5. Der Entwurf der Klasse Menue

5.1 Analyse und Design des Menüs

Für das Projekt soll im Hinblick auf die Wiederverwendbarkeit eine allgemeingültige Klasse Menue implementiert werden.

☞ **Überlegen Sie, welche Eigenschaften und Methoden ein Menü besitzen muss!**
☞ **Das Menue sollte mit 2 bis 5 Menüpunkten erzeugt und ausgeführt werden können!**

Um ein Menü mit 2 bis 5 Menüpunkten erstellen zu können, muss das Menü in der Lage sein, bei der Erzeugung (Konstruktion) unterschiedlich viele Menüpunkte aufzunehmen.
Der Trick liegt im Überladen des Konstruktors.

Betrachten Sie dazu folgende zwei Beispiele:

```
Menue dasMenue1("Beenden","Start","Wiederholung");
Menue dasMenue2("Beenden","Aktion starten","Buchung","Abbrechen");
```

Menue1 verfügt über drei, Menue2 über vier verschiedene Menüpunkte.

☞ **Über welche Eigenschaften muss ein allgemeines Menue verfügen?**

Zum Speichern der Menüpunkte verwendet man ein Array mit strings. Die Anzahl der enthaltenen Menüpunkte sind ebenfalls als Attribut des Menüs zu implementieren. Mit der Methode char ZeigMenue( ) werden alle Punkte ausgegeben und die getroffene Benutzerauswahl als char zurückgeliefert.

☞ **Erstellen Sie ein Klassendiagramm für die Klasse Menue!**
☞ **Tragen Sie bei den Methoden auch die überladenen Konstruktoren ein!**

5.2 Die Realisierung des Menüs

Bei der Klassendefinition müssen natürlich alle unterschiedlichen Konstruktoren deklariert werden. Vergessen Sie auch nicht die Initialwerte im Standard – Konstruktor zu definieren!

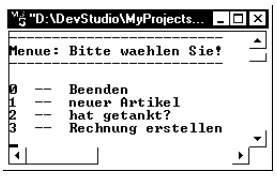
Die eigentliche Definition der Konstruktoren erfolgt dann in der Quelltextdatei „Menue.cpp“.
Anhand des überladenen Konstruktors wird die Anzahl der Menüpunkte gesetzt.
Anschließend kann im Freispeicher das Array aMnuPunkte erzeugt und mit den übergebenen Werten gefüllt werden.

```
// Konstruktor mit zwei Menuepunkten
Menue::Menue(string a, string b)
{
    // Setzt die Anzahl der Menuepunkte
    anzahl = 2;
    // initialisiert das Menüpunkt - array
    aMnuPunkte = new string [anzahl];
    aMnuPunkte[0]=a;
   aMnuPunkte[1]=b;
}
```

Auch der Destruktor des Menüs ist gefordert, da das Menüpunkte - Array im Heap erzeugt wird.
Verliert das Menü seine Gültigkeit, so ist der Freispeicher wieder "zu entrümpeln".

```
Menue::~Menue( )
{
    if(aMnuPunkte != NULL)
        delete [ ] aMnuPunkte;
}
```

Die Funktion ZeigMenue( ) gibt alle Menüpunkte auf dem Bildschirm aus und verlangt vom Benutzer eine Eingabe. Am Ende wird die Eingabe zurückgegeben.
Zur Ausgabe der Menüpunkte aus dem Array benutzt man am einfachsten eine for – Schleife!

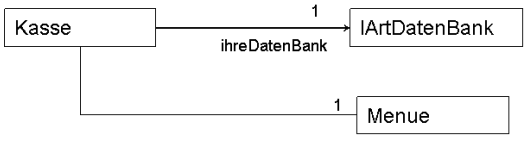


☞ **Implementieren Sie die Klasse Menue im Programm!**

6. Die Klasse Kasse

6.1 Analyse und Design der Kasse

Als nächstes soll das Grundgerüst der Kasse erstellt werden. Da die Kasse mit vielen anderen Klassen des Projekts in Beziehung steht, kann das vollständige Design erst im Laufe des Projekts erarbeitet werden.
Andererseits wiederum können andere Klassen erst nach der Kasse implementiert werden (Henne oder Ei Syndrom ☺).



☞ **Überlegen Sie, welche Funktionen über das Kassenmenü abgefragt werden können!**

Die Steuerung des Programmablaufs sollte die Kasse übernehmen. Dazu verfügt Sie über die Methode StarteAktion( ), die erst beim Beenden des Programms wieder verlassen wird.
Hat der Kunde getankt bzw. eine Ware gekauft, so wird die Weiterverarbeitung des Vorgangs in der Methode Buche( ) ausgeführt.

☞ **Erstellen Sie ein vorläufiges Klassendiagramm für die Klasse Kasse!**

6.2 Die Realisierung der Kasse

6.2.1 Die Headerdatei Kasse.h

Zunächst soll die Headerdatei mit der Klassenschnittstelle für die Kasse erzeugt werden. Überlegen Sie, welche Headerdateien mit einzubinden sind. Die IArtikelDatenBank kann über den Namensraum DB_Treiber eingebunden werden!

☞ **Implementieren Sie die Klassendefinition der Kasse!**

6.2.2 Die Implementierung der Methoden in Kasse.cpp

Die Methoden der Kasse sollen bis auf den Konstruktor und StarteAktion( ) zunächst unberücksichtigt bleiben. Um das Projekt trotzdem kompilieren und testen zu können müssen void Buche( ), void Getankt( ), void NeueWare( ) und void Rechnung( ) aber mit leerem Funktionsrumpf definiert werden.

☞ **Erstellen Sie die genannten Methodendefinitionen mit leerem Funktionsrumpf!**

Der **Konstruktor** übernimmt laut Klassendiagramm einen Zeiger auf eine I**ArtDatenBank**. **ihreDatenBank** der Kasse soll mit der übergebenen Datenbank initialisiert werden! Damit ist die geplante Assoziation mit der Artikeldatenbank hergestellt.

Das **Menue** der Kasse ist ebenfalls im Konstruktor zu spezifizieren:
Betrachten Sie die Erzeugung des Menüs in der Klassendefinition der Kasse! Normalerweise müssten dem Konstruktor des Menüs die Menüpunkte als Parameter übergeben werden. Da die Ausführung einer Methode innerhalb einer Klassendefinition nicht möglich ist, wird ihrMenue nur mit dem Standardkonstruktor (ohne Menüpunkte) erzeugt.

Das richtige Menü der Kasse kann über den Konstruktor der Kasse initialisiert werden!

Da die Kasse das Menue im Heap erzeugt, ist die Kasse auch für die Zerstörung des Menüs zuständig.

```
Kasse::Kasse( IArtDatenBank* db )
{
...
 ihrMenue = new Menue("Beenden","neue Ware","hat getankt?","Rechnung erstellen");
}
Kasse::~Kasse( )
{
 delete ihrMenue;
}
```

☞ **Programmieren Sie den C++ Quellcode für den Konstruktor der Kasse!**
☞ **Erklären Sie den oben aufgeführten Quelltextausschnitt des Konstruktors der Kasse!**

Die Methode **void StarteAktion( )** ist das eigentliche Steuergrundgerüst der Kasse. Von hier aus werden die Menüeingaben des Benutzers abgefragt und ausgewertet.
Für die Methode eignet sich eine do... while – Schleife mit einer switch – Anweisung, welche die Benutzereingaben weiterleitet.

- 0 – Beenden — Das Kassenprogramm wird abgebrochen.
- 1 – Neue Ware — Die Methode NeueWare( ) wird aufgerufen.
- ... — ...

Zum Abfangen von Fehleingaben sollten Sie einen Default – Zweig einrichten!

☞ **Implementieren Sie die Methode void StarteAktion( )!**

6.3 Der Einsatz von Kasse und Menue im Programm

Nach den Implementierungen des Menüs und der Kasse erfolgt der Test der zweiten Version.
Bei den Änderungen der Methode Run( ) der Applikation müssen Sie die Reihenfolge der Instantiierungen von ArtDatenBank und Kasse beachten.

☞ **Schreiben Sie in der Methode void Applikation::Run( ) ein kurzes Testprogramm, das Objekte der Klassen Kasse und ArtDatenbank erzeugt und anschließend die Methode StarteAktion( ) der Kasse ausführt!**