FH-OÖ Hagenberg/HSD
Software Entwurf 2, SS 2011
Übung 11

Name: Abgabetermin: KW 26

Mat.Nr: Punkte:

Übungsgruppe: korrigiert:

Aufwand in h:

Beispiel 1 (24 Punkte) Klassenhierarchie "GraphicObject": Implementieren Sie die angegebene Klassenhierarchie:

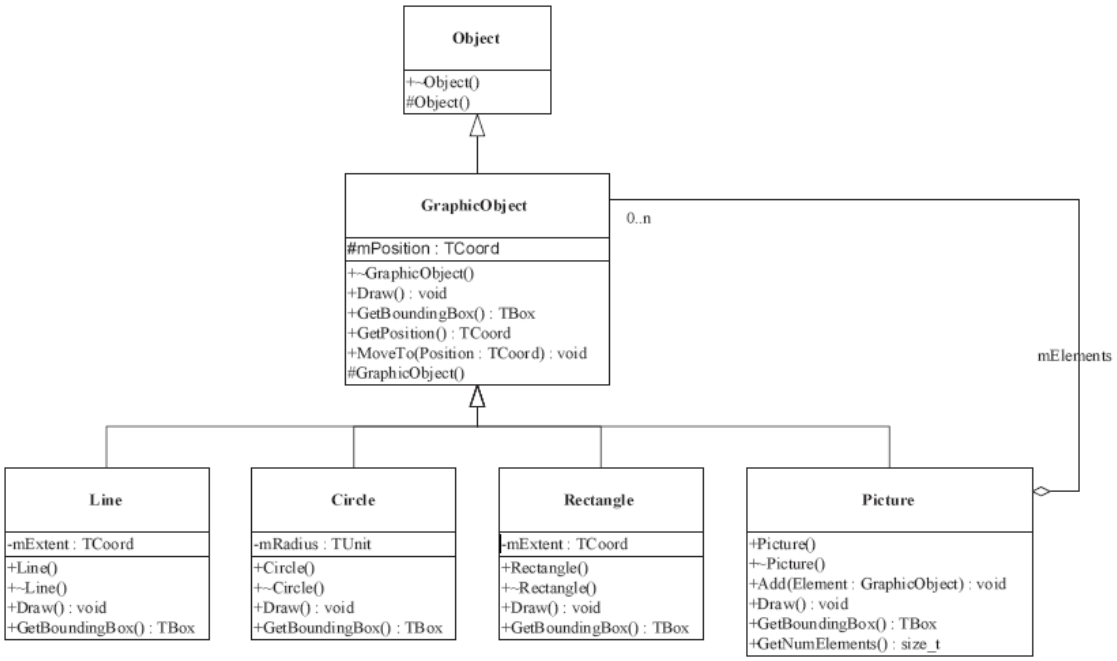

Bild 1: Die Klassenhierarchie "GraphicObject"

Entscheiden Sie selbst,

- wie Sie die Datentypen `TBox`, `TCoord` und `TUnit` definieren (verwenden Sie `typedef` und `pair`),
- welche Methoden Sie abstrakt (`virtual`) und welche Methoden Sie konkret implementieren,
- wie Sie die Datenkomponente `Picture::mElements` und ihre Elementtypen entwerfen (verwenden Sie einen polymorphen Container),
- wo Sie Delegation verwenden,
- welche Datentypen Ihre Methoden-Parameter genau besitzen (verwenden Sie sooft als möglich const und "&").

Es gelten:

`TUnit` ... für positive und negative ganzzahlige Koordinatenwerte
`TCoord` ... für ein Koordinatenpaar x, y
`TBox` ... für zwei Koordinatenpaare zur Angabe eines umschließenden Rahmens

`mPosition` entspricht der Ursprungsposition jedes `GraphicObject` und ist anfänglich (0, 0). `MoveTo()` versetzt die Ursprungsposition `mPosition` auf die übergebene Position, `GetPosition()` liefert sie zurück. `GetBoundingBox()` liefert die Koordinaten des umschließenden Rahmens.

Eine Linie etwa wird beginnend von `mPosition` bis zu `mExtent` 'gezeichnet'. Analog hat ein Kreis seinen Mittelpunkt in `mPosition` und ein Rechteck eine Ecke. Ergänzen Sie die nötigen *overridden constructor* bzw. Methoden um diese Objekte zu erzeugen.

Ein `Picture` kann beliebig viele `GraphicObjects` (und damit auch ganze `Pictures`) aufnehmen. `GetBoundingBox()` liefert hier die Koordinaten des umschließenden Rahmens vom gesamten Bild. `GetNumElements()` gibt die Anzahl der `GraphicObjects` zurück, aus denen ein Bild besteht (2 `Circle` + 1 `Picture` = 3). Die `Draw()`-Methode in `Picture` gibt alle enthaltenen `GraphicObjects` aus - auch jene von weiteren `Pictures`.

Begründen Sie alle Ihre Entscheidungen ausführlich! Testen Sie Ihr Programm, indem Sie in die `Draw`-Methoden sinnvolle und aussagekräftige Textausgaben (`ostream`) einbauen.

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Verwenden Sie immer **Module**, um den Testtreiber und die eigentliche Implementierung zu trennen! Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit **Kommentaren! Testen** Sie ihre Implementierungen ausführlich! Geben Sie **Lösungsideen** an!