

Sommersemester 2007	Zahl der Blätter: 15 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
Das Objekt <code>p2</code> der Klasse <code>Person</code> wird mit dem Objekt <code>p1</code> dieser Klasse initialisiert: <code>Person p1, p2 = p1;</code> Dazu wird der Zuweisungsoperator verwendet. Begründung:		
Wird beim Aufruf einer Methode mit Defaultargumenten ein Argument weggelassen, dann dürfen keine weiteren Argumente weggelassen werden. Begründung:		
Statische Datenelemente einer Klasse können auch <code>private</code> -Elemente sein. Begründung:		

Sommersemester 2007	Blatt Nr: 2 / 15
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
<p>Das Template</p> <pre>template <class T> T min (T a, T b);</pre> <p>definiert eine Templatefunktion. Begründung:</p>		
<p>Mit</p> <pre>bool fkt(int&);</pre> <p>wird eine überladene Funktion von <code>int fkt(int&);</code> deklariert. Begründung:</p>		
<p>Eine Klasse enthalte eine konstante und eine nicht konstante Version einer Methode. Die nicht konstante Version kann dann als überladene Version der ersteren angesehen werden. Begründung:</p>		
<p>Polymorphe Vererbungshierarchien verwenden „Dynamisches Binden“. Begründung:</p>		

Sommersemester 2007	Blatt Nr: 3 / 15
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
Objekte einer polymorphen Klasse besitzen einen Zeiger auf die VMT. Begründung:		
Die Methode <code>at()</code> der Klasse <code>string</code> kann eine Ausnahme auslösen. Begründung:		
Um ein Objekt der Klasse <code>Bruch</code> in einen <code>double</code> -Wert zu casten, ist ein Operator in der Klasse <code>Bruch</code> mit der folgenden Schnittstelle zu implementieren: <code>operator double();</code> Begründung:		

Sommersemester 2007	Blatt Nr:	4 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 2: Klassendefinition und Vererbung

(ca. 17 Min.)

2.1 Eine Klasse `schiff` mit folgenden Elementen soll definiert werden.

- a) Instanzvariable `name` vom Typ `string`, die den Namen des Schiffes speichert
- b) eine konstante Instanzvariable `maxAnzPass` vom Typ `int`, die angibt, wieviele Passagiere maximal auf das Schiff passen
- c) die Instanzvariable `anzPass` vom Typ `int`, die speichert, wieviele Passagiere sich gerade auf dem Schiff befinden
- d) eine Klassenvariable `gesMaxAnzPass` vom Typ `int`. Diese Variable gibt an, wieviele Passagiere alle Schiffe zusammen maximal aufnehmen können.
- e) einen Konstruktor, der erlaubt, die Instanzvariablen eines Schiffes zu initialisieren. Die Instanzvariable `anzPass` soll dabei per default auf 0 gesetzt werden.
- f) einen Destruktor
- g) eine Instanzmethode `einsteigen()` mit einem Parameter. Der Parameter gibt an, wieviele Passagiere auf das Schiff zusteigen wollen. Es werden nur soviele Passagiere auf das Schiff gelassen, wie die maximale Anzahl von Passagieren erlaubt.
- h) eine Methode `getAnzPass()`, die die den Wert der Instanzvariablen `anzPass` zurückliefert
- i) eine Methode `getGesMaxAnzPass()`, die die Variable `gesMaxAnzPass` zurückliefert
- j) eine Methode `print()`, die die Instanzvariablen eines Schiffes am Bildschirm ausgibt
- k) einen Ausgabeoperator `<<`, der den Namen eines Schiffes an den Ausgabestream übergibt (Beispiel: `cout << schiff << endl;`)

2.2 Die Klasse `Motorschiff` soll als abgeleitete Klasse der Klasse `schiff` definiert werden. Die Klasse `Motorschiff` soll die Instanzvariable `motorleistung` vom Typ `int` haben. Die Klasse hat einen geeigneten Konstruktor und eine Methode `print()`, die die Instanzvariablen eines Motorschiffes am Bildschirm ausgibt.

2.3 Auch die Klasse `segelschiff` soll als abgeleitete Klasse der Klasse `schiff` definiert werden und hat die Instanzvariable `segelflaeche`. Die Klasse hat einen geeigneten Konstruktor und eine Methode `print()`, die die Instanzvariablen eines Segelschiffes am Bildschirm ausgibt.

Sommersemester 2007	Blatt Nr: 5 / 15
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

2.4 Die Klasse `Motorsegelschiff` soll sowohl von `Motorschiff` als auch von `Segelschiff` abgeleitet werden. Die Klasse `Motorsegelschiff` soll keine Variablen doppelt erben.

Ergänzen Sie das folgende Programmgerippe. Schützen Sie die Datenelemente vor Zugriffen durch klassenfremde Methoden; erlauben Sie aber abgeleiteten Klassen den Zugriff.

Prototypen der Klassen (Aufgabe 2)

```
#include
```

```
// Klasse Schiff  
class Schiff  
{
```

```
    // Instanzvariablen
```

```
    // Prototyp des Konstruktors
```

```
    // Prototyp des Destruktors
```

```
    // Prototyp der Methode einsteigen()
```

```
    // Prototyp der Methode getAnzPass()
```

Sommersemester 2007	Blatt Nr:	6 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```
// Prototyp der Methode print()

// Prototyp der Methode getGesMaxAnzPass()

// Prototyp für den Ausgabeoperator,
// der den Namen eines Schiffes ausgibt

};

// Klasse Motorschiff
class Motorschiff
{
    // Instanzvariablen

    // Prototyp des Konstruktors

    // Prototyp der Methode print()
    void print() const;
};

// Klasse Segelschiff
class Segelschiff
{
    // Instanzvariablen
```

Sommersemester 2007	Blatt Nr:	7 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```
// Prototyp des Konstruktors

// Prototyp der Methode print()
void print() const;
};

// Klasse Motorsegelschiff
class Motorsegelschiff
{

// Prototyp des Konstruktors

// Prototyp der Methode print()
void print() const;
};
```

Sommersemester 2007	Blatt Nr:	8 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 3: Implementierung der Methoden der Klassen

(ca. 19 min)

Programmieren Sie bitte hier und auf den folgenden Seiten außerhalb der Klassen **Schiff**, **Motorschiff**, **Segelschiff** und **Motorsegelschiff** die angegebenen Elemente aus:

```
// Definition des Konstruktors der Klasse Schiff
```

```
// Definition des Destruktors der Klasse Schiff
```

```
// Definition der Methode einsteigen()
```

```
// Definition der Methode getAnzPass()
```

Sommersemester 2007	Blatt Nr:	9 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```
// Definition der Methode print()
void Schiff::print() const
{
    cout << "Name:          " << name << endl;
    cout << "MaxAnzPass: " << maxAnzPass << endl;
    cout << "AnzPass:      " << anzPass << endl;
}

// Definition der Methode getGesMaxAnzPass()

// Ausgabeoperator, der den Namen eines Schiffes ausgibt

// Definition des Konstruktors der Klasse Motorschiff

// Annahme: Der Konstruktor der Klasse Segelschiff ist analog
// zu dem von Motorschiff definiert. Sie brauchen den Konstruktor
// hier nicht zu definieren.

// Definition der Methode print() fuer Motorschiff
```

Sommersemester 2007	Blatt Nr:	10 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```
// Definition der Methode print() fuer Segelschiff
```

```
// Definition des Konstruktors der Klasse Motorsegelschiff
```

```
// Definition der Methode print() fuer Motorsegelschiff
```

```
// gesMaxAnzPass initialisieren
```

Sommersemester 2007	Blatt Nr:	11 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 4: Polymorphie

(ca. 6 min)

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

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

class A
{
protected:
    char a;
public:
    A(char a) : a(a) {}
    virtual void print(bool=true) = 0;
};

class B : public A
{
protected:
    char b;
public:
    B(char a, char b) : A(a), b(b) {}
    virtual void print(bool e=true)
    {
        cout << a << " ";
        cout << b << " ";
        if (e) cout << endl;
    }
};

class C : public B
{
protected:
    char c;
public:
    C(char a, char b, char c) : B(a, b), c(c){}
    void print(bool e=true)
    {
        B::print(false);
        cout << c;
        if (e) cout << endl;
    }
};
```

Sommersemester 2007	Blatt Nr:	12 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```
int main(void)
{
    B b_obj('b', 'b');
    C c_obj('c', 'c', 'c');
    A * a_arr[2] = { &b_obj, &c_obj };
    c_obj.print();
    B(c_obj).print();
    for (int i=0; i<2; i++)
        a_arr[i]->print();
    ((B*)(a_arr[1]))->print();
    dynamic_cast<A*>(a_arr[1])->print();
    return 0;
}
```

Ausgabe des Programms:

Sommersemester 2007	Blatt Nr:	13 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 5: Klassendefinition und STL

(ca. 20 Min.)

Die Klassen **Punkt**, **Linie** und **Linienzug** sind wie folgt vorgegeben

```

class Punkt {
    double x,y;
public:
    Punkt(double x, double y);
    double getX() const;
    double getY() const;
    bool operator!=(const Punkt& p);
};

class Linie {
    Punkt anfang, ende;
public:
    Linie(Punkt a, Punkt e);
    Punkt getEnde() const;
    double getLaenge() const;
};

class Linienzug {
    list<Linie> linien;
public:
    Linienzug();
    void linieAnfuegen(Punkt& a, Punkt& e);
    double getGesamtLaenge() const;
};

```

Programmieren Sie bitte nun im Folgenden außerhalb der Klasse **Linienzug** die auf der folgenden Seite angegebenen Methoden aus.

Sommersemester 2007	Blatt Nr:	14 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Lösungsblatt für die Aufgabe 5

5.1 Die Methode `linieAnfuegen()` fügt einem Objekt der Klasse `Linienzug` ein Objekt der Klasse `Linie` an. Als Parameter werden der Anfangs- und der Endpunkt der Linie benötigt. Die Methode prüft, ob der Endpunkt der zuletzt eingetragenen Linie mit dem als Parameter gegebenen Anfangspunkt übereinstimmt. Falls nicht, wird die neue Linie nicht dem Linienzug angefügt, sondern eine Ausnahme der Klasse `string` geworfen. Als Information ist der String "Endpunkt ungleich Anfangspunkt" dem `string`-Objekt mitzugeben.

5.2 Die Methode `getGesamtLaenge()` berechnet den Gesamtlänge des Linienzugs mithilfe der Methode `getLaenge()` der Klasse `Linie`.

Sommersemester 2007	Blatt Nr:	15 / 15
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aufgabe 6: Exception-Handler

(ca. 8 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 `linieAnfuegen()` 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 `string`-Objekt gespeicherten String aus.

```
int main (void) {  
  
    Linienzug lzug;  
  
    lzug.linieAnfuegen(Punkt(0,0), Punkt(3,0));  
    lzug.linieAnfuegen(Punkt(3,0), Punkt(3,4));  
    cout << lzug.getGesamtLaenge() << endl;  
  
    return 0;  
};
```