# Prüfung **Programmentwicklung 2**

| Nachname | Vorname |
|---|---|
| Matrikelnummer | Studiengang |

**Hinweise:**

- Bearbeitungszeit: 120 Minuten.
- Die Bearbeitung darf nur mit dokumentenechten Stiften erfolgen. Verwenden Sie keinen Bleistift. Verwenden Sie keinen Stift, der in Rot schreibt.
- Bei unterschiedlichen abgegebenen Lösungen für eine Aufgabe, wird die Aufgabe nicht bewertet. Antworten, die der Korrektor nicht lesen kann, werden nicht bewertet.
- Verwenden Sie keine Korrekturflüssigkeiten (*TippEx*) oder Tintenkiller. Nicht zu bewertende Lösungsteile sind durchzustreichen.
- Erlaubte Hilfsmittel: ein nicht-programmierbarer Taschenrechner sowie alle weiteren nicht-technischen Hilfsmittel.
- Mobiltelefone, Smartwatches und ähnliche Geräte sind ausgeschaltet in der Tasche zu verwahren. Die Verwendung eines solchen Gerätes wird als Täuschungsversuch gewertet.

Durch Unterschrift bestätigen Sie, dass Sie die Hinweise zur Bearbeitung der Klausur gelesen und verstanden haben, und sie als Bestandteile der Prüfungsbedingungen anerkennen.

Unterschrift

| Aufgabe: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Σ | **Note** |
|---|---|---|---|---|---|---|---|---|---|---|
| Punkte: | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 160 | |
| Erreicht: | | | | | | | | | | |

1. Prüfer: Dr. Michael Gref

2. Prüfer*in

# 1 Aufgabe (Fehlersuche)

Die nachfolgenden Teilaufgaben enthalten Codeausschnitte, die jeweils einen oder mehrere Fehler beinhalten. Finden und korrigieren jeweils Sie den oder die Fehler direkt im jeweiligen Codeausschnitt oder auf den Rückseiten unter Angabe der entsprechenden Zeilennummern.

## 1.1 Teilaufgabe [10]

Gegeben sei das folgende `C++`-Programm, das in einem Videospiel den Gesundheitsstatus (health) von Charakteren (Helden/Hero und Monstern) berechnet:

```
1  #include<iostream>
2
3  class Character {
4    private:
5    int health;
6    Character(int h) : health(h) {}
7    int getHealth() { return health; }
8    void setHealth(int h) { health = h; }
9  };
10
11 class Hero : public Character {
12   public:
13   Hero(int h) : Character(h) {}
14   void attack(Character m) { m.setHealth(m.getHealth() - 50); }
15 };
16
17 class Monster : public Character {
18   public:
19   Monster(int h) : Character(h) {}
20 };
21
22 int main() {
23   Hero hero(100);
24   Monster monster(100);
25   hero.attack(monster);
26   std::cout << "Monster Health: " << monster.getHealth() << std::
       endl;
27   return 0;
28 }
```

Die erwartete Ausgabe ist `Monster Health: 50`. Finden und korrigieren Sie den oder die Fehler unter Angabe der entsprechenden Zeilennummern.

### 1.2 Teilaufgabe [10]

Gegeben seien die folgenden Klassen für Musikinstrumente:

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

class Musikinstrument {
  public:
  string name;

  Musikinstrument(string n) : name(n) {}

  void spiel() {
    cout << name << " spielt: " << "Unbekannter Ton" << endl;
  }
};
class Gitarre {
  public:
  Gitarre() : Musikinstrument("Gitarre") {}

  void spiel() {
    cout << name << " spielt: " << "Gitarrensound" << endl;
  }
};
class Schlagzeug {
  public:
  Schlagzeug() : Musikinstrument("Schlagzeug") {}

  void spiel() {
    cout << name << " spielt: " << "Schlagzeugsound" << endl;
  }
};
```

Die gegeben Klassen sollen zu folgendem Programmabschnitt (der selbst keinen Fehler beinhaltet) kompatibel sein, enthalten jedoch Fehler:

```
Musikinstrument* instrument1 = new Gitarre();
Musikinstrument* instrument2 = new Schlagzeug();
instrument1->spiel();
instrument2->spiel();
```

Hierfür ist die erwartete Ausgabe:

```
Gitarre spielt: Gitarrensound
Schlagzeug spielt: Schlagzeugsound
```

# 2 Aufgabe (Programmverständnis)

## 2.1 Teilaufgabe [10]

Gegeben sei das folgende C++-Programm:

```
#include<iostream>
class Test {
  public:
  Test() {
    std::cout << "A ";
  }

  Test(const Test& t) {
    std::cout << "B ";
  }

  ~Test() {
    std::cout << "C ";
  }
};

Test fun(Test t) {Test x = t;}

int main() {
  Test t1;
  fun(t1);
  Test t2 = t1;
  return 0;
}
```

Wie ist die Ausgabe in der Konsole?

### 2.2 Teilaufgabe [10]

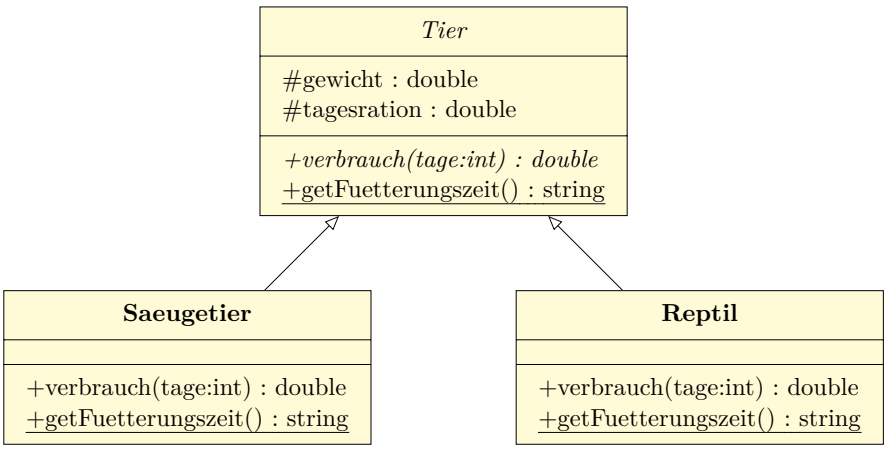
Gegeben sei das folgende C++-Programm:

```
1  #include<iostream>
2
3  template<typename T>
4  T add(T a, T b) {
5    return a + b;
6  }
7
8  int main() {
9    std::cout << add<int>(3, 4) << "\n";
10   std::cout << add<double>(3, 4.5) << "\n";
11   std::cout << add<std::string>("Hello, ", "World!") << "\n";
12   return 0;
13 }
```

Ist das gegebene Programm funktional oder beinhaltet es Fehler? Wie ist die Ausgabe des Programms für alle Zeilen, die keinen Fehler beinhalten?

## 3 Aufgabe (Polymorphie) [20]

Im folgenden Klassendiagramm ist ein Ausschnitt einer Zoo-Verwaltungssoftware gezeigt. Jedes Tier hat spezielle Fütterungsanforderungen, die den Verbrauch bestimmen.




Die Methode `verbrauch` und `getFuetterungszeit` der jeweiligen Klassen gestalten sich wie folgt:

- Säugetier: verbauch = gewicht $*$ tagesration $*$ tage
  Fütterungszeit ist `"Mittags"`
- Reptilen erhalten nur für alle vollen 7 Tage jeweils

  $$10 * \text{tagesration} * \text{gewicht}$$

  Bei weniger als 7 Tagen, erhalten Reptilien keine Essensration.
  Fütterungszeit ist `"Nachts"`

Stellen Sie mittels dynamischer Bindung sicher, dass die Methoden der zugehörigen Objekte verwendet werden, auch wenn auf Objekte der Unterklassen mittels Zeigern vom Typ der Oberklasse zugegriffen wird.

Geben Sie eine Implementierung in `C++` an, die die Klassen samt Methoden entsprechend der Anforderungen implementiert.

(Platz für Lösungen)

## 4 Aufgabe (Ein-/Ausgabe) [20]

Gegeben ist eine Textdatei zur Verwaltung des Warenbestands eines Supermarktes, in der pro Zeile für jeden Artikel nacheinander eine ID (max. 3 Ziffern), die vorhandene Stückzahl, der Preis und der Namen (max. 24 Stellen) gespeichert ist. Die einzelnen Werte sind mittels Komma voneinander getrennt. Beispiel:

```
21,121,1.2,Gurken
131,32,0.49,Haferflocken
101,49,0.19,Gemuesebruehe
56,0,11.99,Cola
```

Schreiben Sie ein **C++**-Funktion `void printSupermarktData(const string& filename)` , die den Dateipfad als Argument übergeben bekommt, die einzelnen Werte in Variablen geeigneten Typs einliest und abschließend für jeden Artikel den Wert des Gesamtwarenbestandes (Anzahl $*$ Preis) formatiert ausgibt. Für das obige Beispiel soll dies wie folgt aussehen:

```
ID: 021 (Gurken): 145.20 EUR
ID: 131 (Haferflocken): 15.68 EUR
ID: 101 (Gemuesebruehe): 9.31 EUR
ID: 056 (Cola): 0.00 EUR
```

Hinweise: Falls die Datei nicht geöffnet werden konnte, soll eine Hinweis in die Standardausgabe geschrieben werden. Verwenden Sie keine Funktionen wie scanf oder printf aus der C-Standardbibliothek. Geben Sie alle notwendigen Header an.

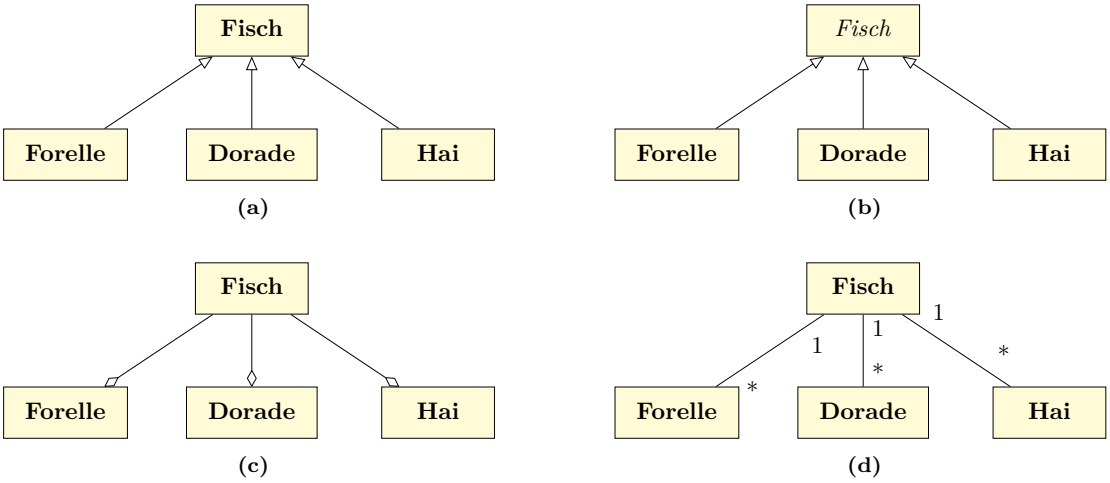
(Platz für Lösungen)

# 5 Aufgabe (Klassenbeziehungen)

In den folgenden Teilaufgaben, ist jeweils eine Aussage sowie verschiedene UML-Klassendiagramme gegeben. **Begründen** Sie kurz(!) für jedes Diagramm, ob eine Realisierung auf Basis dieses Diagramms geeignet ist, die Zusammenhänge der gegebene Aussage als Klassenbeziehung zu modellieren oder nicht.

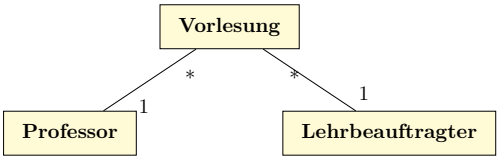
## 5.1 Teilaufgabe [10]

**Aussage**: Es gibt viele verschiedene Fischarten, unter anderem Forelle, Dorade und Hai. (Alle Fischarten, die nicht in eine dieser drei Kategorien fallen, wollen wir einfach als *Fisch* bezeichnen.)

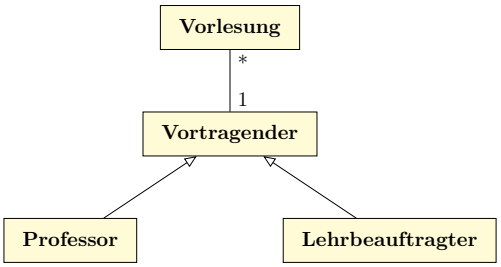


### 5.2 Teilaufgabe [10]

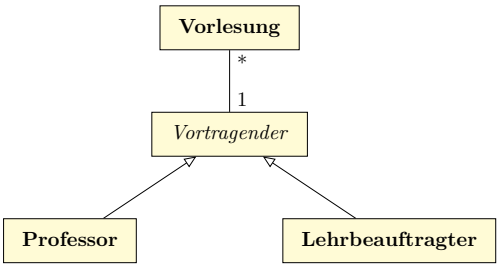
**Aussage**: Jede Vorlesung wird ausschließlich entweder von einem Professor oder einem Lehrbeauftragten gehalten.



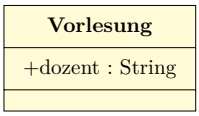

**(a)**




**(b)**




**(c)**




**(d)**

## 6 Aufgabe (Standard Template Library) [20]

Schreiben Sie eine `C++`-Funktion `void wordcount(string text)`, die für einen übergebenen Text die Häufigkeit aller auftretender Worte zählt und ausgibt, wie oft jedes Wort vorkommt. Hierbei soll die Ausgabe sortiert nach der Häufigkeit der vorkommenden Worte sein.

Sie können davon ausgehen, dass der Text bereits vorverarbeitet und normalisiert ist, d.h. dass keine Satzzeichen im Text vorkommen und alles kleingeschrieben ist. Beispiel:

```
1 string text = "der Fuchs springt ueber den Zaun der Hund springt
     auch der Fuchs ist schneller als der Hund armer Hund";
2 wordcount(text);
```

Erwartete Ausgabe (ersten 5 Zeilen):

```
1 der: 4
2 Hund: 3
3 springt: 2
4 Fuchs: 2
5 Zaun: 1
6 ...
```

Verwenden Sie die Container, Algorithmen und Funktionen der *Standard Template Library* (und `<algorithm>`), um die Aufgabe zu vereinfachen. Geben Sie alle notwendigen Header-Dateien an.

(Platz für Lösungen)

## 7 Aufgabe (Entwurfsmuster) [20]

Sie entwickeln die Logik für ein neues Strategiespiel eines Softwareunternehmens. In dem Spiel gibt es Figuren, die einzeln oder in Gruppen agieren können. Eine Gruppe kann hierbei aus einzelnen Figuren sowie wiederum aus Gruppen bestehen. Jede Einheit und jede Gruppe kann die folgenden Befehle ausführen:

- *Bewegen*: `void move()`
- *Angreifen*: `void attack()`
- *Verteidigen*: `void defend()`

Ihr Ziel ist es, unter Einsatz objektorientierter Programmierung eine flexible Softwarearchitektur zu entwerfen, die es auch in Zukunft erlaubt, neue Arten von Einheiten hinzuzufügen, ohne dass die bestehenden Schnittstellen geändert werden müssen. Insbesondere sollen es möglich sein, Einheiten und Gruppen dynamisch zu erstellen und zu organisieren.

1. Welches der aus der Vorlesung bekannten Entwurfsmuster würde diese Anforderungen am besten erfüllen und warum (**Begründung**)?
2. Skizzieren Sie ein UML-Klassendiagramm für die entsprechende Lösung, und erklären Sie kurz(!) die Rolle jeder Klasse in Ihrem Diagramm.
3. Wir gehen davon aus, dass die Methode `move(int x, int y)` für die Klasse, die die Bewegung einzelner Figuren realisieren, bereits implementiert ist. Hierbei ist `int x, int y` die Position auf der zweidimensionalen Karten, zu der sich die Figuren bewegen sollen.
   Schreiben Sie die Methode `move(int x, int y)` für die Klasse, die in Ihrer Lösung das Verhalten von Gruppen realisiert unter Verwendung der `move`-Methode der einzelnen Figuren.

(Platz für Lösungen)

## 8 Aufgabe (Verteilte Systeme) [20]

Gegeben sei die folgende C++-Implementierung eines Servers in einer Server-Client-Anwendung, die via Sockets Daten in Form eines Strings vom Client zum Server sendet.

```
#include <iostream>
#include <sys/socket.h>
#include <arpa/inet.h> // For inet_addr
#include <unistd.h>

#define MAX_BUFFER_SIZE 1024
#define PORT 8081

int main() {
  int sock, client_sock;
  struct sockaddr_in server, client;
  char buffer[MAX_BUFFER_SIZE];

  // Create socket
  sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock == -1) {
    std::cerr << "Could not create socket\n";
    return 1;
  }

  server.sin_family = AF_INET;
  server.sin_addr.s_addr = INADDR_ANY;
  server.sin_port = htons(PORT);

  // Bind
  if (bind(sock, (struct sockaddr*)&server, sizeof(server)) < 0) {
    std::cerr << "Bind failed\n";
    return 1;
  }

  // Listen
  listen(sock, 3);

  // Accept incoming connections
  std::cout << "Waiting for incoming connections...\n";
  socklen_t c = sizeof(struct sockaddr_in);
  client_sock = accept(sock, (struct sockaddr*)&client, &c);
  if (client_sock < 0) {
    std::cerr << "Accept failed\n";
    return 1;
  }

  std::cout << "Connection accepted\n";
```

```
44
45    // Receive a message from the client
46    while (true) {
47      ssize_t received = recv(client_sock, buffer, MAX_BUFFER_SIZE,
       0);
48      if (received <= 0) {
49        break;
50      }
51      std::cout << "Received message: " << buffer << "\n";
52    }
53
54    std::cout << "Closing connection...\n";
55    close(client_sock);
56    close(sock);
57
58    return 0;
59  }
```

1. Erweitern Sie die Anwendung in der Art, dass der Server eine Antwort mit dem Inhalt `"ACK"` an den Client zurücksendet.
   Hinweis: Sie müssen nicht den bestehend Quellcode abschreiben. Geben Sie Zeilennummern an, um kenntlich zu machen, wo Sie welche Änderungen vornehmen.
2. Erläutern Sie kurz, welche Änderungen auf Seiten des Clients notwendig sind, um die gewünschte Funktionalität zu realisieren.

(Platz für Lösungen)