FH Osnabrück
FB Elektrotechnik und Informatik
Prof. Dr.-Ing. B. Lang

Objektorientierte Programmierung
Praktikumsübung 7

Beschreibung der Aufgabenstellung

In der vorliegenden Übung soll ein animierter AVI-Videofilm erstellt werden. Dazu wird schrittweise ein Container für Objekte der `GrafikElement`-Klassenhierarchie (siehe letzte Übung) entworfen. Die in einem Container abgespeicherten `GrafikElement`-Objekte werden dann in ein Bild eingezeichnet und das Bild in einen AVI-Film gespeichert. Anschließend werden die `GrafikElement`-Objekte im Container verändert, erneut in ein Bild gezeichnet und im AVI-Film gespeichert. Dies wird bis zur vorgegebenen Länge des Films wiederholt.

Aufgabe 1: Erstellung einer Klasse `Kiste` für eine Klassenhierarchie: (4 Punkte)

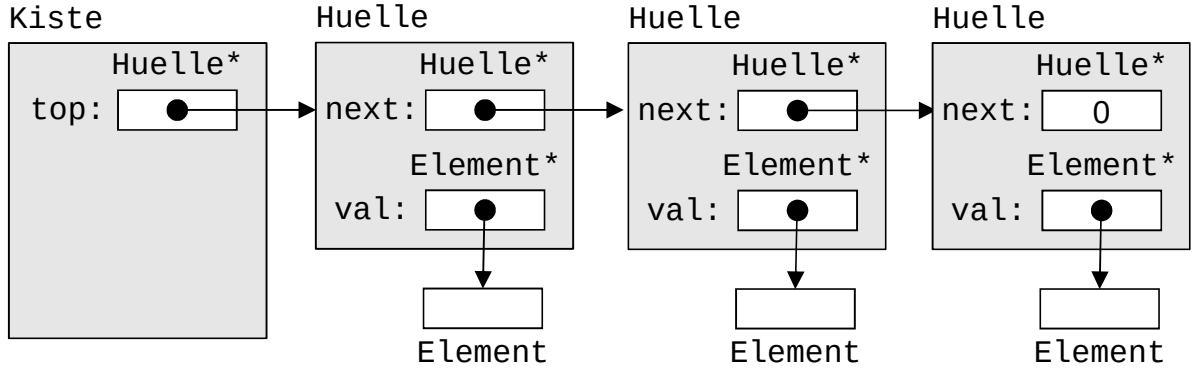
Es soll eine Containerklasse `Kiste` entworfen werden, die Objekte von einer Basisklasse `Element` und von deren Unterklassen speichern kann. Zum Kopieren eines Objekts besitzen die Klasse `Element` und ihre Unterklassen eine virtuelle `clone`-Methode.

Die Containerklasse wird mit den folgenden öffentliche Methoden ausgestattet:

- **`Kiste()`**: Konstruktor: Erzeugt einen leeren Container.
- **`~Kiste()`**: Destruktor: Löscht alle `Element`-Objekte, die im Container stehen, und löscht alle intern im Container angeforderten dynamischen Objekte.
- **`operator int()`**: Gibt die Anzahl der im Container eingeschriebenen `Element`-Objekte zurück (Füllstand des Containers).
- **`Kiste& operator<< (const Element &v)`**: Speichert eine Kopie des übergebenen `Element`-Objekts im Container. Zum Kopieren des Objekts muss die clone-Methode verwendet werden, welche den wirklichen Typ des Objekts beachtet. Das übergebene Objekt wird nicht verändert, daher der `const`-Parameter.
- **`Kiste& operator<< (const Kiste& r)`**: Kopiert alle `Element`-Objekte, die in `r` enthalten sind, und fügt die Kopien in den aktuellen Container ein. Beim Kopieren der Objekte muss der wirkliche Typ jedes Objekts beachtet werden. Bei der Ausführung des Operators darf der Container `r` nicht modifiziert werden, daher der `const`-Parameter.
- **`Element& operator[] (int i)`**: Gibt eine Referenz auf das im Container gespeicherte Objekt mit Index `i` zurück.

Zuweisungsoperator und Kopierkonstruktor sollen gesperrt sein.

Das Abspeichern der Elemente in der `Kiste`-Klasse soll durch eine verzeigerte Liste erfolgen. Eine Hilfsklasse `Huelle` kann beim Aufbau der verzeigerten Liste helfen.



Mindestpunktzahl für die 7. Praktikumsübung: 5 Punkte

Testen Sie Ihre Klasse mit dem Testprogramm „*Test_Kiste.cxx*“, welches die Klasse `Element` und die daraus abgeleitete Klasse `UnterElement` verwendet, die beide in „*Element.h*“ vereinbart sind.

Machen Sie sich die Abhängigkeiten zwischen den Klassen `Element` und `UnterElement` klar.

Analysieren Sie die Anforderungen, die Ihre `Kiste`-Klasse an die Klasse `Element` stellt.

Aufgabe 2: Klasse `Kiste` in Template überführen: (4 Punkte)

Überführen Sie Ihre Kiste-Klasse in ein Template `TKiste`. Als Template-Parameter wird die Element-Klasse `E` übergeben:

```
template<class E> class TKiste {
  ...
};
```

Spezifizieren Sie die Anforderungen an den Datentyp `E`.

Testen Sie Ihr Template mit dem Testprogramm „*Test_TKiste.cxx*“, welches unter anderem die `GrafikElement`-Klassenhierarchie aus Versuch 6 verwendet.

Aufgabe 3: Ableitung einer Klasse `GrafikKiste` aus `TKiste`: (2 Punkte)

Leiten Sie aus der Klasse `TKiste<GrafikElement>` eine Klasse `GrafikKiste` ab, welche die folgenden zusätzlichen Methoden bereitstellt:

`void add_offset (int x, int y)`
: Addiert zu allen Objekten, die im Container gespeichert sind, den Offsetwert `x`,`y`.

`void draw (Image& I)`
: Zeichnet alle im Container enthaltenen `GrafikElement`-Objekte in das Bild `I` ein. Als Reihenfolge zum Zeichnen der Elemente wird die Reihenfolge verwendet, mit der die Elemente in den Container eingeschrieben wurden.

Die Klassendefinition ist in der Datei „*GrafikKiste.h*“ angedeutet. (Hinweis: Die Klasse `GrafikKiste` ist kein Template.)

Zum Test der Klasse `GrafikKiste` ist das Testprogramm „*Test_GrafikKiste.cxx*“ vorgegeben, welches einen kurzen AVI-Videofilm berechnet.

☞ Machen Sie sich deutlich, wo die Methoden und Operatoren der erstellten Klassen im Testprogramm aufgerufen werden.

Hinweis: Das Testprogramm „*Test_GrafikKiste.cxx*“ verwendet die Klasse `AviWrite` zur Erzeugung des AVI-Films. Beim Erzeugen eines `AviWrite`-Objekts wird der Dateiname und die Breite und Höhe der Bilder spezifiziert. Als dritter Parameter kann die Bildwiederholrate angegeben werden, voreingestellt ist der Wert 25 Bilder pro Sekunde. Anschließend können Bildobjekte vom Typ `Image` über den Ausgabeoperator << in das AVI-Objekt eingeschrieben werden.

Das Testprogramm legt einen ersten Container `f1` an, in welchen erste Grafikobjekte eingeschrieben werden. In einer Schleife wird dann ein Container `f` angelegt und gefüllt. Am Ende wird der Inhalt des Containers `f1` über den Ausgabeoperator << in den Container `f` kopiert.

Der Inhalt des Containers `f` wird in ein `Image`-Objekt gezeichnet, welches anschließend in das AVI-Objekt geschrieben wird.

Hinweis: Die Darstellung des AVI-Films auf den SUN-Rechnern des EDVSZ sollte mit dem vorhandenen Player möglich sein. Auf Windows-Rechnern können die erzeugten AVI-Videos mit dem Programm VirtualDub (www.virtualdub.org) angezeigt und umkodiert werden.

Mindestpunktzahl für die 7. Praktikumsübung: 5 Punkte