C23 – Laborübung 3

Kontext

Ein Taxiunternehmen benötigt ein Programm zur Erfassung und Abrechnung von Fahrten. Das Programm soll für jedes der Taxis den Tageskilometerstand, den aktuellen Tankinhalt und die Geldbilanz erfassen.

Grundelemente dieses Programms wurden in Labor 2 erstellt. In Labor 3 wird das Programm verändert und erweitert, um gutem Programmierstil zu entsprechen, und um als Grundlage für zukünftige Laboraufgaben dienen zu können.

Aufgabenstellung

Aufgabe 1 – Code-Modularisierung

a. Zerlegen Sie den Programm-Quellcode aus Labor 2 in Definition (.h-Datei) und Implementierung (.cpp-Datei).
b. Entscheiden Sie, welche Funktionen `inline` in der Header-Datei implementiert werden sollen, und welche Implementierungen in der .cpp-Datei vorgenommen werden sollen. Geben Sie für die `inline`-Funktionen eine kurze Begründung für Ihre Entscheidung als Kommentar im Quellcode.
c. Entscheiden Sie auch für die in Labor 3 noch zu implementierenden Funktionen bewusst, ob diese als `inline` implementiert werden sollen oder nicht.

Aufgabe 2 – Member-Variablen und Funktionen

a. Führen Sie eine neue private Member-Variable ein, in der ein Bezeichner für das Taxi gespeichert werden kann (als C++-String).
b. Schreiben Sie Zugriffsfunktionen `setName(…)` und `getName()` für diese Member-Variable, mit der ihr Inhalt festgelegt bzw. als String zurückgegeben werden kann. Der Taxi-Bezeichner soll nicht länger als 8 Zeichen sein; dies ist in der Funktion zu prüfen. Implementieren Sie eine sinnvolle Reaktion bei Abweichung von der 8-Zeichen-Vorgabe.
c. Erweitern Sie den Konstruktor so, dass ein Taxi-Bezeichner als Parameter übergeben werden kann und die entsprechende Member-Variable unter Einhaltung der 8-Zeichen-Vorgabe initialisiert wird.
d. Passen Sie Ihre Menü-Funktion so an, dass der Nutzer bei der Auswahl des Taxis 1 oder 2 deren Bezeichner angezeigt bekommt.

Aufgabe 3 – Ausgabe mit Strings und Streams

Hintergrund: Member-Funktionen sollen für gewöhnlich nicht direkt auf Ein- und Ausgabe-Streams zugreifen (guter Programmierstil).

a. Ersetzen Sie die Funktion `print()` der Klasse Taxi durch eine Member-Funktion `getState()`, die die aktuellen Zustandsinformationen eines Taxis in Spalten formatiert in einem String zurückgibt. Benutzen Sie zur Konstruktion dieses Strings die Möglichkeiten von Stringstreams aus der C++-Standardbibliothek. Der String soll folgendes Format haben (bitte beachten Sie die Breite der Spalten):

```
12345678901234567890123456789012345678901234567890
Name____ >> xxxx.xx km, xxxx.xx l, xxxx.xx Euro
Taxi_001 >>  235.00 km,   75.00 l,  823.50 Euro   <<< Beispiel
```

b. Benutzen Sie überall im Programm anstelle der Funktion `print()` die neue Funktion `getState()` zusammen mit `std::cout` zur Ausgabe der Statusinformationen der Taxis.

Aufgabe 4 – Konstruktoren (Fortsetzung aus Labor 2)

a. Ergänzen Sie die Geldbilanz als Default-Parameter im Konstruktor.
b. Beantworten Sie folgende Frage: Wie verhält sich die Anwendung, wenn Sie ein Objekt mit dem Default-Konstruktor (also ohne Angabe von Parametern) instanziieren, nachdem Sie einen eigenen Konstruktor implementiert haben? Beschreiben und erklären Sie das Verhalten (als Kommentar im Quelltext).
c. Taxi-Objekte sollen trotz eigenem Konstruktor zusätzlich über einen Standard-Konstruktor initialisiert werden. Ergänzen Sie die Klasse entsprechend.

Aufgabe 5 – Dynamische Speicherverwaltung

a. Verwalten Sie die „Taxi"-Objekte in einer Liste. Legen Sie dazu eine Liste von Zeigern auf „Taxi"-Objekte an, mit 2 Listenelementen. Erzeugen Sie die Taxi-Objekte dynamisch. Ändern Sie das Programm so, dass im gesamten Programm über die in der Liste gespeicherten Zeiger auf die „Taxi"-Objekte zugegriffen wird.
   Vergessen Sie nicht, den dynamisch allokierten Speicher am Ende des Programms wieder freizugeben.

Aufgabe 6 – Statische Variable

a. Zählen Sie die Instanzen der Klasse „Taxi" mit Hilfe einer statischen Member-Variablen. Verwenden Sie die Konstruktoren der Klasse, um die Variable hochzuzählen und den Destruktor, um sie wieder herunterzuzählen.
b. Überprüfen Sie, ob die Instanzen auch bei der folgenden Initialisierung korrekt gezählt werden:

```
Taxi a(…);
Taxi b(a);
```

   Falls nicht, dann suchen Sie nach der Ursache und ergänzen Sie das Programm, so das such bei der obigen Initialisierung die Instanzenzahl korrekt hochgezählt wird.
   Stellen Sie in Ihrer Lösung sicher, dass das Objekt `b` trotzdem eine Kopie von Objekt `a` ist.

c. Ergänzen Sie den Default Konstruktor so, dass unter Verwendung einer statischen Variablen ein eindeutiger Bezeichner generiert wird. Zählen Sie dazu die jemals erzeugten Instanzen des Typs Taxi, und erzeugen Sie aus dieser Zahl und einem Präfix einen Bezeichner der Art „Taxi_001" (der Präfix ist „Taxi_", und die Zahl soll immer vierstellig angezeigt und mit führenden Nullen aufgefüllt werden).
   Beachten Sie, dass Sie nicht die statische Member-Variable aus Aufgabe 6.a benutzen können, da diese wieder heruntergezählt wird wenn ein Destruktor aufgerufen wird – die Eindeutigkeit des Bezeichners wäre also nicht gegeben.