Programmieren C++:
Abstrakte Klassen, statische Member und Methoden

Klaus Kusche

Gesucht ist ein Programm, mit dem die EDV-Geräte einer Firma inventarisiert werden können: Es soll u.a. möglich sein, eine Liste aller EDV-Geräte und den aktuellen Wert der gesamten EDV-Ausstattung auszugeben.

Das Programm sollte wieder auf getrennte Header- und Code-Files pro Klasse und einen File für **`main`** aufgeteilt werden.

Alle inventarisierten Gegenstände werden durch ein Objekt einer von **`Inventar`** *abgeleiteten* Klasse dargestellt. Objekte der Klasse **`Inventar`** selbst darf es *nicht* geben.

Die Klasse `Inventar`

… bietet folgende Eigenschaften:

- Jeder Gegenstand hat einen *Namen* (verwende die C++-Klasse "**`string`**" aus dem Header **`<string>`**, sie funktioniert ähnlich den "normalen" Datentypen **`int`** usw.). Der Name muss *beim Konstruktor angegeben* werden (es gibt daher *keinen* Standard-Konstruktor), abgeleitete Klassen können direkt darauf zugreifen.
- Weiters hat jeder Gegenstand eine *Inventarnummer*. Sie wird nicht explizit angegeben, sondern *im Konstruktor automatisch mit der nächsten fortlaufenden Nummer belegt*. Sie soll für abgeleitete Klassen *nicht* direkt sichtbar sein.

  Erinnere dich: Für eine automatische Nummerierung brauchst du zwei Member!
- Für beides soll es je eine *Get-Funktion* (**`getName`** und **`getInvNr`**) geben, aber keine Set-Funktion.
- Weiters gibt es *vier virtuelle Methoden*, die in **`Inventar`** *nicht implementiert* sind:
  - **`getValue`** *liefert den Wert des Gegenstandes* und **`setValue`** *setzt den Wert neu*.
    (der Wert selbst ist *kein* Member von Inventar,
    er wird in den abgeleiteten Klassen unterschiedlich gehandhabt).
  - Eine **`print`**-Methode gibt *alle Informationen* zu einem Gegenstand mit **`cout <<`** aus (Inventarnummer, Name, Wert, ev. andere Informationen).
  - **`isComp`** liefert ein boolsches Ergebnis:
    "Ist der Gegenstand ein Computer (**`true`**) *oder etwas anderes* (**`false`**) *?"*
    (Dafür gibt es in C++ bessere Mechanismen, aber die lernen wir noch nicht.)
- *Was sollte außer diesen vier Funktionen noch virtuell sein?*
  (wir brauchen dieses "Ding" in **`Inventar`** zwar nicht,
  aber es soll in abgeleiteten Klassen überschrieben werden!)

Die Klasse Zubehör

Sie ist *von* **`Inventar`** *abgeleitet* und für alle Gegenstände gedacht, die *kein Computer* sind.

Sie enthält eine Member-Variable für den *Wert des Gegenstandes*.
Der ursprüngliche Wert wird beim Konstruktor angegeben, er kann
mit den virtuellen Funktionen **getValue** und **setValue** gelesen und geändert werden.

Die Klasse Computer

Sie ist ebenfalls von **Inventar** abgeleitet, aber viel umfangreicher, weil Computer häufig aufgerüstet werden und RAM und Disk starken Wertschwankungen unterliegen:
Der Wert eines Computers setzt sich zusammen aus dem *Wert des Grundgerätes*
und dem *Wert von RAM und Disk* (mengenabhängig, soviel eben gerade eingebaut ist).

- Der *Wert des Grundgerätes* wird im *Konstruktor* angegeben
  und in einer *Member-Variablen* gespeichert.
  Er kann mit der virtuellen Methode **setValue** nachträglich geändert werden.
- Weiters wird im Konstruktor angegeben, *wie viele GB RAM* und *wie viele GB Disk* der Computer hat. Auch dafür gibt es *zwei Member-Variablen* mit je einer *Get-Methode* (**getRamGB** und **getDiskGB**) sowie je einer *Methode* **addRamGB** bzw. **addDiskGB**, um das RAM bzw. die Disk-Kapazität des Computers zu *erhöhen*
  (oder mit negativem Argument zu erniedrigen), wenn er umgerüstet wird.
  Die beiden Werte werden auch von der virtuellen **print**-Funktion mit *ausgegeben*.
- Die Methode **getValue** *für den Wert* berechnet den Wert wie oben angegeben aus dem Grundgeräte-Wert + dem Wert der *verbauten GB RAM und Disk*, wobei in der Klasse für RAM und Disk *jeweils ein klassenweit einheitlicher Preis pro GB* gespeichert ist.

  Was brauchst du dafür?

  Für diese beiden klassenweiten Preise gibt es ebenfalls jeweils
  eine (*klassenweite*!) Get-Methode (**getRamValPerGB** und **getDiskValPerGB**)
  und eine Set-Methode (**setRamValPerGB** und **setDiskValPerGB**).
- *Zusatzaufgabe:*

  Weiters ist noch gefordert, dass jederzeit die *Gesamtzahl der Computer*
  in der Firma, die Menge des *insgesamt vorhandenen RAM*,
  und die *insgesamt vorhandene Disk-Kapazität* mit entsprechenden *Get-Methoden*
  getCompCount, **getTotalRamGB** und **getTotalDiskGB** gelesen werden kann
  (und zwar *ohne alle Computer-Objekte durchzugehen!*).

  Was brauchst du, um diese drei Werte zu speichern?

  Von welchen Methoden / Pseudo-Methoden müssen diese Werte aktualisiert werden?

  Wie müssen die Get-Methoden deklariert sein, damit sie ohne Angabe
  eines konkreten Computer-Objektes aufgerufen werden können?

Hauptprogramm

Das Hauptprogramm dient nur zum Testen der Klassen und Methoden. Es soll intern ein Array von Pointern auf **`Inventar`**-Objekte enthalten und dem Benutzer in einer Schleife eine Auswahl an einzelnen Operationen bieten:

- Programm beenden.
- Einen Computer neu inventarisieren oder ein Zubehör neu inventarisieren.
- Einen Inventar-Eintrag löschen.
- Einem Inventar-Eintrag einen neuen Wert geben.
- Die RAM- und Disk-Ausstattung eines Computers ändern.
- Den Wert pro GB RAM / Disk ändern.
- Die Gesamtliste und den Gesamt-Wert aller inventarisierten Gegenstände anzeigen oder einen einzelnen Artikel anzeigen.
- Die Anzahl der Computer und die Summe von RAM und Disk-Kapazität anzeigen.

Einzelne bestehende Geräte werden dabei über ihre *Inventarnummer* angegeben.