Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

5. August 1997

Aufgaben zur Klausur **Objektorientierte Programmierung** im SS 97 (IA42)

Zeit: 60 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 6 Seiten

---

Aufgabe 1:

Gegeben sei das folgende Programm, das mit virtuellen Funktionen arbeitet:

```
#include <iostream.h>

int linecnt = 0;

class X { public:
  virtual const char * f1 () const { return " X"; }
  virtual const char * f2 () const { return " X"; }
  virtual void g () const {
    cout << ++linecnt << "." << f1() << f2() << endl;
  }
};

class Y : public X { public:
  virtual const char * f1 () const { return " Y"; }
};

class Z : public Y { public:
  virtual const char * f2 () const { return " Z"; }
};

int main() {
  Y y1; y1.g();
  Z z1; z1.g();

  X x1(y1); x1.g();
  Y y2(z1); y2.g();

  X & x3 = z1; x3.g();
  X & x4 = y1; x4.g();

  X * p2 = new Y(); p2->g();
  Z * p3 = new Z(); p3->g();

  X * p4 = p3; p4->g();

  return 0;
}
```

Welche 9 Ausgabezeilen erzeugt dieses Programm

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 2:

Gegeben ist das folgende C++ Programmstück. Dieses soll in eine header Datei *p.h* und eine Implementierungsdatei *p.cc* aufgeteilt werden, so daß es von mehreren anderen Modulen verwendet werden kann. Kennzeichnen Sie durch Ankreuzen des *.h* Feldes für die header Datei oder *.cc* für die Implementierungsdatei, in welche Datei die einzelnen Codestücke übertragen werden müssen.

```
static int counter;                                          // .h  .cc

typedef unsigned long Nat;                                   // .h  .cc

class X;                                                     // .h  .cc

const Nat limit = 8888;                                      // .h  .cc

extern Nat theAnswer;                                        // .h  .cc

#include <iostream.h>                                        // .h  .cc

static int counter = 4711;                                   // .h  .cc

static int foo(const X & x1);                                // .h  .cc

int bar(const X & x1);                                       // .h  .cc

Nat theAnswer = 42;                                          // .h  .cc

class X {                                                    // .h  .cc
private:
  Nat d;
  int check() const;
public:
  X();
  X(const & Nat);
  Nat get() const;
  friend ostream & operator ≫ (ostream &, const X &);
};

Nat X :: get() const {                                       // .h  .cc
  return d + counter;
}

#define lowerbound 1111                                      // .h  .cc

int X :: check() const {                                     // .h  .cc
```

```
  return d ≥ lowerbound;
}

ostream & operator ≫ (ostream & o, const X & x1) {          //  .h  .cc
  return
    o ≪ x1.d ≪ endl;
}

extern Nat numbers[];                                       //  .h  .cc

int bar(const X & x1) {                                     //  .h  .cc
  return
    foo(x1) + 2;
}

static int foo(const X & x1) {                              //  .h  .cc
  return
    x1.get();
}

inline X::X () { d = 0; }                                   //  .h  .cc

struct Field { int data[lowerbound]; };                     //  .h  .cc

extern struct Field aField;                                 //  .h  .cc
```

Aufgabe 3:

In der Objektorientierten Programmierung spielt die Wiederverwendung eine zentrale Rolle. Dabei können schon entwickelte, fertige Klassen von anderen Klassen beerbt oder benutzt werden.

Vererbung

```
class Y : public X { ... };
```

Benutzung

```
class Y {
    X d;
    ...
};
```

Welches sind Vorteile der Vererbung gegenüber der Benutzung der Klasse X in der Klasse Y:

1.	Der Quellcode wird kürzer.	ja ☐	nein ☐	weiß nicht ☐
2.	Der Objektcode wird kürzer.	ja ☐	nein ☐	weiß nicht ☐
3.	Die Laufzeit des Programms verringert sich.	ja ☐	nein ☐	weiß nicht ☐
4.	Die Compilierzeit verringert sich.	ja ☐	nein ☐	weiß nicht ☐
5.	Der Quelltext wird selbstdokumentierend.	ja ☐	nein ☐	weiß nicht ☐
6.	Die Erweiterbarkeit ist verbessert.	ja ☐	nein ☐	weiß nicht ☐
7.	Die Veränderung der Klasse X hat keinen Einfluß auf Programmcode außerhalb von Y.	ja ☐	nein ☐	weiß nicht ☐
8.	Die Namensgebung für Methoden in Y kann besser an die Aufgabenstellung angepaßt werden.	ja ☐	nein ☐	weiß nicht ☐
9.	Die Konstruktoren brauchen nicht neu entwickelt werden.	ja ☐	nein ☐	weiß nicht ☐
10.	Die Speicherverwaltung wird einfacher.	ja ☐	nein ☐	weiß nicht ☐
11.	Die modulare Verständlichkeit nimmt zu.	ja ☐	nein ☐	weiß nicht ☐