PR3 - Fragen zur Vorlesung 10 (Lösungen)

(Stand 2021-12-01 11:08)

Prof. Dr. Holger Peine
Hochschule Hannover
Fakultät IV – Abteilung Informatik
Raum 334, Tel. 0511-9296-1830
Holger.Peine@hs-hannover.de

10.c Objektkomposition

Weitere Erläuterung zur Klicker-Frage
"Wie oft werden `string`-Konstruktoren/-Destruktoren aufgerufen?"

Die Aufrufe von Konstruktoren und Destruktoren der Klasse `string` sind nicht balanciert. Dies führt im vorliegenden Fall zu einem Programmabsturz:

```
class X {
public:
  string* s;
  X() { s = new string(); }
  X(string &ps) : s {&ps} { }
  ~X() { delete s; }
};

void aufgabe() {
  string leer {""};
  X a;
  X b {leer};
}
```

a) Es finden 2 Konstruktoraufrufe und 3 Destruktor-Aufrufe statt.

b)
Die Ursache liegt darin, dass der Destruktor `~X()` annimmt, dass das Attribut `s` immer von der Klasse `X` selbst mit `new` erzeugt worden ist und deshalb auch hier mit `delete` gelöscht werden muss. Der Default-Konstruktor von `X`() erfüllt diese Annahme auch (d.h. er erzeugt tatsächlich `s` mit `new`) – aber der Konstruktor `X(string& ps)` verletzt diese Annahme, denn er lässt `s` auf ein extern erzeugtes `string`-Objekt zeigen. Deshalb ruft das `delete` in `~X()` den `string`-Destruktor auf dem externen Objekt auf, und wenn das Objekt in seinem externen Kontext zerstört wird, wird der `string`-Destruktor ein zweites Mal auf demselben `string`-Objekt aufgerufen, was zu einem Absturz oder schwer zu findenden Laufzeitfehler führt.

Korrigieren lässt sich das Programm, indem wir auch im zweiten Konstruktor eine Kopie des übergebenen Strings mit `new` erzeugen:

```
  X(string& ps) : s {new string(ps)} { }
```

10.d Details zu Klassen und Objekten

Weitere Erläuterung zur Klicker-Frage
"Welche Klasse(n) sind kein korrekter C++-Code?"

```
class A {
  public: A* get() const { return this; }
};

class B {
  public: B* get() { return this; }
};

class C {
  public: const C* get() const {
    return this;
  }
};

class D {
  public: const D* get() {
    return this;
  }
};
```

Musterlösung:
Nur Klasse `A` ist fehlerhaft

Erläuterung:
In der Beobachtermethode `A::get()` ist `this` vom Typ `const A*`. Das passt nicht zum deklarierten Rückgabetyp.
Wir ergänzen zur Verdeutlichung ein Attribut:

```
class A {
public:
  int i;
  A* get() const { return this; };
};
```

Wäre diese Klasse übersetzbar, könnte man ein konstantes Objekt `a` sehr leicht wie folgt versehentlich verändern:

```
const A a;
a.i = 6; // Compilerfehler (gut, der Compiler passt auf ☺)
a.get()->i = 6;  // Würde vom Compiler akzeptiert, wenn A korrekt wäre ☹
```

Die folgende Variante passt zum Beobachterstatus von `get()` und ist korrekt:

```
class B {
public:
  int i;
  const B* get() const { return this; };
};
```

Mit dieser Klasse wird der folgende Code vom Compiler zurück gewiesen. Eine versehentliche Veränderung des konstanten Objekts b ist daher ausgeschlossen:

```
const B b;
b.i = 6; // Compilerfehler (gut, der Compiler passt auf ☺)
b.get()->i = 6;  // Compilerfehler (gut, der Compiler passt auf ☺)
```