Name, Vorname _____ Matrikelnummer _____

Klausur zur Vorlesung

Objektorientierte Anwendungsentwicklung

Krefeld, 2. Juli 2012

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	7	Summe
Maximalpunkte	15	22	10	10	6	8	8	79
erreichte Punkte								

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

_____ _____
(Datum) (2. Prüfer)

Aufgabe 1: (Quickies) (15 Punkte)

(a) In welchen Fällen wird der Kopierkonstruktor automatisch aufgerufen?

(b) Welche Methoden muss eine Klasse zwingend besitzen? Hinweis: Diese Methoden werden vom Compiler automatisch erzeugt, wenn sie nicht ausdrücklich in der Klasse definiert sind.

(c) Warum muss bei Klassen mit virtuellen Funktionen der Destruktor ebenfalls virtuell sein?

(d) Was bedeutet das Schlüsselwort `this`?

(e) Was ist der Unterschied zwischen den folgenden Definitionen?

(1) `int *pi = new int(100);`

(2) `int *pi = new int[100];`

Aufgabe 1: (Fortsetzung)

(f) Was bedeutet das Schlüsselwort `static` vor einem Attribut einer Klasse?

(g) Die Voreinstellung für den Elementzugriff einer mit `struct` definierten Struktur ist

(h) Was versteht man unter einer flachen Kopie?

(i) Was ist eine abstrakte Methode? Geben Sie ein Beispiel an.

(j) Was ist eine Referenz und wie wird sie realisiert?

(k) Was versteht man unter Delegation? Erläutern Sie einen typischen Anwendungsfall, z.B. in einem Entwurfsmuster.

(l) Was ist der Unterschied zwischen den Containern `set` und `multiset`?

Aufgabe 2a: (C++-Programmierung) (6 Punkte)
Durch das nachfolgende Programm wird Folgendes ausgegeben:

```
Basis
Ableitung
```

Bitte begründen Sie die Ausgabe und erläutern Sie dabei die im Quelltext mit `/*CC*/` markierten Stellen!

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

class B {
public:
    virtual void eineMethode() {
        cout << "Basis" << endl;
    }
};

class A : public B {
public:
    void eineMethode() {
        cout << "Ableitung" << endl;
    }
};

int main() {
    try {                  /*CC*/
        throw A();         /*CC*/
    } catch (B b) {        /*CC*/
        b.eineMethode();   /*CC*/
    }
    try {
        throw A();
    } catch (B& b) {       /*CC*/
        b.eineMethode();   /*CC*/
    }
    return 0;
}
```

Aufgabe 2b: (C++-Programmierung) (6 Punkte)
Durch das nachfolgende Programm wird Folgendes ausgegeben:

```
B
D
X
Jetzt geht's los
Bye, bye!
~X
~D
~B
```

Bitte erklären Sie, wie es zu dieser Ausgabe kommt!

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

class B {
public:
    B() {cout << "B\n";}
    ~B() {cout << "~B\n";}
};

class D : public B {
public:
    D() {cout << "D\n";}
    ~D() {cout << "~D\n";}
};

class X {
private:
    D d;
public:
    X() {cout << "X\n";}
    ~X() {cout << "~X\n";}
};

int main() {
    X x;
    cout << "Jetzt geht's los" << endl;
    cout << "Bye, bye!" << endl;
    return 0;
}
```

Aufgabe 2c: (C++-Programmierung) (10 Punkte)
Der nachfolgende Quelltext ist fehlerhaft. Die Zeilen, in denen sich Fehler befinden, sind mit entsprechenden Kommentaren markiert. Erläutern Sie jeweils den Fehler und geben Sie – wenn möglich – einen Lösungsvorschlag an!

```
#include <iostream>
using namespace;                    // 1 Fehler

class C {
protected:
    int a;

public:
    void f();

    C* g() {
        C c = new C;                // 1 Fehler
        return c;
    }

    virtual C() {                   // 1 Fehler
        a = b;                      // 1 Fehler
    }
};


class D : public C {
public:
    int b;
    b = 0;                          // 1 Fehler
}                                   // 1 Fehler

void C::f(int b) {                  // 1 Fehler
    a = 17;
}

int main() {
    C x;
    cout << x << x.g() << this;     // 2 Fehler
    return x.a;                     // 1 Fehler
}
```

Aufgabe 3: (C++-Programmierung) (10 Punkte)
Betrachten Sie eine Klasse `Liste`, die beliebig viele `int`-Werte speichern kann. Die Werte können mittels `insert` eingefügt werden.

```
class Liste {
protected:
    int *values, size, pos;

    void resize() {
        int *tmp = new int[2 * size];

        for (int i = 0; i < size; i++)
            tmp[i] = values[i];
        delete[] values;
        values = tmp;
        size *= 2;
    }

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

    ~Liste() {
        delete[] values;
    }

    void insert(int val) {
        if (size == pos)
            resize();
        values[pos++] = val;
    }
};
```

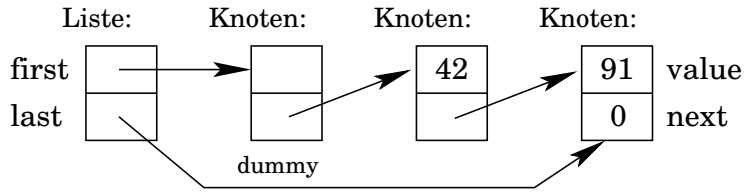
Deklarieren und definieren Sie eine abgeleitete Klasse `ExtListe`, die folgende Methoden enthält: (1 Punkt)

- `find(val)` liefert die Position, an der `val` im Array `values` gespeichert ist, ansonsten ist der Rückgabewert -1. (2 Punkte)
- `insert` soll so überschrieben werden, dass keine zwei gleichen Werte abgespeichert werden können. (2 Punkte)
- `toString()` liefert eine Ausgabe des Listeninhalts als `string`. (2 Punkte)
- `remove(val)` entfernt den Wert `val` aus der Liste. (3 Punkte)

Aufgabe 3: (Fortsetzung)

Aufgabe 4: (C++-Programmierung) (10 Punkte)

Die folgende Abbildung zeigt den prinzipiellen Aufbau einer einfach verketteten Liste.



Betrachten Sie die folgenden Klassen `Knoten` und `Queue`.

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

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

class Queue {
private:
    Knoten *first, *last;

public:
    Queue() { initialize(); }

    void initialize(void) {
        first = new Knoten();
        last = first;
    }

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

    void append(int value) {
        Knoten *k = new Knoten(value);
        last->next = k;
        last = k;
    }
};
```

Aufgabe:

- Ändern Sie die Klassen mittels Templates so, dass beliebige Datentypen gespeichert werden können. (2 Punkte)

Aufgabe 4: (Fortsetzung)
Ergänzen Sie die ursprüngliche Klasse `Queue` um folgende Methoden.

- Implementieren Sie den Copy-Konstruktor, so dass eine tiefe Kopie einer `Queue` erstellt wird. (2 Punkte)

- Überladen Sie den Ausgabe-Operator `<<`. Beachten Sie, dass dazu eventuell `friend`-Deklarationen notwendig sind. (2 Punkte)

- Implementieren Sie eine Methode `remove(val)`, die den Wert `val` aus der Queue entfernt. Falls der Wert nicht in der Queue vorhanden ist, soll eine Exception geworfen werden. (4 Punkte)

Aufgabe 5: (C++-Programmierung) (6 Punkte)
Betrachten Sie folgendes Programm:

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

int main() {
    vector<int> v(6);

    for (unsigned int i = 0; i < v.size(); i++)
        v[i] = i * 2;

    vector<int>::iterator iter = find(v.begin(), v.end(), 6);
    if (iter != v.end())
        cout << *iter << " gefunden bei " << iter - v.begin() << endl;
    else cout << "6 NICHT gefunden" << endl;
}
```

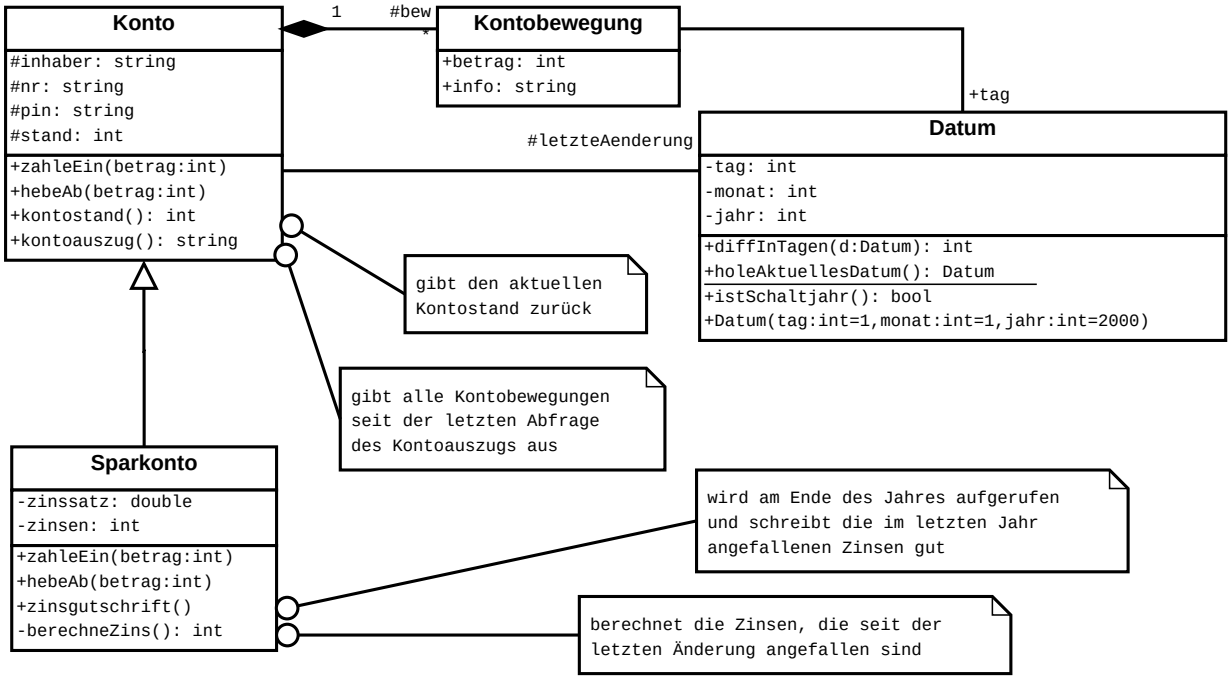
Was ist die Ausgabe des obigen Programms?

Wie könnte eine Implementierung der Klasse `iterator` aussehen?

Aufgabe 6: (UML und Entwurfsmuster) (8 Punkte)
Das Entwurfsmuster „Dekorierer“ ist eine flexible Alternative zur Unterklassenbildung, um eine Klasse um zusätzliche Funktionalitäten zu erweitern.

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

Aufgabe 7: (C++-Programmierung) (8 Punkte)
Geben Sie eine Deklaration der Klassen entsprechend des nachfolgenden UML-Klassendiagramms an. Es ist hier nur die Header-Datei anzugeben, die Methoden sollen nicht implementiert werden.



Welche Methoden sind als `virtual` zu deklarieren, welche Methoden sind abstrakt?

Aufgabe 7: (Fortsetzung)