Hochschule Esslingen
University of Applied Sciences

Objektorientierte Programmierung mit C++:

Wiederholung

Profs. M. Dausmann, D. Schoop, A. Rößler

Fakultät Informationstechnik, Hochschule Esslingen

Frage 1

Was ist ein Konstruktor?

Konstruktoren sind Methoden ohne Rückgabetyp. Der Methodenname eines Konstruktors ist identisch mit dem Namen der Klasse.

Konstruktoren können nicht explizit aufgerufen werden, sondern der Kompiler benutzt die Konstruktoren, wenn Objekte zu 'konstruieren' sind (Definition, new, ...).

Frage 2

Welche Konstruktoren gibt es?

- Standardkonstruktor
- Kopierkonstruktor
- Konvertierungskonstruktor
- Parametrisierter Konstruktor

Frage 3

Was ist ein Standardkonstruktor?

Ist kein Konstruktor definiert, ruft der Kompiler den Standatfkonstrukor (hat keine Parameter) auf

```
Klasse(){}
```

Der vom Kompiler zur Verfügung gestellte Standardkonstruktor initialisiert die Datenelemente mit zufälligen Werten (also quasi gar nicht).

Frage 4

Warum können Konstruktoren nicht überladen werden?

Falsch,

Konstruktoren können wie Methoden überladen werden.

In diesem Punkt sind sie Methoden gleich

Frage 5

Was ist ein Kopierkonstruktor?

Ein Kopierkonstruktor ist dafür da, eine Klasse mit werten eines anderen Objektes der gleichen Klasse zu initialisieren.

Frage 6

Was ist ein Standard-Kopierkonstruktor?

Ein Standard-Kopierkonstruktor ist für jede Klasse automatisch definiert. Man kann jedoch auch einen Kopierkonstruktor selbst definieren:

```
Bruch(const Bruch & b){
        this->zaehler = b.zaehler;
        this->nenner = b.nenner;
}
```

Frage 7

Was ist der Unterschied zwieschen einer Zuweisung und Initialisierung?

Eine Zuweisung erfolgt immer an ein bereits existierendes Objekt, verändert es und kann an beliebiger Stelle mehrmals auf dieses Objekt angewendet werden.

Eine Initialisierung erfolgt immer mit der Definition, also zu Beginn der Lebenszeit eines Objekts; mit der Initialisierung erhält ein Objekt einen quasi voreingestellten inneren Zustand.

Zuweisung: `bruch1 = bruch2;`

Initialisierung: `Bruch bruch1 = bruch2;`

Frage 8

Wird ein Konstruktor bei Call-by-value aufgerufen?

Ja,

Der Kopierkonstruktor wird von der Klasse aufgerufen, die dem Kopierkonstruktor übergeben wird.

```
int fkt_cbv(Bruch o){
      return o.getZaehler()*o.getNenner();
}
```

Frage 9

Warum muss beim Kopierkonstruktor immer die Referenz des Parameters übergebenen werden?

Oihne Referenz würden rekursive Kopien des Parameters gemacht werden.

```
Bruch(const Bruch & b){
        this->zaehler = b.zaehler;
        this->nenner = b.nenner;
    }
```

Frage 10

Was ist der Unterschied zwischen einer flachen und tiefen Kopie?

- Der Standard-Kopierkonstruktor kopiert nur den/die Pointer, z.B. nur den Anker einer verketteten Liste.
  - Man spricht von einer **flachen Kopie**.
- Will man jedoch die komplette Struktur kopieren, z.B. die ganze verkettete Liste, dann spricht man von einer **tiefen Kopie.**
  - Für eine tiefe Kopie benötigt man einen selbstdefinierten Kopierkonstruktor.

Frage 11

Was ist ein Konvertierkonstruktor?

- Ein Konvertierkonstruktor ist ein Konstruktor mit ei-nem Parameter, der nicht ein Objekt der Klasse ist.
- Er dient dazu ein Objekt einer spezifischen Klasse in ein anderes Objekt einer anderen Klasse zu konvertieren (casten)

```
Bruch b(3);
Bruch b = 3;
Bruch b = (Bruch) 3;
Bruch b = Bruch (3);
```

Frage 12

Was ist ein Destruktor?

- Ein Destruktor ist das Gegenstück zum Konstruktor.
  - Er hat keine Parameter und kann nicht überladen werden.
- Er wird automatisch aufgerufen, wenn der Gültigkeits-bereich eines Objektes verlassen wird, oder delete aufgerufen wird.
  - Name: **`~Bruch();`**

Frage 13

Wann brauche ich einen eignen selbtdefinierten Destruktor?

Mit einem **eigen geschriebenen Destruktor** kann man Aktionen veranlassen, die mit dem zu 'destruierenden' Objekt zusammenhängen (Speicher löschen, Datei freigeben, 'tiefe' Struktur löschen).