C++ Kurs

Teil 5

Stefan Westerfeld <stefan@space.twc.de>

Funktionsprototypen

- Will man eine Funktion verwenden, die man
  - später in derselben Datei implementiert
  - in einer anderen Datei implementiert

  => dann braucht man Prototypen

```
int a (int n);       // Prototyp von a

int main()
{
   return a (13);
}

int a (int n)
{
   // Implementation von a
}
```

C++ Objekte als „Blackbox“ (1)

- Konstruktor legt Objekt an

```
std::string hallo („Hallo“);
std::string leer;  // aber nicht uninitialisiert
```

- Methoden ändern Objekte

```
hallo.append („ Welt“);
```

- Methoden fragen Werte ab

```
printf („hallo='%s'\n“, hallo.c_str());
printf („hallo hat %d Zeichen\n“, hallo.length());
```

- Operatoren können „überladen“ werden

```
hallo += „!!!“;
hallo = „... ändern durch Zuweisung“;
printf („Das 3. Zeichen ist '%c'\n“, hallo[2]);
```

C++ Objekte als „Blackbox“ (2)

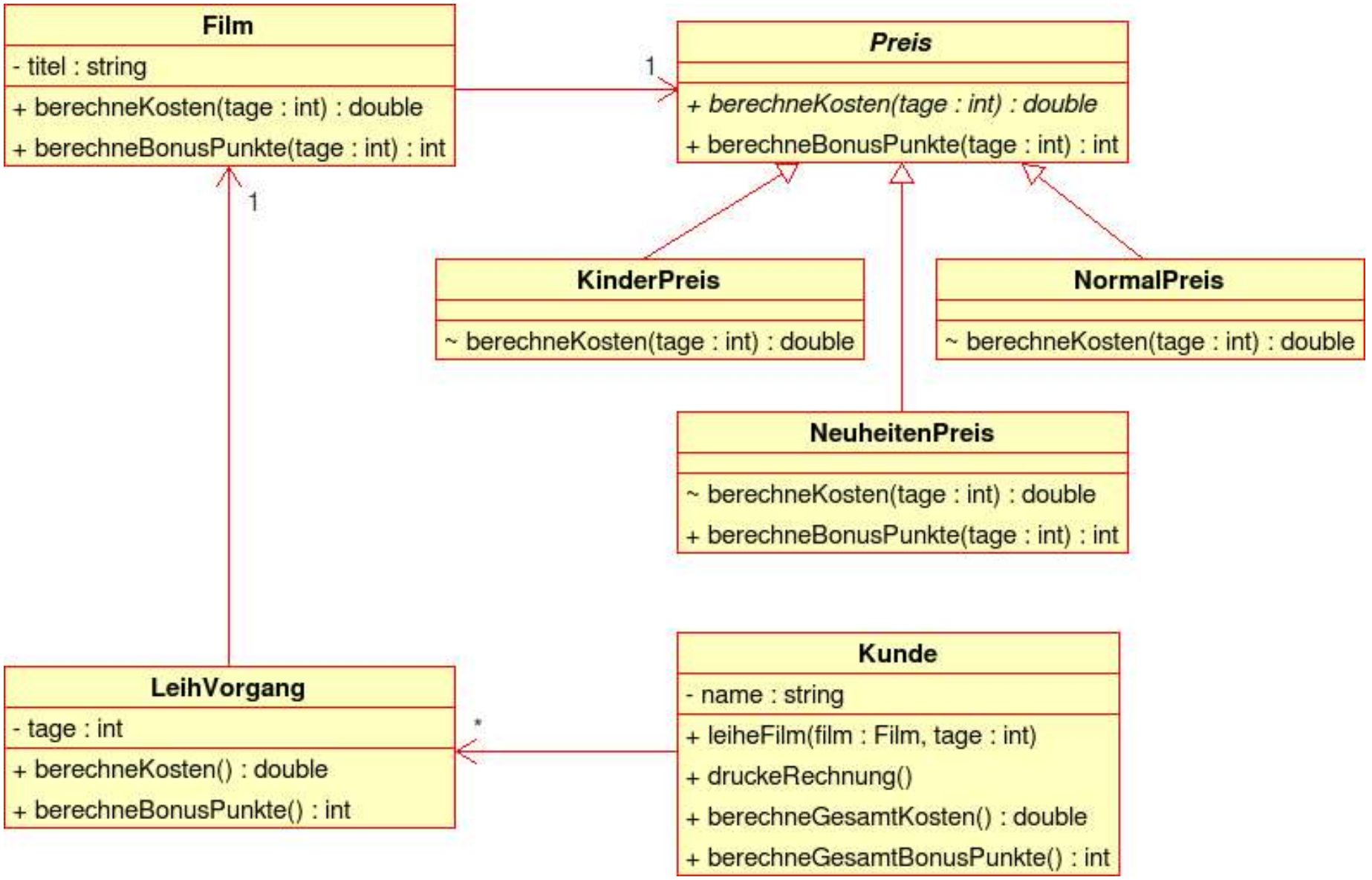
- Destruktor gibt Objekt frei

```
void f()
{
    std::string hallo („Hallo“);  // Konstruktor
     ...
}   // automatischer Aufruf des Destruktors
```

- Meist:

Kein direkter Zugriff auf die Daten des Objekts möglich

Wozu? Ein Beispiel



Aufgabe 1 (Rekursion)

(a) Implementiere die Türme von Hanoi

- http://de.wikipedia.org/wiki/Türme_von_Hanoi

(b) Implementiere zwei Funktionen

```
bool istGerade (unsigned int zahl);
bool istUngerade (unsigned int zahl);
```

die folgende Regeln verwenden:

- eine Zahl ist gerade, wenn ihr Vorgänger ungerade ist
- eine Zahl ist ungerade, wenn ihr Vorgänger gerade ist

finde den dafür geeigneten Rekursionsabschluß

Aufgabe 2 (Approximation)

(a) die Taylorreihe erlaubt Approximationen von Funktionen durch ein Polynom:

$$T_n(x) := f(a) + \frac{f'(a)}{1!}(x-a) + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n,$$

insbesondere gilt:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \quad \text{für alle } x \qquad \cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} \quad \text{für alle } x$$

verwende dieses Wissen, um eine eigene Funktion zur Berechnung von sin(x) zu implementieren, welche im Intervall [0, 2*pi] gut funktioniert

Aufgabe 2 (Approximation)

- messe die Genauigkeit, indem Du Deine Implementation mit sin (x) vergleichst
- finde einen Kompromiss zwischen Genauigkeit und Rechenschritten
- bedenke: es gibt Symmetrien, die man mittels Fallunterscheidung nutzen kann
- http://de.wikipedia.org/wiki/Taylorreihe