Programmieren 1 Übung: Geometrische Objekte: Rechteck-Klasse (Init-Liste, Copy-Konstruktor, dyn. Anlegen, ...)

Klaus Kusche

Diese Übung baut auf dem Beispiel aus der *vorigen Übung* auf.

1. Baue wie in der vorigen Übung beschrieben die Klasse für Punkte auf eine **Klasse für Rechtecke** um (mit *zwei zusätzlichen Member-Variablen und Parametern im Konstruktor* für die Ausdehnung in Pixeln ab Mittelpunkt in x- und y-Richtung).
2. Schreib die in der vorigen Übung angegebenen **zusätzlichen Methoden**:
   - Eine Methode **`setSize`** mit zwei Parametern:
     Sie soll die *Größe*, d.h. x- und y-Ausdehnung des Rechtecks, *neu setzen*.
   - Eine Methode **`scale`**: Sie wird mit zwei **`int`**-Werten aufgerufen,
     die *Prozentwerte* darstellen, mit denen *die Breite und die Höhe zu skalieren* ist
     (d.h. 100 ... bleibt gleich, 200 ... doppelt so groß, 50 ... halb so groß).
   - Eine Methode **`rotate`** ohne Parameter: Sie *vertauscht* Höhe und Breite.

   Alle diese Methoden sollen als erstes das alte Rechteck weglöschen
   und am Ende das neue Rechteck zeichnen.
   - Auch *zwei* **`get`**-*Methoden* für die beiden neuen Member wären praktisch,
     damit man im **`main`** die *aktuelle Größe des Rechteckes abfragen* kann.

   Ändere dein *Hauptprogramm*, um ein Rechteck statt einen Punkt anzulegen
   und die neuen Methoden zu testen! (Größe des Rechteckes ändern, ...)
3. Bau die *Konstruktoren* von **`Color`** und **`Rechteck`** so um,
   dass sie eine **Initialisierungsliste** verwenden!

   Wenn der Umbau geglückt ist, solltest du die *Default-Werte*
   für die Parameter des Color-Konstruktors wieder *entfernen* können,
   denn dann sollte *kein Standard-Konstruktor* für **`Color`** mehr benötigt werden!

   Warum wurde er bisher benötigt?
4. Trenne dein Programm sauber in **`.h`-Files und `.cpp`-Files** auf:
   - **`Color`** hat *keine* Methoden außerhalb der Klasse, es reicht also ein **`.h`**-File.
   - Für die Rechteck-Klasse brauchst du *beide* Files.
   - Und das **`main`** bekommt auch einen *eigenen* **`.cpp`**-File.

   Vergiss nicht auf den *Schutz gegen mehrfaches Include* in deinen **`.h`**-Files (wie in C).
5. Lege in deinem Hauptprogramm ein **Array von Pointern auf Rechteck-Objekte** an.

   Fülle es mit Werten, indem du in einer Schleife entsprechend viele Rechtecke
   einzeln mit **`new`** anlegst, und zwar jeweils als *Kopie* jeweils vorigen Rechteckes.

   Rufe für jedes der so angelegten Rechtecke in einer Schleife *irgendeine Methode auf*
   (bewege es z.B. an eine zufällige Stelle).

   Gib die Rechecke dann einzeln in einer Schleife wieder *frei*.

6. Der automatisch erzeugte Copy-Konstruktor für Rechtecke hat zwei Nachteile:
   - Die Kopie liegt *exakt über dem Original* und verdeckt es daher komplett.
   - Die Kopie wird im Unterschied zu anderen neuen Rechtecken *nicht sofort angezeigt* (weil im automatischen Copy-Konstruktor ja kein **`draw()`** aufgerufen wird).

   Implementiere daher einen **Copy-Konstruktor** für Rechtecke (erinnere dich: Ein Copy-Konstruktor bekommt eine *konstante Referenz* auf das Original-Objekt!): Er soll die Kopie *um ein paar Pixel kleiner und etwas dunkler* anlegen als das Original und gleich zeichnen.

   Verwende wieder eine *Initialisierungsliste*!

   Hinweis:
   Arbeite dann in deinem Hauptprogramm zuerst einmal mit der Kopie weiter, denn sobald du etwas am Original änderst, *übermalt* das Original wieder die Kopie!

7. Schreib eine *Methode* **`moveOnTop`**, die ein *Rechteck als Parameter* hat und das *eigene Rechteck über das übergebene Rechteck legt* (d.h. das eigene Rechteck an der alten Position löscht, den Mittelpunkt des eigenen Rechteckes auf den Mittelpunkt des übergebenen Rechteckes setzt und dann das eigene Rechteck neu zeichnet).

   Verwende diese Methode, um die Rechtecke aus dem vorigen Punkt vor dem Löschen *wieder alle aufeinander zu legen*, und zwar einzeln mit einer kleinen Verzögerung zwischen zwei **`moveOnTop`**-Aufrufen.

   Experimentiere mit der Parameter-Übergabe: Deklariere den Parameter von **`moveOnTop`** einmal *"by Value"* und einmal *"by Reference"*.

   Siehst du einen Unterschied? Kannst du den Unterschied erklären?