Name, Vorname _____ Matrikelnummer _____

Klausur zur Vorlesung

Objektorientierte Anwendungsentwicklung

Krefeld, 16. März 2012

Hinweise:

- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf dieses Deckblatt. Die Aufgabenstellung und alle Klausurbögen müssen am Ende abgegeben werden.
- Alle Antworten sind auf den Aufgabenblättern einzutragen. Die Klausurbögen können Sie bei Bedarf als Schmierblatt verwenden. Schreiben Sie auf jeden Klausurbogen Ihren Namen und Ihre Matrikelnummer.
- 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
Punkte	21	4	8	15	8	15	71
erreichte Punkte							

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

_____ (Datum) _____ (2. Prüfer)

Aufgabe 1: (Quickies) (21 Punkte)
Kreuzen Sie die jeweils richtige Antwort bzw. Aussage an.

(a) Die Anweisung `return *this;` befinde sich innerhalb einer Methode. Dann gilt:

- ☐ Das aktuelle Objekt wird zurückgegeben.
- ☐ Die Adresse des aktuellen Objekts wird zurückgegeben.
- ☐ Das erste Datenelement des aktuellen Objekts wird zurückgegeben.

(b) Speicher für ein Datenelement wird angelegt, wenn

- ☐ die Klasse definiert wird.
- ☐ die Header-Datei mit der Klassendefinition inkludiert wird.
- ☐ ein Objekt vom Typ der Klasse angelegt wird.

(c) Was ist richtig?

- ☐ Referenzen werden mittels `delete` gelöscht.
- ☐ Pointer und Referenzen auf Objekte verhalten sich im Kontext der Polymorphie gleich.
- ☐ Zwischen Pointer und Referenzen gibt es überhaupt keine Unterschiede im Verhalten.
- ☐ Eine Referenz kann man mit `++` erhöhen, einen Pointer nicht.

(d) Eine Klasse heißt abstrakt, wenn ...

- ☐ ... sich niemand etwas darunter vorstellen kann.
- ☐ ... sie mindestens eine rein virtuelle Methode besitzt.
- ☐ ... sie einen generischen Algorithmus enthält.
- ☐ ... sie einen abstrakten Vektorraum aufspannt.

(e) Was ist richtig? Die Schnittstelle einer Klasse ...

- ☐ ... dient dem Datenaustausch zwischen Computern und Peripheriegeräten.
- ☐ ... besteht aus polymorphen Ringen.
- ☐ ... wird für den Ausdruck auf einem Drucker benötigt.
- ☐ ... besteht aus den `public`-Elementen einer Klasse.
- ☐ ... besteht aus zwei konzentrischen Kreisen.

Aufgabe 1: (Fortsetzung)

(f) Der Compiler fügt den Maschinencode einer inline-Funktion

- ☐ nur an die Stelle des ersten Aufrufs ein.
- ☐ an jeder Stelle ein, an der die Funktion aufgerufen wird.
- ☐ nirgendwo ein. Der Linker übernimmt diese Aufgabe.

(g) Eine Referenz auf ein Objekt ist

- ☐ die Adresse des Objektes.
- ☐ der Typ des Objektes.
- ☐ ein anderer Name für das Objekt.

(h) Angenommen, die Klasse `Measure` enthält die folgende `public`-Methode:
`bool isGreater(Measure m) const;`

Beim Aufruf der Methode `isGreater`

- ☐ verweist der Parameter `m` auf das übergebene Argument.
- ☐ erzeugt der Default-Konstruktor der Klasse `Measure` das Objekt `m`, dem dann das Argument zugewiesen wird.
- ☐ erzeugt der Copy-Konstruktor das Objekt `m` und initialisiert es mit dem übergebenen Argument.

(i) Wenn Sie einen binären Operator als Methode einer Klasse überladen, dann ist ein Objekt, für das die Methode aufgerufen wird, der

- ☐ rechte Operand der Operation.
- ☐ linke Operand der Operation.

(j) Bei einer durch eine `public`-Vererbung entstandenen abgeleiteten Klasse handelt es sich um

- ☐ einen Teil der Basisklasse.
- ☐ eine Spezialisierung der Basisklasse.
- ☐ eine Verallgemeinerung der Basisklasse.

Aufgabe 1: (Fortsetzung)

(k) Eine Teile-Ganzes-Beziehung nennt man _____ und wird im UML-Klassendiagramm wie folgt dargestellt:

(l) Was ist der Unterschied zwischen den Containern `map` und `multimap`?

(m) Was ist die Ausgabe von folgendem Programm?

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

void swap1(int a, int b) {
    int t = a;
    a = b;
    b = t;
}
void swap2(int &a, int &b) {
    int t = a;
    a = b;
    b = t;
}
int main(void) {
    int i = 2;
    int j = 4;

    swap1(i, j);
    cout << "i: " << i << ", j: " << j << endl;

    i = 2;
    j = 4;
    swap2(i, j);
    cout << "i: " << i << ", j: " << j << endl;
}
```

Aufgabe 1: (Fortsetzung)
Bitte kreuzen Sie jeweils an, ob die Aussage richtig oder falsch ist. **Achtung:** Für jede falsche Antwort wird eine richtige Antwort nicht bewertet. Für eine unbeantwortete Frage werden keine Punkte abgezogen.

(n) Ein statisches Datenelement einer Klasse muss separat definiert und initialisiert werden.

- ☐ richtig
- ☐ falsch

(o) Bei der Überladung eines Operators kann sein Vorrang geändert werden.

- ☐ richtig
- ☐ falsch

(p) Falls eine Methode über den Namen eines Objektes aufgerufen wird, handelt es sich immer um statische Bindung.

- ☐ richtig
- ☐ falsch

(q) Der Konstruktor einer abstrakten Klasse muss rein virtuell sein.

- ☐ richtig
- ☐ falsch

(r) Einem Zeiger auf eine abstrakte Basisklasse kann nur die Adresse eines Objektes vom Typ einer abgeleiteten, nicht abstrakten Klasse zugewiesen werden.

- ☐ richtig
- ☐ falsch

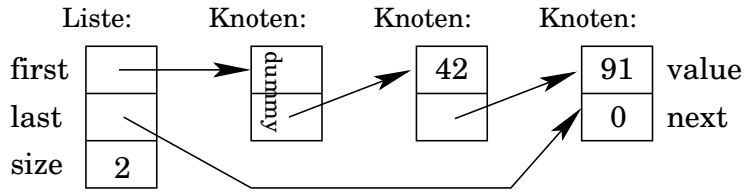
(s) Ein statisches Datenelement einer Klasse muss im Konstruktor initialisiert werden.

- ☐ richtig
- ☐ falsch

Aufgabe 2: (C++-Programmierung) (4 Punkte)
Was wird durch das nachfolgende Programm ausgegeben?

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

// ------------------------------------------------
class B {
public:
    virtual void eineMethode();
};

void B::eineMethode() {
    cout << "Hier die Basisklasse" << endl;
}

// ------------------------------------------------
class A : public B {
public:
    void eineMethode();
};

void A::eineMethode() {
    cout << "Hier die abgeleitete Klasse" << endl;
};

// ------------------------------------------------
int main() {
    try {
        throw A();
    } catch (B b) {
        b.eineMethode();
    }

    try {
        throw A();
    } catch (B& b) {
        b.eineMethode();
    }

    return 0;
}
```

Aufgabe 3: (C++-Programmierung) (8 Punkte)
Die folgende Abbildung zeigt den prinzipiellen Aufbau einer Liste, deren Implementierung unten angegeben ist.




```
struct Knoten {
    int value;
    Knoten *next;

    Knoten(int v = 0) {
        next = 0;
        value = v;
    }
};


class Liste {
protected:
    Knoten *first, *last;
    int size;


public:
    Liste();
    ~Liste();
    bool contains(int val);
};


Liste::Liste() {
    size = 0;
    first = new Knoten();
    last = first;
}


Liste::~Liste() {
    while (first != 0) {
        Knoten *k = first->next;
        delete first;
        first = k;
    }
}
```

Aufgabe 3: (Fortsetzung)
Implementieren Sie die Methode `contains(val)`, die prüft, ob der Wert `val` in der Liste vorhanden ist. (3P)

Leiten Sie von der Klasse `Liste` eine Klasse `SortierteListe` ab. (1P)

Implementieren Sie eine Methode `SortierteListe::insert(val)`, die den Wert `val` sortiert in die Liste einfügt. (4P)

Aufgabe 4: (Templates) (15 Punkte)
Entwickeln Sie eine Template-Klasse mit Namen `CArray`, die für beliebige Datentypen ein Array zur Verfügung stellt. Neben dem Konstruktur, der als Parameter die Größe des Arrays übergeben bekommt, implementieren Sie eine Methode `length()`, die die Größe des Arrays als `int`-wert zurückliefert.
Darüber hinaus soll für die Klasse der Indexoperator `[]` implementiert werden, sodass man auf ein beliebiges Element des Arrays zugreifen kann. Der Indexoperator soll dabei auch den Fehler abfangen, wenn auf einen Index kleiner 0 oder größer der Maximalgröße des Arrays zugegriffen wird. In einem solchen Fall soll die Ausnahme `COutOfBounds` geworfen werden, die Sie bitte ebenfalls implementieren. Dem Konstruktor wird ein `string` übergeben, der den Fehler beschreibt.
Bitte ergänzen Sie den nachfolgend gegebenen Code so, dass Sie ein lauffähiges Programm zur Verfügung stellen.

```
#include ..........................

#include ..........................




// ------------------------------------------------------
class COutOfBounds {









};
```

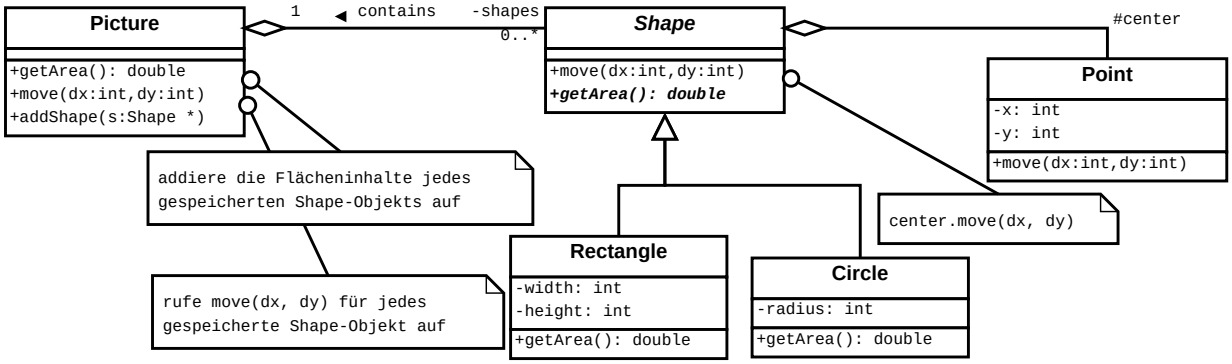
Aufgabe 4: (Fortsetzung)

```
// ------------------------------------------------------


class CArray {









};

// ------------------------------------------------------
int main() {
    CArray<int> a(3);

    try {
        a[5] = 2;
    } catch (COutOfBounds e) {
        cout << e.getFehlermeldung();
    }

    return 0;
}
```

Aufgabe 5: (UML und Entwurfsmuster) (8 Punkte)
Das Entwurfsmuster „Beobachter“ dient dazu, Änderungen an einem Objekt an andere interessierte, abhängige Objekte weiterzugeben.

Zeichnen Sie das Klassendiagramm zu diesem Entwurfsmuster inklusive Erklärungen. Beschreiben Sie ein Beispiel, wo dieses Entwurfsmuster angewendet wird.

Aufgabe 6: (C++-Programmierung) (15 Punkte)
Im folgenden UML-Klassendiagramm sehen Sie eine Anwendung des Strategie-Musters. Geben Sie eine sinnvolle, mögliche Implementierung der dargestellten Klassen an. Füllen Sie dazu den Lückentext auf den folgenden Seiten aus.




```
// ********************************************************************
class Point {

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

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

    int _x, _y;

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

    Point() {
        _x = 0;
        _y = 0;
    }

    Point(int x, int y) {
        _x = x;
        _y = y;
    }

    void move(int dx, int dy) {
        _x += dx;
        _y += dy;
    }
};
```

```
// ********************************************************************
class Shape {

.....................................................................
    Point _center;

.....................................................................
    Shape() {
        _center._x = 0;
        _center._y = 0;
    }

    Shape(Point p) {
        _center = p;
    }

    void move(int dx, int dy) {

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


    ............................. getArea() ........................
};


// ********************************************************************

class Circle ..................................................... {

.....................................................................
    int _radius;

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

    Circle(Point p, int r) ....................................... {
        _radius = r;
    }

    double getArea() {
        return _radius * _radius * 3.1415;
    }
};
```

```
// ********************************************************************
class Picture {

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

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

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

    ~Picture() {
        shapes.clear();
    }

    void addShape(Shape *s) {

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

    double getArea() {
        double sum = 0.0;

        ..................................................... iter;


        for (.................................................. ;

             .................................................. ;

             .................................................. )

            sum += ................................................ ;

        return sum;
    }
};

// ********************************************************************
int main(void) {
    Picture pic;

    pic.addShape(new Rectangle(Point(2, 0), 1, 2));
    pic.addShape(new Circle(Point(0, 1), 3));
    pic.addShape(new Rectangle(Point(0, 2), 3, 7));
    pic.addShape(new Circle(Point(1, 0), 4));

    cout << "Summe Flaecheninhalt: " << pic.getArea() << endl;
}
```