Name, Vorname _____ Matrikelnummer _____

Klausur zur Vorlesung

Objektorientierte Anwendungsentwicklung

Krefeld, 23. September 2011

Hinweise:

- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf dieses Deckblatt und auf alle verwendeten Klausurbögen. Diese Aufgabenstellung und alle Klausurbögen müssen am Ende abgegeben werden.
- Alle Antworten sind auf den Aufgabenblättern einzutragen, *nicht* auf dem Klausurbogen. Den Klausurbogen können Sie bei Bedarf als Schmierblatt verwenden.
- Benutzen Sie kein mitgebrachtes Papier. Weiteres Schreibpapier kann bei Bedarf von den Betreuern angefordert werden.
- Die Bearbeitung der Aufgaben darf nur mit dokumentenechten Stiften erfolgen, verwenden Sie *keinen* Bleistift. Verwenden Sie keinen Stift, der in rot schreibt.
- Werden mehrere unterschiedliche Lösungen für eine Aufgabe abgegeben, so wird die Aufgabe nicht gewertet. Antworten, die ein Korrektor nicht lesen kann, werden nicht bewertet.
- Es sind *keine* Hilfsmittel zugelassen.
- Mobiltelefone sind auszuschalten.
- Es ist nur ein Toilettengang erlaubt.

Viel Erfolg!

Bestätigen Sie mit Ihrer nachfolgenden Unterschrift, dass Sie die obigen Hinweise gelesen und verstanden haben, und dass Sie die Klausur selbständig bearbeitet haben.

_____
Unterschrift

Bewertung:

Aufgabe	1	2	3	4	5	6	Summe
Maximalpunkte	19	14	13	26	8	14	94
erreichte Punkte							

_____ (Note) _____ (Datum) _____ (1. Prüfer)

_____ (Datum) _____ (2. Prüfer)

Aufgabe 1: (Quickies) (19 Punkte)

Bitte bearbeiten Sie die folgenden Aufgaben:

(a) Es sind der Präfix-Operator `++` und die binären Operatoren `+=` und `-` als Methoden der Klasse `Test` überladen. Außerdem sind `a` und `b` Objekte der Klasse `Test`. Ersetzen Sie folgende Ausdrücke mit den direkten Aufrufen der Operatorfunktionen:

`a += b;` _____

`b - a;` _____

`++a;` _____

(b) Gegeben sei der folgende Ausschnitt aus einer Klassendefinition. Geben Sie die Liste der Elementinitialisierer (Initialisiererliste) an, die `customer` mit `str` und `id` mit `n` initialisiert.

```
class Assurance{
private:
    string customer;
    long id;
public:
    Assurance(const string &str, long n) : ......................

    ..........................................................
```

(c) Gegeben seien folgende Deklarationen:

```
bool func1(double, double&, double&);
bool func2(double,double*,double*);
double x = 9.7, y = 0.0, z = 0.0;
bool res;
```

Formulieren Sie je eine Anweisung, um die Funktionen `func1()` und `func2()` mit den Variablen `x`, `y` und `z` als Argument aufzurufen. Der Return-Wert soll der Variablen `res` zugewiesen werden.

_____

_____

_____

Aufgabe 1: (Fortsetzung)

(d) Welche der folgenden Klassen sollte zwingend mit einem Destruktor ausgestattet werden. Bitte erklären Sie Ihre Wahl.

```
class A {
private:
    int flaeche;
public:
    A(int x) {
        flaeche = x;
    }
};

class B {
private:
    int *flaeche;
public:
    B(int x) {
        flaeche = new int;
        *flaeche = x;
    }
};
```

(e) Worin besteht der Unterschied zwischen folgenden Listings?

```
class Adresse {
    char name[30];
    char strasse[30];
    long plz;
    char ort[30];
};

struct Adresse {
    char name[30];
    char strasse[30];
    long plz;
    char ort[30];
};
```

Aufgabe 1: (Fortsetzung)

Bitte kreuzen Sie jeweils die richtige Antwort an:

(f) Mit welchem Hilfsmittel lässt sich die Mehrfacheinbindung von Deklarationen in Header-Dateien verhindern?

- ☐ bedingte Kompilierung
- ☐ `extern` Deklaration
- ☐ `static` Deklaration
- ☐ Keine Antwort ist richtig.

(g) Der Begriff virtuelle Methode gehört am ehesten zu welchem der folgenden Aspekte?

- ☐ Kapselung
- ☐ Polymorphie
- ☐ Überladen
- ☐ Keine Antwort ist richtig.

(h) Welche Aussage über Templates triff zu?

- ☐ Templates dienen dazu, abstrakte Klassen zu definieren.
- ☐ Templates dienen dazu, auftretende Datentypen mit `T` abkürzen zu können.
- ☐ Templates sind veraltet und werden in zukünftigen C++-Standards nicht mehr unterstützt.
- ☐ Templates dienen dazu, gleichen Code für verschiedene Datentypen zu verwenden.
- ☐ Keine Antwort ist richtig.

(i) Die korrekte Stelle Default-Argumente für Methoden zu definieren ist ...

- ☐ ... nur in der Deklaration (d.h. in der Schnittstelle).
- ☐ ... nur in der Definition (d.h. in der Implementation).
- ☐ ... sowohl in der Deklaration als auch in der Definition.
- ☐ In C++ gibt es keine Default-Argumente für Methoden.

(j) Statische Membervariablen ...

- ☐ ... sind automatisch auch immer `public`.
- ☐ ... können nur durch statische Methoden verändert werden.
- ☐ ... werden von allen Objekten einer Klasse gemeinsam verwendet.
- ☐ Keine Antwort ist richtig.

Aufgabe 1: (Fortsetzung)

Bitte kreuzen Sie jeweils an, ob die Aussage richtig oder falsch ist. Achtung: Bei einer falschen Antwort erfolgt ein Punktabzug!

		richtig	falsch
(k)	Im Gegensatz zu einer Referenz verfügt eine Zeigervariable über eine eigene Adresse im Hauptspeicher.	☐	☐
(l)	Wenn Sie einen unären Operator als Methode einer Klasse überladen, weist die entsprechenden Operatorfunktion nur einen Parameter auf.	☐	☐
(m)	Ein in einer abgeleiteten Klasse redefiniertes Element verdeckt immer ein gleichnamiges Element, das in der Basisklasse definiert ist.	☐	☐
(n)	Der Konstruktor einer abstrakten Klasse muss rein virtuell sein.	☐	☐
(o)	Wenn für eine Klasse mit dynamischen Elementen bereits der Zuweisungsoperator überladen wurde, dann muss kein eigener Kopierkonstruktor mehr definiert werden, um Objekte dieser Klasse per Werteübergabe („Call by value“) an Funktionen übergeben zu können.	☐	☐
(p)	Angenommen zwei verschiedene Zeiger `ptr1` und `ptr2` zeigen auf denselben dynamisch reservierten Speicher. Dann wird mit der Anweisung `delete[] ptr1;` der dynamische Speicher nicht freigegeben, da der Zeiger `ptr2` immer noch dorthin zeigt.	☐	☐

Aufgabe 2: (C++-Programmierung) (14 Punkte)

(a) Welche Ausgabe erzeugt das folgende C++-Programm? (3 Punkte)

```
#include <iostream>
using namespace std;

class C {
    int n;
public:
    void f() {
        n = n * 10;
    }
    int g() {
        return n;
    }
    C(int i) {
        n = i;
    }
};

int main() {
    C a(1);
    C* b = new C(2);

    a.f();
    cout << a.g() << endl;
    a.f();
    cout << a.g() << endl;
    b->f();
    cout << a.g() + b->g();

    return 0;
}
```

Aufgabe 2: (Fortsetzung)

(b) Welche Ausgabe erzeugt das folgende C++-Programm? (6 Punkte)

```
#include <iostream>
using namespace std;

class A {
public:
    A() {
        cout << "A" << endl;
    }
    virtual A* create() {
        cout << "create A\n";
        return new A;
    }
};

class B: public A {
public:
    B() {
        cout << "B" << endl;
    }
    A* create() {
        cout << "create B\n";
        return new B;
    }
};

void f(A x) {
    x.create();
}

int main() {
    A a;
    f(a);
    B b;
    f(b);
    return 0;
}
```

(c) Ändern Sie das obige Programm so ab, dass der Parameter `x` der Funktion `f` ein Referenzparameter wird. Schreiben sie alle dafür nötigen Änderungen rechts neben die entsprechenden Zeilen des Programms! (1 Punkt)

(d) Wie lautet jetzt die Ausgabe des Programms? (4 Punkte)

Aufgabe 3: (C++-Programmierung) (13 Punkte)

(a) Bestimmen und korrigieren Sie die Fehler in der Definition folgender Funktionen:

1. ```
   double func(double){
       return ((2* double));
   }
   ```

2. ```
   double func(double x){
       return ((2* x))
   };
   ```

(b) Bestimmen und korrigieren Sie die Fehler in den folgenden Anweisungen:

1. ```
   float *p = new float = 1.0F;
   ```

2. ```
   int x = 7.0;
   int *px = new int(x);
   delete x;
   ```

(c) Welche Fehler liegen in folgenden Klassendefinitionen vor?

1. ```
   class A {
   private:
       long secretKey;
   public:
       encode( const string&);
       decode( const string&);
   };
   ```

2. ```
   class B {
   private:
       long limit = 1000L;
       double x = 0.0;
   public:
       void save();
   };
   ```

Aufgabe 3: (Fortsetzung)

(d) In dem folgenden Programm sind insgesamt 10 Fehler verborgen. Markieren Sie mindestens 7 der 10 Fehler und erläutern Sie kurz, was falsch ist.

```
#include <iostream>

class A {
public
    A() {}
    A(int val);
    void const printIt();
private:
    int val;
};

A:A(int nval): val(nval) {
    cout >> "OK" << endl;
}


class B extends A {
    B() {}
    B(int val) : A(val) {}
};


int main() {
    B b(5);
    b->printIt;
}
```

Aufgabe 4: (C++-Programmierung) (26 Punkte)

Betrachten Sie eine Klasse `Liste`, die beliebig viele `int`-Werte speichern kann. Die Werte können mittels `insert` eingefügt werden.

Die Klasse `ExtListe` erweitert die Klasse `Liste` um die Methoden `find`, `remove` und `toString`, außerdem wird die Methode `insert` der Basisklasse überschrieben.

(a) Ergänzen Sie im nachfolgenden C++-Code die Lücken sinnvoll zu einer syntaktisch korrekten Datenstruktur. (22 Punkte)

```
class Liste {

.....................................

    int *values, size, pos;

    // verdoppeln des Speicherplatzes
    void resize() {

        int *tmp = ...............................................

        for (int i = 0; i < size; i++)

            ..........................................................

        delete[] values;
        values = tmp;
        size *= 2;
    }

public:
    // Konstruktor
    Liste(int n = 8) {
        pos = 0;
        size = n;
        values = new int[n];
    }

    // Destruktor
    ~Liste() {
        delete[] values;
    }
```

Aufgabe 4: (Fortsetzung)

```
    // Wert val einfügen und ggf. Speicher vergrößern
    void insert(int val) {
        if (size == pos)

            ..........................................................

        values[pos++] = value;
    }
};


..................................... public Liste {

.....................................

    // Liefert Position, an der val im Array values gespeichert ist.
    int find(int val) {

        ..........................................................

        ..........................................................

        ..........................................................

        return -1;
    }

.....................................

    // Wird so überschrieben, dass eine Exception geworfen wird, falls
    // der einzufügende Wert val bereits gespeichert ist.
    void insert(int val) {
        if (find(val) != -1)

            ..........................................................

        ..........................................................
    }
```

Aufgabe 4: (Fortsetzung)

```
// entfernt den Wert val aus der Liste
void remove(int val) {

    int i = ..................................................

    if (i == -1)
        throw "Value Not Found Exception";

    // aufschieben der Werte im Array um eine Position nach links
    for ( ; i < pos - 1; i++)

        ..........................................................

    pos -= 1;
}

// liefert eine Komma-separierte Ausgabe der Liste als string
string toString() {

    ..........................................................

    for (int i = 0; i < pos; i++) {
        if (i > 0)

            ..................................................

        os << values[i];
    }
    return ...................................................
}

// Operatorüberladung zur Ausgabe der kompletten Liste

.............. operator<<(................................) {

    ..........................................................

    ..........................................................

    ..........................................................
}
```

Aufgabe 4: (Fortsetzung)

```
    // Operatorüberladung zum Zugriff auf das Element an Position p
    // inklusive einer Fehlerbehandlung.

    ..............  operator[](.................................) {

        ..........................................................

        ..........................................................

        ..........................................................
    }
};
```

(b) Ändern Sie die Klasse `Liste` mittels Templates so, dass beliebige Datentypen gespeichert werden können. (4 Punkte)

Aufgabe 5: (Entwurfsmuster) (8 Punkte)

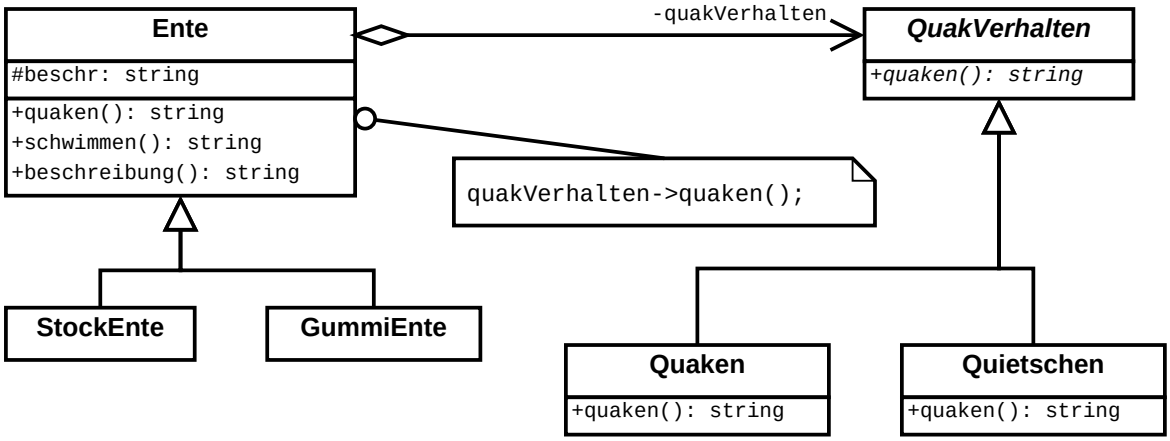
In grafischen Anwendungen, wie Zeicheneditoren, ist es möglich, komplexe Diagramme (Graphics) aus einfachen Komponenten aufzubauen. Als Primitive stehen oft Komponenten wie Circle, Rectangle und Figure zur Verfügung. Der Benutzer kann Komponenten zu komplexeren Komponenten zusammenfassen, die wiederum zu noch komplexeren Komponenten zusammengefasst werden können. Alle diese Komponenten können in vielen Fällen gleich behandelt werden. Alle haben eine

- `move()`-Methode, mit der sich das Objekt verschieben lässt.
- `area()`-Methode, die den Flächeninhalt des Objekts liefert.
- `draw()`-Methode, die das Objekt zeichnet.

Zeichnen Sie das UML-Diagramm, das die oben beschriebene Situation so modelliert, dass primitive als auch zusammengesetzte Objekte einheitlich behandeln werden können.
Mit welchem Entwurfsmuster kann diese Situation adäquat modelliert werden?

Aufgabe 6: (UML und Entwurfsmuster) (14 Punkte)

Im folgenden UML-Klassendiagramm sehen Sie eine Anwendung des Entwurfsmusters Strategie, das Ihnen aus der Vorlesung bekannt sein sollte. Eine Ente kann quaken oder quitschen.



Geben Sie eine sinnvolle, mögliche Implementierung der dargestellten Klassen an. Füllen Sie dazu den Lückentext auf den folgenden Seiten aus.

```
// Einzubindende Dateien und ähnliches

.....................................................................

.....................................................................

.....................................................................


// ********************************************************************
class QuakVerhalten {
public:

    .............. string quaken() ...................;
};

class Quaken : public QuakVerhalten {
public:
    string quaken() {
        return "quak, quak";
    }
};

class Quitschen : public QuakVerhalten {
public:
    string quaken() {
        return "quitsch, quitsch";
    }
};
```

Aufgabe 6: (Fortsetzung)

```
// ********************************************************************
class Ente {

    ............................. quakVerhalten;


.............................
    string beschr;


.............................
    Ente(string beschr);
    string quaken();
    string schwimmen();
    string beschreibung();
};

Ente::Ente(string b) {

    ..............................................................
}

string Ente::quaken() {

    return ......................................................
}

string Ente::schwimmen() {

    return ......................................................
}

string Ente::beschreibung() {

    return ......................................................
}
```

Aufgabe 6: (Fortsetzung)

```
// ********************************************************************
class StockEnte: public Ente {
public:
    StockEnte(string beschr) ..................................... {

        ..........................................................
    }
};

class GummiEnte: public Ente {
public:
    GummiEnte(string beschr) ..................................... {

        ..........................................................
    }
};


// ********************************************************************
int main(void) {
    Ente *enten[2];

    enten[0] = new StockEnte("Stockente Uschi");
    enten[1] = new GummiEnte("Gummiente Walter");

    for (int i = 0; i < 2; i++) {
        cout << enten[i]->beschreibung() << endl;
        cout << "   " << enten[i]->schwimmen() << endl;
        cout << "   " << enten[i]->quaken() << endl;
    }

    return 0;
}
```