Klausur OOP SS2001 (Gampp)

Aufgabe 1 (22 Punkte)

a) Während einer bargeldlosen Giro-Bank Transaktion können Beträge zwischen Währungen konvertiert werden. Eine Klasse *Transaktion* soll solche Buchungen beschreiben und verwaltet neben dem eigentlichen Buchungsbetrag (Gleitkommazahl) auch eine Kennung für die Währung. In der Aufgabe soll die Währungskonvertierung als C++ String repräsentiert sein, mit Werten „DM“ bzw. „EUR“ für Euro. Ein Transaktionsobjekt soll neben Betrag und Währungskennung noch folgende Eigenschaften besitzen:
   - Es kann bei der Definition mit Betrag und Währungskennung versehen werden
   - Mit der Elementfunktion *convert* lassen sich Währung und damit auch Betrag verändern. Sie hat keinen Returnwert und als einzigen Parameter die Kennung der neuen Währung. Für die Aufgabe brauchen Sie nur Konvertierungen zwischen DM und Euro zu implementieren (in beide Richtungen), für andere Währungskombinationen hat die Methode keine Wirkung.
   - Die Elementfunktion *print* gibt den Zustand des Objekts aus. Die Ausgabe erfolgt mit genau 2 Nachkommastellen und nachgestellter Währung (z.B. „123.45 EUR“). Sie soll aber nicht nur auf der Konsole, sondern z.B. auch in eine Datei erfolgen können. Deshalb besitzt die Funktion einen Parameter der das Ausgabeobjekt festlegt, und als Typ eine geeignete Klasse der Standardbibliothek hat.

   Geben Sie eine Definition der Klasse *Transaktion* an, die den Prinzipien der Objektorientierung genügt. Dazu gehören auch die nötigen Include-Files !

   Hinweis:
   Definieren Sie eine Elementfunktion innerhalb der Klasse und eine außerhalb !
   Die Konvertierung zwischen DM und EUR erfolgt nach der Vorschrift: DM-Betrag = EUR-Betrag*1.95583

b) Geben Sie die kurze Hauptprogramm-Anweisungsfolge an, welche eine solche Transaktion mit 10 DM dynamisch erzeugt, sie in EUR konvertiert und dann auf der Konsole ausgibt.

c) Zur Ausgabe der Transaktion soll statt *print* der Operator << verwendet werden.Geben Sie die dazu nötigen Änderungen an.

Aufgabe 2 (16 Punkte)

Das folgende Quellfile wird gemäß Ansi-C++ Sprachstandard compiliert und gelinkt. Welche Fehler treten dabei auf. Markieren Sie die für die Fehler verantwortlichen Stellen und geben Sie neben dem Code oder auf der Rückseite eine kurze stichwortartige Fehlerbeschreibung.

Logische oder Laufzeitfehler sind nicht gefragt.

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

    class A {
         int a=2;
         public:
         A(int b) {
         a=b;
         }

    };


class B {
  public:

     void f(char& b) {
       A p;
       p.a=0;

     }
     void g() {}
     friend class A;

};

virtual void h() {
  B d;
  B.g();
```

```
  B e = new B;
  char* p = “Alles ok?”;
  cout << p;
  d.f(p);

}
```

Aufgabe 3 (15 Punkte)

Es liegt folgender fehlerfreier Programmausschnitt vor:

…

```
    class A {
      public:
      A() {
      cout << "A“ << endl;
      }

 virtual A* create() {
   cout << “create A\n“;
   return new A;

 }
};


class B: public A {

  public:
    B() {cout << “B” << endl;}
    A* create() {
    cout << “create B\n”;
    return new B; }

};


 void f(A x) {
   x.create();
   }

int main() {
  A a;
  f(a);
  B b;
```

```
  f(b);

}
```

a) Welche Ausgabe erzeugt die Auführung von *main()* ?

b) Ändern Sie das Programm so ab, dass der Parameter x der Funktion f ein Referenzparameter wird. Schreiben sie alle dafür nötigen Änderungen rechts neben die entsprechenden Zeilen des Programms! Wie lautet jetzt die Ausgabe des Programms?

Lösung:

Aufgabe 2)

1. using namspace std; <-fehlt
2. int a=2 nicht möglich da elementvariable.
3. A p; nicht möglich da kein parameterloser Konstruktor vorhanden ist.
4. p.a =0; nicht möglich da "a" private. friend steht zu weit unten.
5. Class B .... } " ;" fehlt
6. virtual void h() <- virtual ist nur in Klassen erlaubt.
7. B.g(); <- Klassenmethode bezieht sich nicht auf ein Objekt richtig ->d.g();
8. B e = new B; -> müsste B *e = new B; new B gibt ein Pointer zurück e ist aber kein Pointer.
9. d.f(p); -> p ist ein pointer die Funktion hat aber einen Referenzparameter.

Aufgabe 3)

a)
A
create A
A
A
B
create A
A

Wieso ist das so?
A a; -> Konstruktoraufruf von A -> cout "A"
f(a) -> cout "create A" -> nun wird mit return new A wieder der Konstruktor aufgerufen daher -> cout A
B b; -> B ist abgeleitete Klasse von A daher wird erst der Konstruktor von Class A aufgerufen und daher -> cout A nun wird der Konstruktor von B aufgerufen -> cout B
f(b) -> Achtung die Funktion hat einen Parameter vom Typ A daher wird die Funktion von Class A aufgerufen und daher auch selbe Ausgabe wie f(a)

b)
void f(A &x)
Durch den Referenzparameter wird nun A &x als das angesehen was übergeben wird in dem Fall B x und damit wird nun auch die Funktion von B aufgerufen.:
A
create A
A
A
B

create B
A
B
Funktion von Class B wird aufgerufen dann der Konstruktor von A dann von B siehe oben.