UNIVERSITÄT BONN

Algorithmen und Programmierung
Übungsblatt 13
WS 2022/23

Dr. Felix Jonathan Boes
Benedikt Bastin, Ellen Bundschuh, Anna Höpfner, Gina Muuss, Adrian Oeyen, Felix Roth, Thore Wolf

Ausgabe: 16.01.2023
Abgabe: keine Abgabe; Präsentation in der Übung

Dies ist der letzte Übungszettel, der für die Studienleistung (Klausurzulassung) relevant ist. Insbesondere ist dies der letzte Übungszettel, der in der Übung bestanden werden kann.

Aufgabe 1 (Statischer und dynamischer Typ)**.** Betrachten Sie den folgenden Programmcode.

```
1  class X {
2  public:
3    void f()          { std::cout << "X::f" << std::endl; }
4    void g()          { std::cout << "X::g" << std::endl; }
5  };
6
7  class Y : public X {
8  public:
9    void f()          { std::cout << "Y::f" << std::endl; }
10   virtual void g() { std::cout << "Y::g" << std::endl; }
11 };
12
13 class Z : public Y {
14 public:
15   virtual void f() { std::cout << "Z::f" << std::endl; }
16   void g()          { std::cout << "Z::g" << std::endl; }
17 };
18
19 int main() {
20   X x;
21   Y y;
22   Z z;
23
24   X& Xref_auf_y = y;
25   X& Xref_auf_z = z;
26   Y& Yref_auf_z = z;
27
28   Xref_auf_y.f();
29   Xref_auf_z.f();
30   Yref_auf_z.f();
31
32   Xref_auf_y.g();
33   Xref_auf_z.g();
34   Yref_auf_z.g();
35 }
```

Lösen Sie die folgenden Teilaufgaben.

(1) Erstellen Sie das zugehörige UML-Diagram im Sketchingdialekt. Dabei sollen nicht-virtuelle Memberfunktionen durch {non-virtual} gekennzeichnet werden.

(2) Erklären Sie, welche Typen polymorph sind und welche Typen nicht polymorph sind.

(3) Benennen Sie bei jedem Objekt den statischen Typ und den dynamischen Typ. Begründen Sie Ihre Antworten.

(4) Erklären Sie (ohne den Code zu kompilieren und auszuführen), welche Memberfunktion jeweils aufgerufen wird. Geben Sie dabei die aufgerufene Memberfunktion mithilfe eines absoluten Identifiers an.

Aufgabe 2 (Statische und dynamische Konzepte)**.** Der Typ eines Objekts hat sowohl eine statische, als auch eine dynamische Ausprägung. Nennen Sie zwei weitere Konzepte oder Vorgehen, die unterschiedliche Ausprägung zur Compilezeit und Laufzeit haben können.

Aufgabe 3 (Ersetzbarkeit von Parametertypen und Rückgabetypen beim Funktionsüberschreiben)**.** Um Nachzuweisen, dass Sie die Ersetzbarkeit von Parametertypen und Rückgabetypen beim Funktionsüberschreiben erklären können, lösen Sie die folgenden Teilaufgaben.

(1) Wie lautet die Ist-Ein-Faustregel und die beiden Varianten des Liskovschen Substitutionsprinzips?

(2) Erklären Sie was die Begriffe Kovarianz und Kontravarianz (in Kontext dieser Aufgabe) bedeuten.

(3) Beantworten Sie die Frage auf Folie 16 in *Objektorientierte Programmierung mit C++ 06* unter Verwendung des Liskovschen Substitutionsprinzips.