

Wintersemester 2008/2009	Zahl der Blätter: 14 Blatt Nr: 1
Fachbereich: Informationstechnik	Semester: IT3A / IT3B
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
<p>1) Gegeben seien folgende Variable/Referenz</p> <pre>int i=0; int &amp;j = i;</pre> <p>Referenz und Variable belegen jeweils vier Bytes im Speicher.</p> <p><b>Begründung:</b></p>		
<p>2) Für jede Klasse muss der Programmierer einen Konstruktor definieren.</p> <p><b>Begründung:</b></p>		

Wintersemester 2008/2009	Blatt Nr: 2 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
<p>3) Gegeben seien Klassendeklarationen</p> <pre>class A {protected: int x;}; class B : private A { ... };</pre> <p>In B kann auf die Instanzvariable A::x zugegriffen werden.</p> <p><b>Begründung:</b></p>		
<p>4) Gegeben sei</p> <pre>class A {     static int x;     A():x(1){} };</pre> <p>Der Compiler findet einen Fehler.</p> <p><b>Begründung:</b></p>		
<p>5) Gegeben sei:</p> <pre>Person p1("Anton"); Person p2 = p1;</pre> <p>Dazu wird der Zuweisungsoperator verwendet.</p> <p><b>Begründung:</b></p>		

Wintersemester 2008/2009	Blatt Nr:	3 / 14
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

Aussage	wahr	falsch
<p>6) Die Methode <code>at()</code> der Klasse <code>string</code> kann eine Ausnahme auslösen.</p> <p><b>Begründung:</b></p>		
<p>7) Virtuelle Konstruktoren können in abgeleiteten Klassen überladen werden.</p> <p><b>Begründung:</b></p>		
<p>8) Alle Initialisierungen in der Initialisierungsliste eines Konstruktors könnte man auch im Rumpf des Konstruktors durch Zuweisungen realisieren.</p> <p><b>Begründung:</b></p>		
<p>9) Alle Ausnahmen sind Objekte der Oberklasse <code>exception</code>.</p> <p><b>Begründung:</b></p>		

Wintersemester 2008/2009	Blatt Nr: 4 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Aussage	wahr	falsch
<p>10) Eine statische Methode der Klasse A kann nur auf alle Klassenvariablen von A zugreifen.</p> <p><b>Begründung:</b></p>		

### Aufgabe 2: Klassendefinitionen

(ca. 13 min)

Sie sollen eine Klasse **Produkt** programmieren. Ein Produkt wird durch eine Maschine aus verschiedenen Teilen zusammengesetzt. Zum Beispiel könnte die Maschine aus den Teilen Unterschale (mit Elektronik), Oberschale, Display und Tastatur ein Produkt Handy zusammensetzen.

Die Klasse **Produkt** ist von der Klasse **Teil** abgeleitet. Die Deklaration der Klasse **Teil** ist folgendermaßen vorgegeben:

```
// Teile sind von einem bestimmten Typ (z.B. Tastatur).
// Dieser Typ wird in einem konstanten String gespeichert.
class Teil
{
protected:
    const string typ;
public:
    Teil(string);
    string getTyp();
};
```

Die Klasse **Produkt** soll folgende Variablen und Eigenschaften besitzen:

- die Klasse **Produkt** ist von der Klasse **Teil** abgeleitet, da ein Produkt wieder ein Teil für ein weiteres Produkt sein kann
- eine Klassenvariable **anz** vom Typ **int**, die zählt, wie viele Produkte bereits hergestellt wurden
- eine konstante Instanzvariablen **nr** vom Datentyp **int**, die die Seriennummer eines Produktes angibt; die Seriennummer soll mit Hilfe der Klassenvariablen **anz** gesetzt werden

Wintersemester 2008/2009	Blatt Nr: 5 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

- d) ein Array von 10 Pointern auf `Teil`; das Array speichert, aus welchen Teilen das Produkt zusammengesetzt ist; wenn das Produkt aus weniger als 10 Teilen besteht, so haben die ungenutzten Pointer den Wert NULL
- e) einen Konstruktor mit Parameter vom Typ `string`; der String soll genutzt werden, um die Instanzvariable `typ` zu initialisieren; der Konstruktor verändert die Klassenvariable `anz` entsprechend
- f) einen Destruktor, der auch alle Teile des Produkts zerstört
- g) eine Instanzmethode `getNr()`, die die Seriennummer eines Produkts zurückliefert
- h) eine Klassenmethode `getAnz()`, die den Wert von `anz` zurückliefert
- i) einen Operator `+`, der als linken Operanden ein Produkt und als rechten Operanden einen Pointer auf ein Teil nimmt und eine Referenz auf das Produkt wieder zurückliefert. Mit dem Operator `+` wird ein Teil zu einem noch nicht fertig gestellten Produkt hinzugefügt (wenn möglich).
- j) einen Ausgabeoperator `<<`, der den Typ des Produkts und die Typen aller Teile, aus denen ein Produkt besteht, am Bildschirm ausgibt

Ergänzen Sie das folgende Programmgerippe. Schützen Sie die Datenelemente vor Zugriffen durch klassenfremde Methoden.

### Prototyp der Klasse Produkt (Aufgabe 2)

```
#include <teil.hpp>
#include

class Produkt
{
    // Instanz- und Klassenvariablen
```

Wintersemester 2008/2009	Blatt Nr: 6 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

```
// Prototyp des Konstruktors
```

```
// Prototyp des Destruktors
```

```
// Prototyp der Instanzmethode getNr()
```

```
// Prototyp der Methode getAnz()
```

```
// Prototyp von Operator +
```

```
// Prototyp des Ausgabeoperators <<
```

```
};
```

### **Aufgabe 3: Implementierung der Methoden und Operatoren der Klasse** (ca. 25 Min)

Programmieren Sie hier und auf den folgenden Seiten außerhalb der Klasse `Produkt` die angegebenen Elemente aus.

```
// Konstruktor
```

Wintersemester 2008/2009	Blatt Nr: 7 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

// Destruktor

// Instanzmethode getNr()

// Klassenmethode getAnz()

// Operator +

// Ausgabeoperator fuer ein Produkt

Wintersemester 2008/2009	Blatt Nr: 8 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

Vervollständigen Sie das folgende Hauptprogramm.

```
//Initialisierung der Klassenvariablen
```

```
int main()
{
    // Anzahl von existierenden Produkten ausgeben
    cout << "Anzahl Produkte: " <<

    // Produkt vom Typ Handy konstruieren

    // Eine Tastatur und ein Display zum Handy hinzufügen

    // Das Handy am Bildschirm ausgeben
}
```

Wintersemester 2008/2009	Blatt Nr: 9 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

#### Aufgabe 4: Polymorphie und RTTI

(ca. 10 min)

In einer Fabrik werden Produkte hergestellt, die ein Gehäuse aus zwei Teilen besitzen.

Gegeben sind dazu:

```
class Teil { virtual print { cout << "PRINT" << endl; };
class Plastik: public Teil {};
class Gehaeuse: public Teil {};
class Produkt: public Teil {};
```

- Erstellen Sie auf der nachfolgenden Seite eine abstrakte Basisklasse **Maschine**. Die Klasse **Maschine** enthält nur eine konstante Instanzmethode **arbeiten**, die einen Parameter vom Typ Zeiger auf **Teil** übernimmt und den gleichen Typ zurückgibt.
- Leiten Sie von der Basisklasse **Maschine** eine konkrete Unterklasse **GussMaschine** ab: Die **GussMaschine** verarbeitet **Plastik** zu **Gehaeuse**.
- Überladen Sie dazu die Instanzmethode **arbeiten** der Basisklasse **Maschine** für die **GussMaschine**. Die Verarbeitung eines **Teils** soll dadurch simuliert werden, dass das übergebene Teil vom Typ **Plastik** gelöscht wird und ein neues **Teil** vom Typ **Gehaeuse** zurückgegeben wird.
- Prüfen Sie in der Instanzmethode **arbeiten** mit Hilfe von RTTI, ob das zu verarbeitende **Teil** den korrekten Typ **Plastik** hat. Falls dies nicht der Fall ist, oder falls der übergebene Zeiger ungültig ist (=0) soll eine Fehlermeldung als Ausnahme geworfen werden.

Wintersemester 2008/2009	Blatt Nr: 10 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

// **Abstrakte Basisklasse Maschine**

// **Abgeleitete Klasse Gussmaschine**

Wintersemester 2008/2009	Blatt Nr:	11 / 14
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

### Aufgabe 5: Ausnahmebehandlung

(ca. 10 Minuten)

Eine Maschinensimulation soll eine Kiste mit Resten verwerten, die **Plastik, Gehaeuse** sowie unbrauchbare Teile enthält. Die Verarbeitung soll so erweitert werden, dass alle drei Arten von Teilen an die Maschinen gegeben werden können.

Gegeben seien dazu:

```
class Teil;
class Plastik;
class Gehaeuse;
class Produkt;
class GussMaschine;
class MontageMaschine;
```

Beide Maschinen enthalten eine Methode **arbeiten**, die im Falle von nicht passenden Teilen eine Ausnahme vom Typ **MaschinenFehler** wirft.

- Erstellen Sie eine Klasse **MaschinenFehler**, die als private Instanzvariablen einen String für die Beschreibung des Fehlers sowie einen Zeiger auf ein Objekt der Klasse **Teil** enthält.
- Weiterhin soll die Klasse einen sinnvollen Konstruktor sowie eine Instanzmethode **print** zur Ausgabe der Fehler enthalten. Geben Sie bei der Methode **print** eine gesonderte Meldung aus, falls der Zeiger auf **Teil** ungültig (= 0) ist.
- Ergänzen Sie dann die nachfolgende Funktion **verwerteReste** um die Fehlerbehandlung.

```
// Klasse MaschinenFehler
```

Wintersemester 2008/2009	Blatt Nr:	12 / 14
Prüfungsfach: Informatik 3	Fachnummer:	KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:	

```

void verwerteReste() {
    stack<Teil*> produkte;
    Maschine* m = new MontageMaschine();
    Maschine* g = new GussMaschine();
    srand( time(0) );           // Krabbeltisch durchmischen
    while(produkte.size() < 10) {
        int zufall = rand() % 3;
        Teil* r;
        switch (zufall) {
            case 0: r = 0; break;           // nicht verwertbar
            case 1: r = new Plastik; break;
            case 2: r = new Gehaeuse; break;
            default:
                cout << "Merkwürdiger Zufall " << zufall << endl;
        }
        //////////////////////////////////// HIER FEHLERBEHANDLUNG EINFUEGEN

        r = g->arbeiten( r );

        r = m->arbeiten( r );

        if (r)
            produkte.push(r);

    } // end of while

    delete m;
    delete g;
} // end of verwerteRest

```

Wintersemester 2008/2009	Blatt Nr: 13 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

### Aufgabe 6: STL

(ca. 10 min)

Auf verschiedenen Fließbändern werden Teile für eine Maschine angeliefert, die ein Produkt aus Teilen zusammensetzt.

- Ein Fließband kann nur eine Art von Teil (typ) transportieren.
- Auf das Fließband passen **anz** viele Teile. Die Variable **anz** wird beim Konstruktor als Parameter übergeben aber nicht in einer Instanzvariablen gespeichert.
- Zu beachten ist, dass nur am Anfang des Bandes Teile auf das Band gelegt und nur am Ende Teile heruntergenommen werden können, die durch die Taktung auch schon am Ende des Bandes angekommen sind. Es handelt sich hier also um einen FIFO-Speicher mit fester Größe.

Der Prototyp der Klasse **Fliessband** ist folgendermaßen gegeben. Die Klasse **Teil** ist in Aufgabe 2 gegeben.

```
class Fliessband
{
    string typ;
    vector<Teil*> f;
public:
    // Konstruktor
    Fliessband(string="NA", int anz=10);

    // Fließband takten, d.h. das Fließband bewegt sich
    // um eine Position weiter. Dies geht nur, wenn das
    // Ende des Bandes leer ist. In diesem Fall wird true
    // zurückgegeben, andernfalls false.
    bool takt();

    // Anzahl Teile auf dem Fließband zurückgeben
    // Zu beachten ist, dass das Fließband nicht voll
    // belegt sein muss. Es kann Lücken geben.
    int anzTeile();
};
```

Programmieren Sie die Methoden aus.

```
// Konstruktor
Fliessband::Fliessband(string s, int anz)
```

Wintersemester 2008/2009	Blatt Nr: 14 / 14
Prüfungsfach: Informatik 3	Fachnummer: KTB/SWB/TIB 3011
Name:	Matrikel-Nr.:

```
// Fließband takten  
bool Fließband::takt()
```

```
// Instanzmethode anzTeile  
int Fließband::anzTeile()
```