Name, Vorname | Matrikelnummer

Klausur zur Vorlesung

Objektorientierte Anwendungsentwicklung

Krefeld, 2. Juli 2014

Hinweise:

- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf dieses Deckblatt. Tragen Sie alle Antworten auf den Aufgabenblättern ein.
- Benutzen Sie kein mitgebrachtes Papier. Zusätzliches Schreibpapier in Form von Klausurbögen können Sie bei Bedarf von den Betreuern anfordern.
- Die Aufgabenstellung und alle Klausurbögen müssen am Ende abgegeben werden.
- Die Bearbeitung der Aufgaben darf nur mit dokumentenechten Stiften erfolgen, verwenden Sie *keinen* Bleistift. Verwenden Sie keinen Stift, der in rot schreibt.
- Falls Sie unterschiedliche Lösungen für eine Aufgabe abgeben, 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	10	10	18	12	7	16	88
erreichte Punkte								

(Note) | (Datum) | (1. Prüfer)

(Datum) | (2. Prüfer)

Aufgabe 1: (Quickies) (15 Punkte)

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

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

(c) Was bezeichnet man als Schnittstelle einer Klasse?

(d) Was bedeutet das Schlüsselwort `this`?

(e) Wo wird ein statisches Datenelement einer Klasse initialisiert? Geben Sie als Beispiel ein C++-Programmfragment an!

Aufgabe 1: (Fortsetzung)

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

(g) Was ist die Voreinstellung für den Elementzugriff einer Struktur (`struct`)?

(h) Was versteht man unter einer flachen Kopie?

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

(j) Was ist der Unterschied zwischen Überschreiben und Überladen von Methoden?

(k) Deklarieren und initialisieren Sie eine Referenz auf einen Integer `n`.

```
int n;

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

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

Aufgabe 2: (Modellierung) (10 Punkte)

In der Vorlesung haben Sie das Quadrat-Rechteck-Problem und verschiedene Lösungsansätze kennengelernt.

(a) In der Vorlesung wurde die Lösung *Quadrat ist ein Rechteck* vorgestellt:

- Erläutern Sie diese Lösung mit Hilfe eines UML-Diagramms, das die Klassen `GeomFigur`, `Rechteck` und `Quadrat` enthält. (1 Punkt)

- Bei der Diskussion dieser Lösung wurde eine „gefährliche Methode“ identifiziert. Benennen Sie diese Methode und erläutern Sie, warum diese Methode gefährlich genannt wird. (1 Punkt)

(b) Beschreiben Sie die Lösung *technische Vererbung*, auch *Vererbung aus Bequemlichkeit (inheritance by convenience)* genannt, mit Hilfe eines UML-Diagramms, das die Klassen `Rechteck` und `Quadrat` enthält. (1 Punkt)

Geben Sie zusätzlich die widersinnige Aussage an, die sich aus dieser Lösung ableiten lässt? (1 Punkt)

Aufgabe 2: (Fortsetzung)

(c) In der Vorlesung wurde auch die Lösung *Quadrat als Eigenschaft von Rechteck* vorgestellt:

- Erläutern Sie diese Lösung mit Hilfe eines UML-Diagramms, das die Klassen `GeomFigur` und `Rechteck` enthält. (2 Punkte)

- Geben Sie den C++-Code des `if`-Konstruktes an, das bei dieser Lösung in der Funktion `void Rechteck::resize(int dx, int dy)` benötigt wird. (1 Punkt)

(d) In der Modellierung verwendet man das Prinzip der Delegation.

- Bitte erläutern Sie in einem Satz, was bei der Modellierung unter der Delegation verstanden wird. (1 Punkt)

- Geben Sie in Form eines UML-Diagramms oder in Form von C++-Codefragmenten an, wie die Delegation zur Lösung des Quadrat-Rechteckproblems verwendet wird. (2 Punkte)

Aufgabe 3: (C++-Programmierung) (10 Punkte)

In dieser Aufgabe soll ein Heap implementiert werden. Ergänzen Sie den folgenden Code-Ausschnitt zu einem sinnvollen, compilierbaren C++-Code. Die Methoden `increase()`, `upHeap()` und `downHeap()` können Sie als gegeben betrachten. Das Vertauschen der Elemente in `upHeap()` und `downHeap()` erfolgt in der Methode `swap()`. Denken Sie jeweils an eine geeignete Ausnahmebehandlung.

```
class Heap {
public:
    Heap() {

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

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

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

    ~Heap() {

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

    bool isEmpty() {

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

    void insert(int val) {




        if (isFull())
            increase();

        _values[_last] = val;
        _last += 1;
        upHeap(_last - 1);
    }

    int minimum() {

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

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

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

Aufgabe 3: (Fortsetzung)

```
    void extractMin() {

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

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

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

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

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


protected:
    int _size;
    int _last;
    int *_values;



    bool isFull() {

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

    void swap(int p1, int p2) {



        int tmp = _values[p1];

        _values[p1] = _values[p2];
        _values[p2] = tmp;







    }

    void increase();
    void downHeap(int pos);
    void upHeap(int pos);
};
```

Aufgabe 4: (C++-Programmierung) (18 Punkte)

Betrachten Sie noch einmal die Klasse `Heap` aus Aufgabe 3.

(a) Ändern Sie direkt im Aufgabentext die Klasse `Heap` mittels Templates so, dass verschiedene Datentypen gespeichert werden können. (2 Punkte)

(b) Ändern Sie direkt im Aufgabentext die Methode `insert()` so, dass kein Wert mehrfach im Heap gespeichert wird. Sie können dazu die Methode `find()` aus Aufgabenteil 4(d) nutzen. (2 Punkte)

(c) Ergänzen Sie die Klasse `Heap` um eine Methode `void decreaseKey(T alt, T neu)`, die den gespeicherten Wert `alt` auf den neuen, kleineren Wert `neu` setzt. Auch hier kann die Methode `find()` aus Aufgabenteil 4(d) genutzt werden.

Falls der Schlüssel nicht gefunden wird, oder der neue Wert größer oder gleich dem alten Wert ist, soll eine Exception geworfen werden. Achten Sie auch darauf, dass im Heap keine Werte mehrfach gespeichert werden sollen. (8 Punkte)

Aufgabe 4: (Fortsetzung)

(d) Erweitern Sie den Heap um eine Methode `int find(T val)`, die zu einem gegebenen Wert `val` dessen Position (Index) im Array `_values` bestimmt.

Damit der Wert `val` nicht gesucht werden muss, führen Sie eine Map aus der Standard Template Library ein, in der zu jedem Wert die entsprechende Position im Heap gespeichert wird. Ergänzen Sie weiter den Programmcode aus Aufgabe 3 so, dass die Map in der Methode `insert()` entsprechend gefüllt wird und in der Methode `swap()` entsprechend aktualisiert wird. (6 Punkte)

Aufgabe 5: (UML und Entwurfsmuster) (12 Punkte)

(a) In der Vorlesung haben Sie das Beobachter-Muster kennengelernt. Es dient dazu, Änderungen an einem Objekt an andere interessierte, abhängige Objekte weiterzugeben.

Zeichnen Sie das entsprechende UML-Diagramm und erläutern Sie an einem Beispiel, wie und wo das Beobachter-Muster angewendet werden kann. (8 Punkte)

Aufgabe 5: (Fortsetzung)

(b) Die Entwurfsmuster *Strategie* und *Zustand* sind vom UML-Klassendiagramm her gleich. Erklären Sie den wesentlichen Unterschied dieser beiden Muster. (2 Punkte)

(c) Die Entwurfsmuster *Kompositum* und *Dekorierer* nutzen Aggregation. Erklären Sie den wesentlichen Unterschied dieser Nutzung. (2 Punkte)

Aufgabe 6: (UML und C++-Programmierung) (7 Punkte)

Betrachten Sie die folgende `main`-Funktion, die die aus der Vorlesung und dem Praktikum bekannte Klasse `Liste` verwendet:

```
int main(void) {
    Liste<int> l;

    for (int j = 6; j < 26; j++)
        l.append(j);

    Liste<int>::Iter iter;

    iter = suchen(l.start(), l.ende(), 12);
    if (iter != l.ende())
        cout << "12 gefunden" << endl;
    else cout << "12 nicht gefunden" << endl;
}
```

Was ist die Ausgabe des Programms?

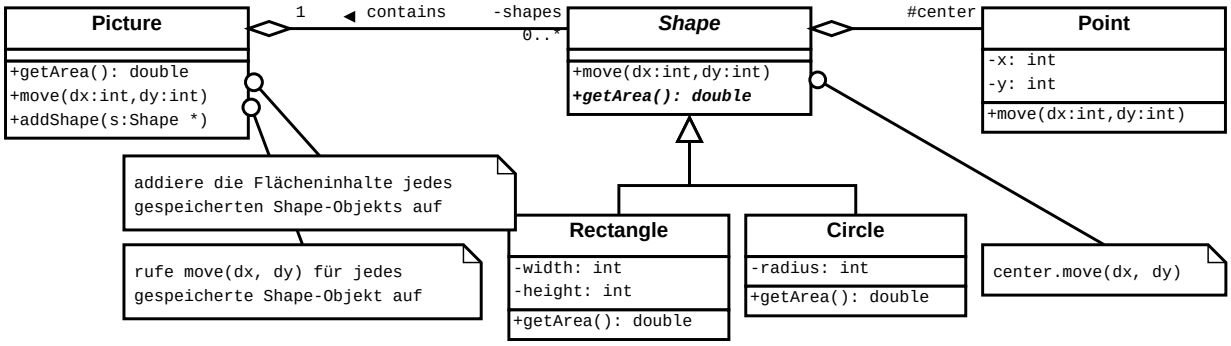
Aufgabe 6: (Fortsetzung)

Vervollständigen Sie den nachfolgenden Code zu einer Implementierung der Klasse `Iter` für die Klasse `Liste`.

```
template <typename T>
class Liste {
    .....

    class Iter {
    private:




    public:
        // -------------------------------------------------------
        Iter(T *cursor = 0) {




        }

        // -------------------------------------------------------
        bool operator!=(Iter iter) const {




        }

        // -------------------------------------------------------
        T& operator*() const {




        }

        // -------------------------------------------------------
        Iter operator++(int) {     // postfix-Operator !







        }
    };     // Ende der Klasse Iter
};     // Ende der Klasse Liste
```

Aufgabe 7: (UML und C++-Programmierung) (16 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.



Achten Sie auf virtuelle Methoden, rein virtuelle Methoden sowie `friend`-Deklarationen.

```
// ************************************************************
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;
    }
};
```

Aufgabe 7: (Fortsetzung)

```
// ***************************************************************
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;
    }
};
```

Aufgabe 7: (Fortsetzung)

```
// ************************************************************
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;
}
```