Technische Hochschule Georg Agricola

**Allgemeine Angaben**

| | | |
|---|---|---|
| Modulprüfung | Klausur | Datum: 25.03.2021 |
| Modulname: **Objektorientierte Programmierung** | Modulnummer: | 40050200 |
| Prüfungsdauer: **60 min + 15 min** | Prüfer: Welp | |

Name: _____ Vorname:_____

Matrikelnummer: _____

**Bewertung**

| Aufgabe | 1 | 2 | 3 | 4 | 5 | | | | | Summe |
|---|---|---|---|---|---|---|---|---|---|---|
| Erreichbare Punkte | 11 | 12 | 11 | 9 | 14 | | | | | 57 |
| Erzielte Punkte | | | | | | | | | | |
| Unterschrift Prüfer | | | | Erzielte Punkte in % | | | | | Note | |

Technische Hochschule Georg Agricola

## Aufgabe 1: Von C nach C++

1. Es soll eine Funktion `polynom` erstellt werden, die lineare oder quadratische Polynome berechnet:
   - Lineares Polynom: $a_1*x + a_0$
   - Quadratisches Polynom: $a_2*x^2 + a_1*x + a_0$

   Der Funktion sollen als Parameter der `x`-Wert, die Koeffizienten $a_1$ und $a_0$ und optional $a_2$ übergeben werden.

   Lösen Sie dieses Problem, indem Sie
   1. `polynom` überladen
   2. `polynom` als Funktion mit Defaultparametern definieren.

   Geben Sie für beide Lösungswege die Funktions**definitionen** an. (5 Punkte)

2. Was gibt folgendes Programmfragment auf dem Bildschirm aus? (6 Punkte)

```
short n[2] = {100,100};
short &r = n[0];
short *p = &r;
r = r*2;
p++;
*p = n[0] + n[1];.
```

   Wie sieht nach Abarbeitung dieser Programmanweisungen das Speicherbild bei 32-Bit-Programmen aus? Tragen Sie in die folgende Abbildung ein, welche Speicherplätze von welchen Variablen belegt werden und wie der Inhalt der Speicherplätze aussieht.

| Name | Inhalt | Adresse |
|---|---|---|
| | | … |
| | | 2000 |
| | | 2002 |
| | | 2004 |
| | | 2006 |
| | | 2008 |
| | | 2010 |
| | | 2012 |
| | | 2014 |
| | | 2016 |
| | | 2018 |
| | | 2020 |
| | | 2022 |
| | | … |

.

## Aufgabe 2: Klassen und Objekte

Definieren Sie eine Klasse `Bruch` für die Darstellung von Brüchen. Die Klasse soll über folgenden Eigenschaften verfügen *(Anm.: Lassen Sie in der Klasse genügend Platz um nach und nach Ergänzungen vornehmen zu können)*:

- Der Zähler und der Nenner sollen als ganzzahlige Attribute in der Klasse gespeichert werden. Der direkte Zugriff auf den Zähler und den Nenner eines Bruches soll nicht möglich sein. Der Zugriff soll über entsprechende setter- und getter-Methoden erfolgen. **Definieren (implementieren)** Sie die Methoden als inline-Funktionen.
- Die Klasse soll über geeignete Kontruktoren zur Initialisierung von `Bruch`-Objekten verfügen. Ein `Bruch`-Objekt soll dabei sowohl über die Angabe von Zähler und Nenner als auch über eine ganze Zahl initialisiert werden können. Eine Objektinstanziierung ohne Parameter soll auch möglich sein. In diesem Fall soll das Objekt mit dem Wert 0 initialisiert werden (*siehe Hinweis*). **Definieren** Sie die Methoden als inline-Funktionen.
- Die Klasse soll eine Methode `getDezimalzahl()` zur Verfügung stellen, die den Bruch als Dezimalzahl zurückliefert. **Definieren** Sie die Funktion ausserhalb der Klasse.
- Die Klasse soll über ein Klassenattribut verfügen, in welchem die Anzahl der `Bruch`-Objekte gespeichert wird (Objektzähler). Die Anzahl der `Bruch`-Objekte soll über eine Klassenmethode abfragbar sein. **Definieren** Sie die Methode.
- Ferner soll der einstellige **~**-Operator für die Klasse `Bruch` überladen werden, sodaß der Bruch mit dem größten gemeinsamen Teiler von Zähler und Nenner gekürzt werden kann. **Deklarieren** Sie den Operator. *(siehe Hinweis)* (12 Punkte)

*Hinweis*

*Die Klasse `Bruch` soll z.B. folgendermaßen verwendet werden können*

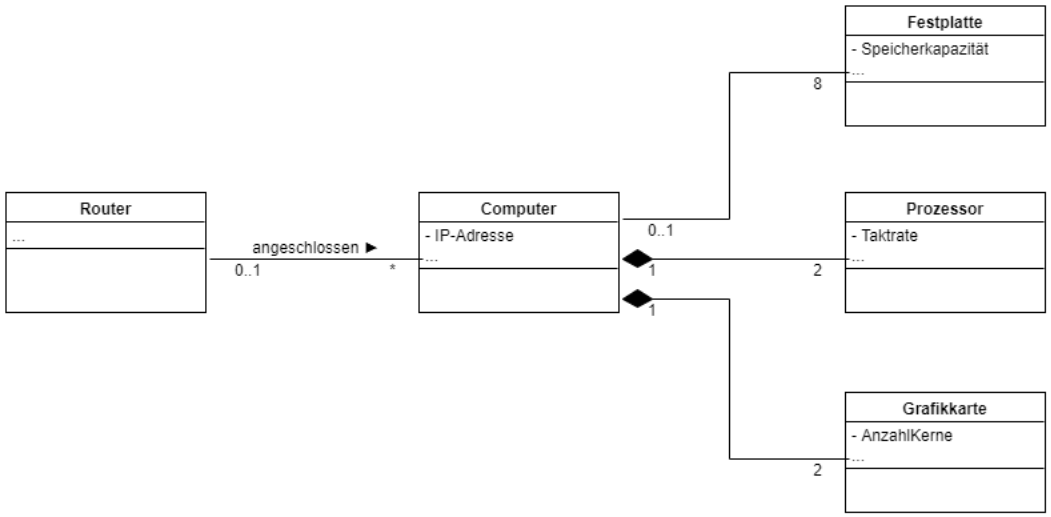
```
int main()
{
        Bruch b1(12,9);
        Bruch b2(5);
        Bruch b3;


        cout << "b1 = " << b1.getZaehler() << "/" << b1.getNenner()
             << " = " << b1.getDezimalzahl() << endl;
        cout << "b2 = " << b2.getZaehler() << "/" << b2.getNenner()
             << " = " << b2. getDezimalzahl () <<  endl;
        cout << "b3 = " << b3.getZaehler() << "/" << b3.getNenner()
             << " = " << b3. getDezimalzahl () <<  endl;

        b3 = ~b1;
        cout << "b1 = " << b1.getZaehler() << "/" << b1.getNenner()
             << " = " << b1. getDezimalzahl () << endl;
        cout << "b3 = " << b3.getZaehler() << "/" << b3.getNenner() << " = "
             << b3. getDezimalzahl () <<  endl;
        cout << "Es gibt insgesamt " << Bruch::getAnzObjekte() <<" Brueche" << endl;
}
```

*Ausgabe:*

```
b1 = 12/9 = 1.33333
b2 = 5/1 = 5
b3 = 0/1 = 0
b1 = 4/3 = 1.33333
b3 = 4/3 = 1.33333
Es gibt insgesamt 3 Brueche
```

## Aufgabe 3: Objektbeziehungen und Templates

In einer Netzwerkmanagement-Software gibt es die Klassen `Router`, `Computer`, `Festplatte`, `Prozessor` und `Grafikkarte`, die entsprechend dem folgendem Klassendiagramm miteinander in Beziehung stehen.

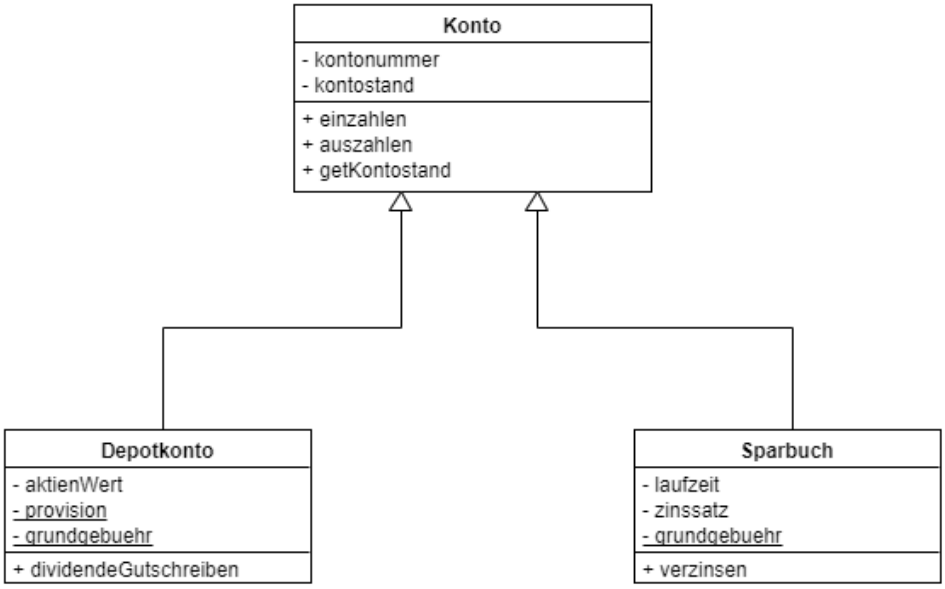



1. Welche Attribute werden in der Klasse `Computer` benötigt um die Beziehungen zu den Klassen `Router`, `Festplatte`, `Prozessor` und `Grafikkarte` zu implementieren? Beantworten Sie die Frage, in dem Sie die Klassendefinition für die Klasse Computer angeben (nur Attribute, Methoden sind nicht erforderlich). (4 Punkte)

2. Geben Sie die Definition des Attributs der Klasse `Router` an, mit dem die Beziehung zu Computerobjekten (*angeschlossen*) verwaltet werden kann. **Verwenden Sie hierzu eine Container-Klasse der Standard Template Library (STL).** (2 Punkte)
   Definieren Sie für die Klasse `Router` unter Verwendung des vorab definierten Attributs eine Methode, mit der gesetzt werden kann, dass der Router eine Verbindung zu einem Computer hat, und eine Methode, die alle mit dem Router verbundenen Computer zurückliefert. (5 Pkt.)

## Aufgabe 4: Vererbung

Gegeben sei folgendes Klassendiagramm:




Die Klasse `Konto` ist wie folgt definiert:

```
class Konto
{
    unsigned int kontonummer;
    float kontostand;
public:
    Konto(unsigned int knr, float ks=0.0.);
    float getKontostand();
    void einzahlen(float b);
    void auszahlen(float b);
};
```

1. Geben Sie Klassendefinition für die Klasse `Depotkonto` an. (3 Punkte)
2. Über welche Attribute verfügen **Objekte** der Klasse `Depotkonto`? (3 Punkte)
3. Über welche Methoden verfügen **Objekte** der Klasse `Depotkonto`? (2 Punkte)

## Aufgabe 5: Entwurfsmuster

Gegeben sei eine Klasse `Staedte`, die viele Informationen über Städte bereitstellt, und eine Klasse `Karte`, die Länder, Städte, Flüsse etc. als Karte darstellen kann.

1. In einem Programm wollen Sie die beiden Klassen `Staedte` und `Karte` über das Observer-Pattern verknüpfen. Welche Klasse nimmt dabei die Rolle des Beobachters und welche die des Subjekts ein? (2 Punkte)
2. Für die Implementierung des Observer-Pattern stehen die beiden allgemeinen Klassen `Subjekt` und und `Beobachter` zur Verfügung. Wie verknüpfen Sie diese beiden Klassen mit den Klassen `Staedte` und `Karte`? (2 Punkte)
3. Die Klasse `Beobachter` enthält eine rein virtuelle Methode `void aktualisiere()`. Wie muss diese Methode in `Beobachter` deklariert werden? (2 Punkte)
4. Die Klasse `Beobachter` wird durch die rein virtuelle Methode `aktualisiere` zu einer abstrakten Klasse. Welche Konsequenzen hat das? (3 Punkte)
5. Welche Aussage ist richtig? (1 Punkte)
    a. Die Klasse `Staedte` hängt von der Klasse `Karte` ab.
    b. Die Klasse `Karte` hängt von der Klasse `Staedte` ab.
6. Durch welchen Qt-Mechanismus kann das Observer-Pattern leicht umgesetzt werden, ohne explizite `Subjekt`- und `Beobachter`-Klassen? Was muss dazu in den Klassen `Staedte` und `Karte` vorhanden sein (d.h. deklariert werden)? Wie heißt die Methode, mit der Objekte der Klasse `Staedte` mit Objekten der Klasse `Karte` verbunden werden können? (4 Punkte)