Klausuren

Prüfungsergebnisse

Allgemeine Informationen

Modus:	Testklausur
Klausurname:	Übungstest Nr. 3
Prüfungsteilnehmer:in:	Henri Schulz
Datum:	28. Aug. 2025
Bearbeitungszeit:	100d
Genutzte Arbeitszeit: :	36min 8s (60.22 %)

Aufgaben

#1 Aufgabe 1.1 [8 Punkte]

1) Welche Aussagen treffen zu? Unterscheiden Sie dabei zwischen Kompilierungsfehlern und Konvention bzw. Sinnhaftigkeit. Gehen Sie davon aus, dass die Methoden außerhalb der Klassen implementiert werden.

```
#include <iostream>

class Cipher {
public:
    encode( const std::string& );
    decode( const std::string& );
private:
    long secretKey;
```


Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

- Kompilierfehler - Die Klasse benötigt einen expliziten Konstruktor
- Konvention - Die Klasse benötigt einen expliziten Konstruktor
- Kompilierfehler - Die Parameter in den Methoden dürfen nicht const sein
- Konvention - Die Parameter in den Methoden dürfen nicht const sein
- Kompilierfehler - Die Methoden `encode()` und `decode()` müssen einen Return-Typ aufweisen
- Konvention - Die Methoden `encode()` und `decode()` müssen einen Return-Typ aufweisen
- Es gibt zusätzliche Kompilierungsfehler
- Es gibt zusätzliche Verstöße gegen Konventionen bzw. Sinnhaftigkeit

#2 Aufgabe 1.2 [6 Punkte]

1) Welche Aussagen treffen zu? Unterscheiden Sie dabei zwischen Kompilierungsfehlern und Konvention bzw. Sinnhaftigkeit. Gehen Sie davon aus, dass die Methoden außerhalb der Klassen implementiert werden.

```
#include <iostream>

class Box {
public:
    Box() = default;
    explicit Box(int i) : m_width(i), m_length(i), m_height(i){}
    Box(int width, int length, int height): m_width(width), m_length(length), m_height(height){}

    int Volume() { return m_width * m_length * m_height; }
private:
    int m_width{ 0 };
    int m_length{ 0 };
    int m_height{ 0 };
};
```

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

- [ ] Konvention - Die Methoden Volume() muss const sein
- [ ] Kompilierfehler - Die Methoden Volume() muss const sein
- [ ] Kompilierfehler - explicit darf nicht vor dem Konstruktor stehen
- [ ] Es gibt keine weiteren Kompilierungsfehler
- [ ] Es gibt keine weiteren Verstöße gegen Konventionen bzw. Sinnhaftigkeit
- [ ] Kompilierfehler - explicit darf nicht vor dem Konstruktor stehen

#3 Aufgabe 1.3 [11 Punkte]

1) Welche Aussagen treffen zu? Unterscheiden Sie dabei zwischen Kompilierungsfehlern und Konvention bzw. Sinnhaftigkeit. Gehen Sie davon aus, dass die Methoden außerhalb der Klassen implementiert werden.

```
class Fraction {
    Fraction(long numerator, long denominator) {
        this->numerator = numerator;
        this->denominator = denominator;
    }
    long numerator, denominator;
private:
    void convert( double );
    long gcd( void );
}
```

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

- [ ] Alles was nicht explizit in einem Bereich liegt, wird implizit public
- [ ] Alles was nicht explizit in einem Bereich liegt, wird implizit private
- [ ] Konvention - Es fehlt der public Bereich

Kompilierfehler - Parameter im Konstruktor müssen anders heißen als die member

Kompilierfehler - Die Methode gcd darf kein void als Parameter enthalten

Kompilierfehler - Es fehlt der public Bereich

Alles was nicht explizit in einem Bereich liegt, wird implizit protected

Konvention - Parameter im Konstruktor müssen anders heißen als die member

Konventionen - Die Methode gcd darf kein void als Parameter enthalten

Es gibt zusätzliche Verstöße gegen Konventionen bzw. Sinnhaftigkeit

Es gibt zusätzliche Kompilierungsfehler

#4 Aufgabe 1.4 [4 Punkte]

1) Welche Aussagen treffen zu? Unterscheiden Sie dabei zwischen Kompilierungsfehlern und Konvention bzw. Sinnhaftigkeit. Gehen Sie davon aus, dass die Methoden außerhalb der Klassen implementiert werden.

```
#include <iostream>

class C {
    std::string test;
public:
    ~C();
    ~C( std::string& );
};
```

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

Konvention - Der Dekonstruktor kann nicht überladen werden

Kompilierfehler - Der Dekonstruktor kann nicht überladen werden

Es gibt zusätzliche Kompilierungsfehler

Es gibt zusätzliche Verstöße gegen Konventionen bzw. Sinnhaftigkeit

#5 Aufgabe 2.1 [1 Punkte]

1) Angenommen die unten dargestellten Klassen sind in der Header-Datei myClasses.h definiert. Für ein Objekt obj der Klasse Add ist dann folgende Anweisung zulässig?

```
int res = obj.calc( 2 );
class Polynom {
    int x;
public:
    Polynom();
    virtual int calc( int a );
};

class Add : public Polynom {
    int y;
public:
    Add();
    int calc();
};
```

Wähle bitte die richtige Antwortmöglichkeit aus.

Antwort

- Nein
- Ja

#6 Aufgabe 2.2 [1 Punkte]

1) Geben Sie die Ausgabe des Programmes an.

#include

class B {

public:

B() { std::cout << "Konstruktor der Klasse B \n"; }

~B() { std::cout << "Destruktor der Klasse B \n"; }

};

class D : public B {

public:

D() { std::cout << "Konstruktor der Klasse D \n"; }

~D() { std::cout << "Destruktor der Klasse D \n"; }

};

class X {

public:

X() { std::cout << "Konstruktor der Klasse X \n"; }

~X() { std::cout << "Destruktor der Klasse X \n"; }

```
private:

D d;

};


int main() {

X xObj;

std::cout << "Bye, bye!" << std::endl;

return 0;

}
```

1 Konstruktor der Klasse B

2 Konstruktor der Klasse D

3 Konstruktor der Klasse X

4 Bye, bye!

5 Destruktor der Klasse X

6 Destruktor der Klasse D

7 Destruktor der Klasse B

#7 Aufgabe 2.3 [4 Punkte]

1) Geben Sie die jeweilige Ausgabe an.

Gegeben ist der in dargestellte Programmcode einer Main-Funktion sowie die Definition mehrere Klassen:

```
#include

class Plant{

public:

Plant(){}

virtual ~Plant(){}

void molder();

virtual int bringSeeds();

};
```

```
class Flower : public Plant{
public:
void molder();
int bringSeeds();
};
class Tree : public Plant{
public:
int bringSeeds();
};

void Plant::molder() {
std::cout << "oh ich zerfalle zu Humus" << std::endl;
}
int Plant::bringSeeds() {
std::cout << "Ich bin eine grundlegende Pflanze." << std::endl;
return 1;
}
int Flower::bringSeeds() {
std::cout << "Ich habe geblueht. Viele Samen." << std::endl;
return 400;
}
void Flower::molder() {
std::cout << "oh nein, meine Schoenheit ist dahin!" << std::endl;
}
int Tree::bringSeeds() {
std::cout << "ich habe Samen geworfen." << std::endl;
return 5000;
}

int main() {
auto* simplePlant = new Plant();
auto* myTreeBernd = new Tree();
auto* daffodil = new Flower();
Plant* anyPlant;
```

```
//Anweisungsblock A

return 0;


new


simplePlant->molder();

myTreeBernd->molder();

daffodil->molder();

anyPlant->molder();

}
```

anyPlant = myTreeBernd; anyPlant->molder();

Ausgabe: oh ich zerfalle zu Humus

anyPlant = daffodil; ((Plant*)anyPlant)->molder();

Ausgabe: oh ich zerfalle zu Humus

myTreeBernd->bringSeeds();

Ausgabe: ich habe Samen geworfen.

simplePlant->bringSeeds();

Ausgabe: Ich bin eine grundlegende Pflanze.

#8 Aufgabe 2.4 [6 Punkte]

1) Betrachten Sie die zwei unterschiedlichen Implementierungen der Klassen Base und Derive. Was wird in der jeweiligen Main-Funktion ausgegeben?

```
#include

//parent class

class Base {

public:

void result(){

std::cout << "Result method of Base class is called" << std::endl;
```

```
}
};

// Following is the Derived class having similar
//result() method as of base class
class Derive : public Base{
public:
//definition of a result method already exists in Base class
void result(){
std::cout << "The result method of derived class is called" << std::endl;
}
};

int main() {
//instantiating object of Base class
Base obj1 = Base();
//object of child class
Base obj2 = Derive();
//object of child class
Derive obj3 = Derive();

obj1.result();
obj2.result();
obj3.result();
}
#include
//Parent class
class Base {
public:
virtual void result(){
std::cout << "Result method of Base class is called" << std::endl;
}
};

// Following is the Derived class having similar result() method as of base class
```

```
class Derive : public Base {

public:

//definition of a result method already exists in Base class

void result(){

std::cout << "The result method of derived class is called" << std::endl;

}

};


int main() {

//instantiating object of Base class

Base * obj1 = new Base();

//object of child class

Base * obj2 = new Derive();

//object of child class

Derive * obj3 = new Derive();


obj1->result();

obj2->result();

obj3->result();

}
```

Obere Implementierung Ausgabenzeile 1: Result method of Base class is called

Obere Implementierung Ausgabenzeile 2: Result method of Base class is called

Obere Implementierung Ausgabenzeile 3: The result method of derived class is called

Untere Implementierung Ausgabenzeile 1: Result method of Base class is called

Untere Implementierung Ausgabenzeile 2: The result method of derived class is called

Untere Implementierung Ausgabenzeile 3: The result method of derived class is called

#9 Aufgabe 3.2 [15 Punkte]

Du hast keine Lösung für diese Aufgabe abgegeben.

1) Kreuzen Sie die richtigen Antworten an.

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort
Im System können mehr C als B enthalten sein.
in Objekt von D steht in Beziehung zu mindestens einer direkten Instanz von E.
Zwei Objekte von G stehen mit drei Objekten von F in Beziehung.
Ein Objekt von D ist in genau einem Objekt von E enthalten.
Ein Objekt von H steht in Beziehung zu mindestens einem Objekt von G und die Beziehung kann von H aus navigiert werden.
Ein Objekt von E kann in Beziehung zu mehreren Objekten von D stehen.
Die Raute bei G wird als Komposition bezeichnet.
Ein Objekt von B muss in Beziehung zu einem Objekt von H stehen.
Ein Objekt von F kann mit sich selbst in Beziehung stehen.
Wenn eine Instanz von F gelöscht wird, werden alle enthaltenen Instanzen von D gelöscht.
Ein Objekt von H muss in Beziehung zu einem Objekt von B stehen.
Ein Objekt von B steht in Beziehung zu entweder genau einem Objekt von H oder genau einem Objekt von D.
Ein Objekt von F kann direkt auf die Variable z zugreifen.
Ein Objekt von H steht in Beziehung zu mindestens einem Objekt von B.
Eine Instanz von B kann auf die Variable x zugreifen.

#10 Aufgabe 3.3 [10 Punkte]

Du hast keine Lösung für diese Aufgabe abgegeben.

Problemstellung

Stellen Sie sich vor, Sie gründen ein Startup, das mit einer neuen Software den Betrieb der Mensa verbessern möchte. Dazu müssen Sie zuerst modellieren, v

Überlegen Sie sich, welche Objekte in einer Mensa miteinander interagieren. Welche Klassen, Attribute und Methoden könnten geeignet sein, um diese Objek sich für drei bis vier Klassen und erstellen sie für diese ein Klassendiagramm.

#11 Aufgabe 4.1 [1 Punkte]

1) Welche Auswahl beinhält keinen Fehler?

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

```
std::shared_ptr<int> p1(new int(42));
std::shared_ptr<int> p2(p1);
```

```
std::shared_ptr<int> p1(new int(42));
std::shared_ptr<int> p2(p1.get());
```

Der Code kompiliert nicht (mit entsprechender Programmstruktur drum herum)

#12 Aufgabe 4.2 [1 Punkte]

1) Welche Aussage trifft auf folgendem Programmausschnitt zu?

```
std::shared_ptr<int> p_shared_int(new int(6));
std::unique_ptr<int> p_unique_int(std::move(p_shared_int));
```

Wähle bitte die richtige Antwortmöglichkeit aus.

Antwort

Komilierfehler

Kompiliert

#13 Aufgabe 4.3 [3 Punkte]

1) Welche Fehler treten in dem folgendem Code auf? Beheben Sie diese unter anderem mit der Nutzung von std::unique_ptr.

```
#include <iostream>
#include <functional>

class Point {
public:
    Point(const double x, const double y) : m_x(x), m_y(y){}
    double m_x;
    double m_y;
};

std::function<int(int,int)> sum;

int main() {
    try {
        auto pt = new Point(1.0, 2.0);

        (*pt).m_x = 3.0;
        (*pt).m_y = 3.0;

        std::cout << sum(pt->m_x, pt->m_y) << std::endl;

        delete pt;
    } catch (std::exception& e) {
        std::cout << e.what() << std::endl;
    }
}
```

Wähle bitte alle richtigen Antwortmöglichkeiten aus.

Antwort

- Kompiliert nicht
- Narrowing conversion
- Memory Leak