Übung 3

Klassen, Konstruktoren, Destruktoren

Hinweis: Für die Bearbeitung der folgenden Aufgaben müssen Sie jeweils mehrere C++ Source/Header-Dateien erstellen und übersetzen. Erstellen Sie dazu alle Quell- und Header-Dateien für eine Aufgabe jeweils in einem eigenen Unterordner.

Um ein ablauffähiges Programm zu erzeugen, können Sie mehrere Quell-Dateien bei der Übersetzung angeben

```
g++  <quelle1>.cpp <qelle2>.cpp …          (executable a.out)
g++ -o main <quelle1>.cpp <qelle2>.cpp     (executable mit-o)
```

Um eine Quell-Datei NUR zu übersetzen verwenden Sie:

```
g++ -c <quelle1>.cpp                  (erzeugt <quelle1.o>)
```

Das Ergebnis ist sogenannter Objektcode, der dann vom Binder (Linker) zu einem ablauffähigen Programm gebunden werden kann.

Bitte beachten Sie auch die Erläuterungen während der Übungen!

Aufgabe 3.1 (**Klasse mit dynamischen Datenelementen**)

a) Implementieren Sie eine Klasse **`Pkw`** nach folgenden Angaben.

- Definieren Sie dazu Datenelemente **`marke`** vom Typ `char *`, **`fgstnr`** (Fahrgestellnummer) vom Typ `unsigned long` und `leistung` vom Typ `unsigned int`.
- Implementieren Sie einen Konstruktor, der die angegebenen Datenelemente initialisiert. Geben Sie für `marke` und `leistung` die Defaultwerte "`Mercedes`" und `100` an. Der Konstruktor gibt die Meldung aus:

  "`Pkw der Marke xxx mit Fahrgestellnummer yyy und Leistung zzz erzeugt`"

  Dabei sollen in die Ausgabe jeweils die tatsächlichen Werte eingesetzt werden.
- Geben Sie Methoden zum Lesen und Schreiben (falls möglich) der Datenelemente an und setzen Sie die Zugriffsrechte geeignet.

b) Schreiben Sie eine Funktion main(), die ein Pkw-Objekt p1 mit Marke Beetle, der Fahrgestellnummer 1001 und Leistung 115 definiert. Anschließend wird die Belegung der Datenelemente von p1 ausgegeben.

Aufgabe 3.2 (Fragen)

Sei folgende Klassendefinition gegeben:

```
class X {
public:
   int a, *b;
   long c, *d;
   X(int d=0);
   ~X();
};
```

- Die Klassendefinition ist o korrekt o nicht korrekt

Beurteilen Sie die folgenden Definitionen. Begründen Sie nicht korrekte Fälle.

- `X::~X() { delete a;};` o korrekt o nicht korrekt
- `X::~X() { delete d;};` o korrekt o nicht korrekt
- `X::~X() { delete [] d;};` o korrekt o nicht korrekt
- `X::X(int e):a(e) { c=a;};` o korrekt o nicht korrekt
- `X::X(int e):a(e) { b=&e;};` o korrekt o nicht korrekt
- `X::X() { a=0;};` o korrekt o nicht korrekt

Weitere Fragen:

- Jede Klasse
  - besitzt einen Defaultkonstruktor o korrekt o nicht korrekt
  - hat einen Destruktor o korrekt o nicht korrekt

Aufgabe 3.3 (Komposition von Objekten)

a) Implementieren Sie eine Klasse `Notenliste` als verkettete Liste. Eine entsprechende Header-Datei „`noten.h`“ finden Sie in den Zusatzdateien `Add-On-3`.

b) Implementieren Sie nun die Klasse `Student`, die neben Matrikelnummer, Name und Semester auch eine Notenliste als Teilobjekt besitzt. Eine entsprechende Header-Datei „`student.h`“ finden Sie in den Zusatzdateien `Add-On-3`.

c) Schreiben Sie ein Hauptprogramm zum Test der Klasse `Student` z.B. geben Sie Ihre Matrikelnummer und Ihren Name an und die abgeschlossenen Semester inklusive der Noten ein. Lassen Sie sich Ihren Notendurchschnitt berechnen.

d) Ergänzen Sie die Klassen - sofern nötig - um Kopierkonstruktor und überladenen Zuweisungsoperator.