

Wintersemester 2006/07	Zahl der Blätter: 16 Blatt Nr: 1
Fachbereich: Informationstechnik	Semester: IT3A/B
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Hilfsmittel: keine elektronischen Hilfsmittel	Zeit: 90 min
Name:	Matrikel-Nr.:

Hinweis: Der auf den Blättern jeweils freigelassene Raum reicht im Allgemeinen vollständig für die stichwortartige Beantwortung der Fragen, bzw. für die Lösungen aus. Tragen Sie daher auf **jedem** Blatt Ihren Namen und Ihre Matrikelnummer ein und nutzen Sie diese Blätter zur Abgabe Ihrer Antworten und Lösungen.

Aufgabe 1: Allgemeine Fragen

(ca. 20 Min.)

Bitte beurteilen Sie die folgenden allgemeinen Aussagen. Machen Sie jeweils ein Kreuzchen in der Spalte „wahr“ oder „falsch“. Begründen Sie jeweils Ihre Wahl.

Aussage	wahr	falsch
Dynamische Casts sind als sicher zu betrachten. Begründung:		
Eine Referenz verhält sich genauso wie ein Pointer. Begründung:		
Die Initialisierungen in der Initialisierungsliste eines Konstruktors könnte man immer auch im Rumpf des Konstruktors durch Zuweisungen realisieren. Begründung:		

Wintersemester 2006/07	Blatt Nr: 2 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
<p>Bei dem Template</p> <pre>template <class T1, class T2> void product (T1 p1, T2 p2) { ... }</pre> <p>können T1 und T2 denselben Typ repräsentieren. Begründung:</p>		
<p>Die Funktion</p> <pre>int fkt(int a, int b=0, int c){ return a+b+c; }</pre> <p>wird vom Compiler erfolgreich kompiliert. Begründung:</p>		
<p>Mit new können Objekte dynamisch auf dem Stack allokiert werden. Begründung:</p>		
<p>Der Begriff „Dynamisches Binden“ umfasst auch das „Overloading“. Begründung:</p>		

Wintersemester 2006/07	Blatt Nr: 3 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
<p>Eine virtuelle Methode kann mit einer „normalen“ Methode, die den gleichen Namen hat, überladen werden.</p> <p>Begründung:</p>		
<p>Alle Ausnahmen sind Objekte der Klasse <code>exception</code>.</p> <p>Begründung:</p>		
<p>Um 3 Objekte der Klasse <code>O</code> auszugeben, kann man einen Operator als Funktion wie folgt definieren:</p> <pre><code>ostream & operator<< (ostream &, O &, O &, O &);</code></pre> <p>Begründung:</p>		

Wintersemester 2006/07	Blatt Nr: 4 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aufgabe 2: Klassendefinition und Vererbung

(ca. 15 Min.)

2.1 Eine Klasse **KFZ** mit folgenden Elementen soll definiert werden.

- a) Instanzvariable **modell** vom Typ **string**, die die Marke und das Modell des KFZ angibt, z.B. Mercedes SLK 300.
- b) Eine konstante Instanzvariable **nr** vom Typ **int**, die für jedes Objekt der Klasse eindeutig sein soll.
- c) Die Klassenvariable **maxnr**, die Sie nutzen, um die Instanzvariable **nr** eindeutig zu setzen.
- d) Eine konstante Instanzvariable **tankkapazitaet** vom Typ **float**. Diese Variable gibt an, wieviele Liter Treibstoff maximal in den Tank des KFZ passen.
- e) Eine Instanzvariable **tankfuellung** vom Typ **float**. Diese Variable gibt an, wieviele Liter sich gerade im Tank befinden.
- f) Für jede der obigen Variablen eine Methode, um die jeweilige Variable auszulesen und zurückzuliefern.
- g) Eine Methode **setModell**, um den Wert der Variablen **modell** zu verändern.
- h) Eine Methode **tanken**, die z.B. mit **tanken(10.5)** die Tankfüllung um 10,5 Liter erhöht. Wenn versucht wird, zuviele Liter zu tanken, wird eine Warnmeldung ausgegeben und die **tankfuellung** bis zum Maximum erhöht.
- i) Ein Konstruktor mit Parametern für die Instanzvariablen **modell** und **tankkapazitaet**. Beide Parameter sollen Defaultwerte bekommen. Der Tank soll bei einem neu konstruierten KFZ leer sein.

2.2 Die Klasse **PKW** soll als abgeleitete Klasse der Klasse **KFZ** definiert werden. Die Klasse **PKW** soll die Variable **sitzplaetze** haben. Sie gibt an, wieviel Personen befördert werden können. Die Klasse hat natürlich auch einen geeigneten Konstruktor.

2.3 Auch die Klasse **LKW** soll als abgeleitete Klasse der Klasse **KFZ** definiert werden und hat die Instanzvariable **gesamtgewicht**. Sie gibt das zulässige Gesamtgewicht an. Die Klasse hat natürlich auch einen geeigneten Konstruktor.

2.4 Die Klasse **Transporter** soll sowohl von **PKW** als auch von **LKW** abgeleitet werden. Sorgen Sie dafür, dass ein Transporter nur einen Tank hat und nicht zwei und einen geeigneten Konstruktor hat.

Ergänzen Sie das Programmgerippe auf den folgenden Seiten. Schützen Sie die Datenelemente vor Zugriffen durch klassenfremde Methoden; erlauben Sie aber abgeleiteten Klassen den Zugriff. Bei Methoden und Konstruktoren fügen Sie nur die **Prototypen** ein.

Wintersemester 2006/07	Blatt Nr: 5 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Prototypen der Klassen KFZ, PKW, LKW und Transporter (Aufgabe 2)

```
#include
```

```
class KFZ  
{
```

```
    // Variablen:
```

```
    // Prototypen aller Methoden:
```

```
    // Prototyp des Konstruktors:
```

```
};
```

Wintersemester 2006/07	Blatt Nr: 6 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

```
class PKW
{
    // Instanzvariablen:

    // Prototyp des Konstruktors:

};

class LKW
{
    // Instanzvariablen:

    // Prototyp des Konstruktors:

};

class Transporter
{
    // Prototyp des Konstruktors:

};
```

Wintersemester 2006/07	Blatt Nr: 7 / 16
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aufgabe 3: Implementierung der Methoden der Klassen

(ca. 15 min)

Programmieren Sie bitte hier und auf der folgenden Seite außerhalb der Klassen **KFZ**, **PKW**, **LKW** und **Transporter** die angegebenen Objekte aus:

`// Konstruktor der Klasse KFZ:`

`// Konstruktor der Klasse PKW:`

`// Konstruktor der Klasse LKW:`

`// Konstruktor der Klasse Transporter:`

Wintersemester 2006/07	Blatt Nr:	8 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

// Methode getModell:

// Methode getMaxnr:

// Methode setModell:

// Methode tanken

Wintersemester 2006/07	Blatt Nr:	9 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 4: Polymorphie

(ca. 10 min)

Analysieren Sie das nachfolgende Programm und schreiben Sie die Ausgabe des Programms auf.

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

class A
{
protected:
    int id;
public:
    A(int id) : id(id) {}
    virtual void print() = 0;
};

class B
{
protected:
    string name;
public:
    B(string name) : name(name) {}
    string getName()
    {
        return "B: name = " + name;
    }
};

class C
{
protected:
    char c;
public:
    C(char c) : c(c){}
    void print()
    {
        cout << "C-Print: c = " << c << endl;
    }
};
```

Wintersemester 2006/07	Blatt Nr:	10 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```

class AB : public B, public A
{
public:
    AB(int id, string name) : A(id), B(name) {}
    virtual void print()
    {
        cout << "AB-Print: id = " << id << ", " << getName() <<
endl;
    }
};

class ABC : public AB, public C
{
public:
    ABC(int id, string name, char c) : AB(id, name), C(c) {}
    void print()
    {
        cout << "ABC-Print: " << endl;
        AB::print();
        C::print();
        cout << endl;
    }
    string getName() const
    {
        return "ABC: name = " + name;
    }
};

void print(A * aptr)
{
    aptr->print();
}

int main(void)
{
    ABC abc(1, "Objekt 1", '1');
    abc.print();
    print(&abc);
    AB * abptr = &abc;
    AB ab = *abptr;
    abptr->print();
    ab.print();
    cout << abptr->getName() << endl;
    cout << ((ABC *)abptr)->getName() << endl;
}

```

Wintersemester 2006/07	Blatt Nr:	11 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Ausgabe des Programms:

Wintersemester 2006/07	Blatt Nr:	12 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 5: Klassendefinition und STL

(ca. 13 Min.)

Die Klassen **Datum**, **Strecke** und **Fahrkarte** sind wie folgt vorgegeben

```

class Datum {
public: int tag,monat,jahr;
        Datum (int t, int m, int j);
};

class Strecke {
        string startort;
        string zielort;
        float preis;

public:
        Strecke (string s, string z, float p);
        string getZielort ();
        void druckeStrecke ();
};

class Fahrkarte {
        vector<Strecke> route;
        Datum ausgestellt;
        float preis;

public:
        void buchenStrecke (string start, string ziel, float preis);
        void fertigstellenKarte (Datum d);
        void druckeKarte ();

};

```

Programmieren Sie bitte nun im Folgenden außerhalb der Klasse **Fahrkarte** die angegebenen Methoden aus.

Wintersemester 2006/07	Blatt Nr:	13 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Lösungsblatt für die Aufgabe 5

5.1 Die Methode `buchenStrecke` bucht eine (Teil-)Strecke zu einer Fahrkarte hinzu. Als Parameter werden der Start- und der Zielort der Teilstrecke benötigt, sowie der Preis dieser Strecke. Die Methode prüft, ob der Zielort der zuletzt eingetragenen Teilstrecke mit dem als Parameter gegebenen Startort übereinstimmt. Falls nicht, wird die neue Strecke nicht in die Route aufgenommen, sondern eine Ausnahme der Klasse `exception` geworfen. Als Information ist der String "Start ungleich Ziel" dem Exception-Objekt mitzugeben.

5.2 Die Methode `fertigstellenKarte` hat einen Parameter vom Typ `Datum`. Sie berechnet den Gesamtpreis der Fahrkarte aus den Preisen der Teilstrecken, übernimmt das Ausstellungsdatum von dem Parameter und druckt die Karte (mit Hilfe der Methode `druckeKarte`).

Wintersemester 2006/07	Blatt Nr:	14 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 6: Exception-Handler

(ca. 5 Min.)

Ergänzen Sie nun im folgenden Hauptprogramm die Ausnahme-Behandlung. Benutzt werden die Klassen aus der vorherigen Aufgabe. Führen Sie einen Exception-Handler für die Ausnahme der Methode `buchenStrecke()` ein, sowie einen Default-Handler, der alle möglichen Ausnahmen fängt. Im Default-Handler geben Sie eine sinnvolle Warnmeldung aus, im anderen Handler geben Sie den im Exception-Objekt gespeicherten String aus.

```
int main (void) {  
  
    Fahrkarte billet;  
    Datum heute (15,12,2006);  
  
    billet.buchenStrecke ("Karlsruhe", "Stuttgart", 7.80f);  
    billet.buchenStrecke ("Stuttgart", "Esslingen", 2.20f);  
    billet.fertigstellenKarte (heute);  
  
    return 0;  
};
```

Wintersemester 2006/07	Blatt Nr:	15 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 7: Zugriffsschutz bei Vererbung

(ca. 12 Min.)

Gegeben sind die folgenden Klassendefinitionen:

```

class basic
{
    private:    int verborgen;

    protected: int geschuetzt;

    public:    int oeffentlich;

    void print(void);

    basic (int priv, int prot, int pub);
};

class sub : private basic
{
    int sublocal;
    public:

    void print (void);

    sub (int priv, int prot, int pub, int loc);
};

class subsub : public sub
{
    int subsublocal;
    public:

    void print (void);

    subsub (int priv, int prot, int pub, int loc);
};

```

7.1 Ergänzen Sie auf der nächsten Seite die separaten Implementierungen der Methoden `print()` der Klassen `sub` und `subsub`. Geben Sie darin alle möglichen eigenen und geerbten Datenelemente aus, ohne Methoden der Basisklasse zu benutzen.

Wintersemester 2006/07	Blatt Nr:	16 / 16
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Lösung der Aufgabe 7.1:

```
// Methode print der Klasse sub:
void sub::print(void) {
```

```
};
```

```
// Methode print der Klasse subclass:
void subclass::print(void) {
```

```
};
```

7.2 Betrachten Sie das folgende Hauptprogramm mit Objekten der Klassen **sub** und **subclass**. Kommentieren Sie alle Zeilen aus, die fehlerhafte Zugriffe auf Attribute der jeweiligen Objekte enthalten.

```
int main (void)
{
    sub    sub_obj (11, 12, 13, 14);
    subclass  ssub_obj (21, 22, 23, 24);

    cout << sub_obj.verborgen;
    cout << sub_obj.geschuetzt;
    cout << sub_obj.oeffentlich;

    cout << ssub_obj.verborgen;
    cout << ssub_obj.geschuetzt;
    cout << ssub_obj.oeffentlich;

    return 0;
}
```