechnet werden.
- Der Wert kann in Abhängigkeit einer Bedingung zurückgegeben werden.

Zugriffsspezifizierer

Beispiele/cppOOP_002_...

```
class clsQueue {

public:
    void pop();
    void push(int);
    int peek();
    bool isEmpty();
    bool isFull();
    void writeSchlange();

private:
    int schlange[5];
    int anzahlElemente = 0;

    void setArrayNull();
};
```

- Wie kann von außen auf Attribute und Methoden zugegriffen werden?
- Werden Attribute oder Methoden in der Klasse gekapselt?
- Wenn keine Angaben zum Zugriff gemacht wurden, sind Attribute und Methoden privat.

… in C++

```
class clsQueue {

   public:

   private:
};
```

- Lesezeichen für einen bestimmten Abschnitt.
- Definition eines Zugriffsspezifizierer am Anfang einer neuen Zeile in der Form „[zugriffsspezifierer]:“.
- Die Zugriffsspezifizierer werden mit einem Doppelpunkt abgeschlossen.

Privater Zugriff (private:)

- Die Methoden oder die Attribute sind wie in einem Brief verschlossen.
- Ein Zugriff ist nur innerhalb der eigenen Klasse möglich.
- Die Elemente sind in einer Box verschlossen. Sie können nur von dem Eigener der Box genutzt werden.

Nutzung

- Attribute sollten immer privat deklariert werden. Durch den privaten Zugriff werden Attribute in einer Klasse gekapselt.
- Methoden zum internen Gebrauch in der Klasse werden als privat deklariert.

Öffentlicher Zugriff

- Die Elemente können von jeden genutzt werden.
- Ein Zugriff von außen ist möglich.
- Die Box, in der die Elemente aufbewahrt werden, ist für jeden einsehbar.

Nutzung

- Methoden, die Nutzer benötigen, um Attribute zu ändern oder zu lesen, werden als öffentlich gekennzeichnet.
- Methoden zum Erzeugen oder Zerstören eines Objektes müssen öffentlich sein. Objekte werden immer durch einen Anstoß von außen neu angelegt oder zerstört.

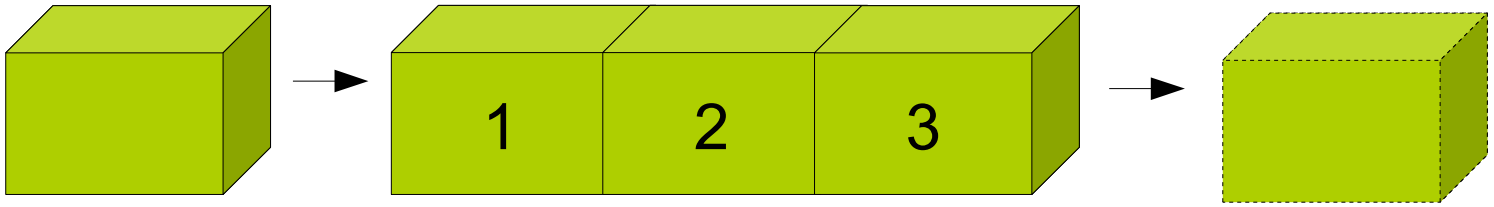
Regeln

- Wenn keine Zugriffsspezifizierer angegeben sind, sind alle Elemente einer Klasse privat. Das heißt, auf keines der Elemente kann von außen zugegriffen werden.
- Zugriffsspezifizierer gelten bis zum nächsten Spezifizierer oder bis zum Ende der Klassendeklaration.
- Ein Spezifizierer kann beliebig oft in einer Klassendeklaration verwendet werden.

Objekte

- Ein Ding (Exemplar, Instanz) aus der realen Welt.
- Beschreibung von Elementen einer Gruppe.
- Alle Elemente einer Kategorie von Dingen haben die gleichen Eigenschaften, aber in unterschiedlichen Ausprägungen. Sie nutzen die gleichen Methoden zum Ändern ihrer Ausprägungen.

Objekt „Warteschlange“



- Eine Warteschlange hat als Attribut ein Array mit x Elementen. Die Anzahl der belegten Elemente in dem Array beschreibt die aktuelle Warteschlange.
- Mit Hilfe der Methode pop() kann die Anzahl der Elemente in der Schlange verringert werden. Ein Element wird am Kopf entfernt. Mit Hilfe der Methode push() wird die Anzahl der Elemente erhöht. Ein Element wird am Ende der Schlange hinzugefügt.

Objektvariable

```
class clsQueue schlangeStandard;
```

- Variable vom Typ „Klasse“.
- Die Klasse muss durch die Anweisung #include eingebunden werden.
- In der Variablen wird als Wert die Speicheradresse für ein Objekt abgelegt.

Deklaration einer Objektvariablen

class	clsQueue	schlangeStandard	;
class	klassenname	variablenname	;

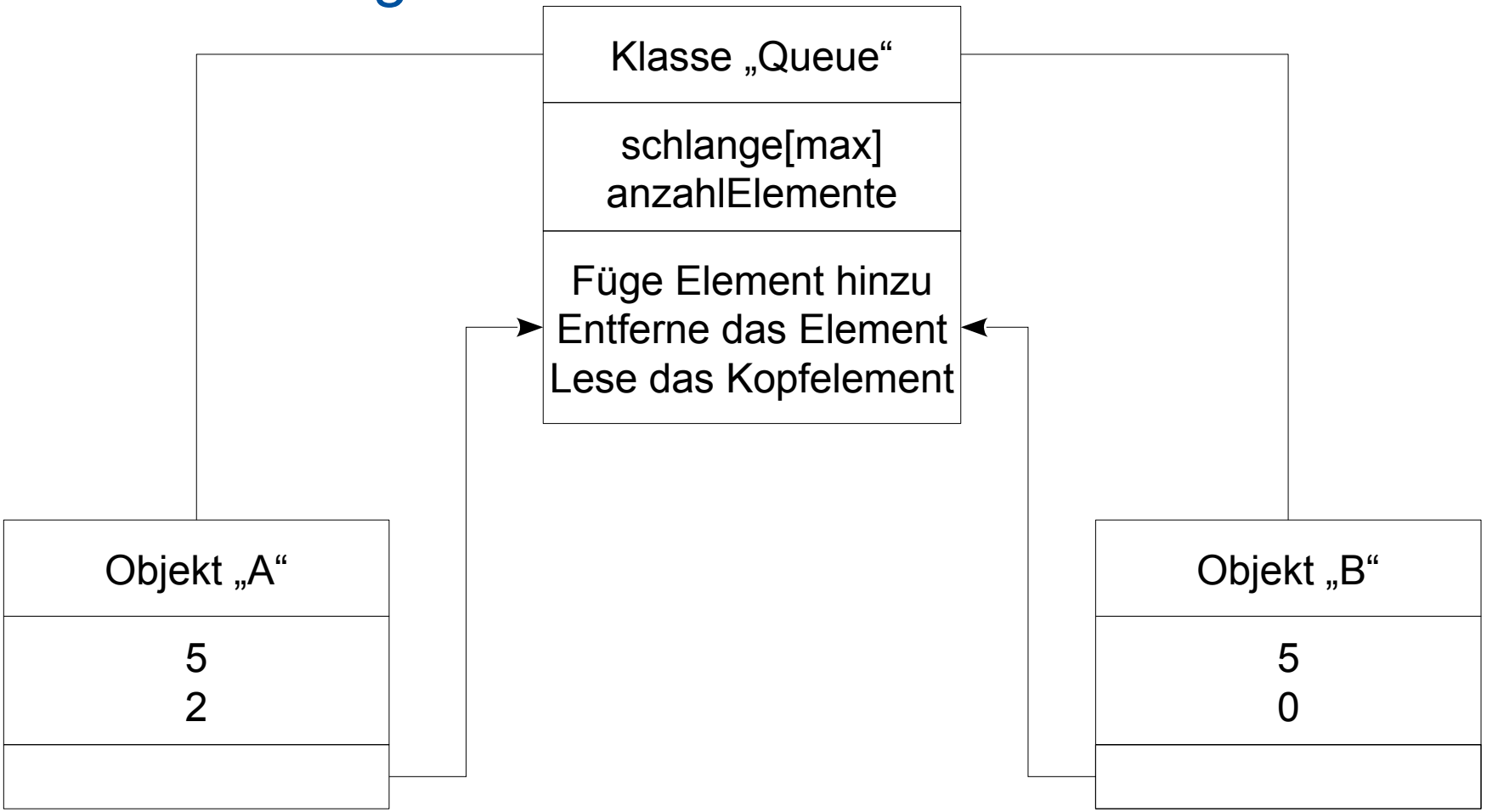
- Als Datentyp wird ein Klassennamen ausgewählt.
- Der Name der Variablen ist frei wählbar.
- Die Deklarationsanweisung kann überall im Anweisungsblock erfolgen. Variablen sollten aber am Anfang eines Anweisungsblockes oder kurz vor deren ersten Nutzung deklariert werden.

Hinweise

- Die genutzten Elemente müssen mit Hilfe einer using-Anweisung für das Modul oder den Anweisungsblock sichtbar gemacht werden.
- Das Schlüsselwort `class` wird bei eindeutigen Klassennamen nicht benötigt, sollte aber immer bei benutzerdefinierten Klassen genutzt werden.

Leibniz Universität Hannover

Instanziierung



Lebenszyklus eines Objekts

… erzeugen:
Aufruf eines Konstruktors

```
class clsQueue schlangeStandard;
```

… arbeiten:
Methoden aufrufen

```
schlangeStandard.push(1);
schlangeStandard.pop();
```

… zerstören:
Aufruf eines Destruktors

Aufruf von Methoden

push	(	1	)	;
methodenname	(	parameterliste	)	;

- Jede Methode wird mit Hilfe ihres Namens aufgerufen.
- Der Name ist ein eindeutiger Platzhalter für die Methode einer Klasse.
- Dem Namen folgt die Parameterliste. Die Parameterliste beim Aufruf entspricht der Liste im Methodenkopf.

Aufruf über eine Objektvariable

schlange	.	push	(	1	)	;
objektvariable	.	methodenname	(	parameterliste	)	;

- Methoden werden immer über eine Objektvariable aufgerufen.
- Die Objektvariable ist vom Typ „Klasse“. Methoden in dieser Klasse können über die Objektvariable aufgerufen werden.
- Falls die aufgerufene Methode nicht in der Klasse deklariert ist, wird der Fehler „... has no member named ...“ vom Compiler geliefert.

Nutzung des Punktoperators

schlange	.	push	(	1	)	;
objektvariable	.	methodenname	(	parameterliste	)	;

- Links vom Punktoperator steht der Name einer Objektvariablen. Die Variable beschreibt eine Instanz einer Klasse
- Rechts vom Punktoperator steht ein Methodenname. Die Methode ist in einer Klasse deklariert.
- Die Instanz ist von dem Typ „Klasse“, in dem die Methode deklariert ist.

Rückgabewerte einer Methode

```
bool istLeer;
istLeer = schlangeStandard.isEmpty();
```

- Eine Methode kann einen Rückgabewert haben, muss aber nicht.
- Der Rückgabewert kann in einer Variablen gespeichert werden.
- Die Variable und der Datentyp der Methode sollte gleich sein.