IMS
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Programmierpraktikum Technische Informatik (C++)
Programmiertutorium Vorlesung 5

Pullen Sie von ppti-common. Sie finden dann in ihrem Repository den Ordner `tutorials/Lecture05/`, in dem die folgenden Aufgaben zu bearbeiten sind.

Teilaufgabe 1

Schreiben Sie eine polymorphe Basisklasse `Shape` mit abgeleiteten Klassen `Rectangle` und `Circle`!

a) Schreiben Sie zunächst `Shape` als polymorphe Basisklasse mit einer abstrakten Memberfunktion `getArea()`, die in den abgeleiteten Klassen überschrieben werden soll und die Fläche des `Shape` als `double` zurückgeben soll!

b) Schreiben Sie die Klasse `Rectangle`, die von `Shape` erbt und ein Rechteck darstellt! Diese Klasse soll bei der Konstruktion zwei `double` Werte `dimX` und `dimY` übergeben bekommen, die die Kantenlängen des Rechtecks angeben. Implementieren Sie die geerbte Funktion `getArea` so, dass als Fläche des Rechtecks das Produkt der beiden bei der Konstruktion übergebenen Kantenlängen zurückgegeben wird!

c) Schreiben Sie die Klasse `Circle`, die ebenfalls von `Shape` erben soll und einen Kreis darstellt! Der Konstruktor von `Circle` soll den Radius des Kreises als `double` Wert übergeben bekommen. Die Funktion `getArea()` soll die Fläche des Kreises nach der Formel πr^2 berechnen. Verwenden Sie für π die in der Standardbibliothek vorhandene Konstante `M_PI`!

Hinweise:

- `M_PI` ist im Header `cmath` definiert, den Sie erst noch einbinden müssen!
- Sie dürfen den Klassen beliebige Membervariablen hinzufügen, solange diese `private` sind.
- Initialisieren Sie alle Membervariablen per Initialisierungsliste im Konstruktor!
- Kommentieren Sie zum Testen ihrer Lösung die Defintion der Präprozessorkonstante `A1` ein!

Appelstraße 4 | 30167 Hannover
fon 0511 762-19691 | fax 0511 762-19694
www.ims.uni-hannover.de | info@ims.uni-hannover.de

Teilaufgabe 2

Schreiben Sie eine Funktion `printShape`, die einen `std::ostream` und `Shape` übergeben bekommt und den Typ des `Shape` auf dem übergebenen `std::ostream` ausgeben soll! Wird ein `Rectangle` übergeben, so soll auf der Konsole also `"Rectangle"` ausgegeben werden, wird `Circle` übergeben, so soll `"Circle"` ausgegeben werden. Kommentieren Sie zum Testen ihrer Lösung die Defintion der Präprozessorkonstante `A2` ein!

Hinweise:

- Die Funktion soll die verschiedenen Klassen ohne Verwendung von dessen Membern unterscheiden!
- `dynamic_cast<const Derived*>(ptr)` kann verwendet werden, um zu prüfen, ob ein Pointer `ptr` auf eine Basisklasse auf ein Objekt vom abgeleiteten Typ `Derived` zeigt.

Teilaufgabe 3

Schreiben Sie eine Funktion `createShapes`, die einen Vektor `vec` von `std::tuple<ShapeType, double, double>` übergeben bekommt und einen Vektor von **polymorphen** `Shape` zurückgibt. Dabei soll für jedes Element von `vec` abhängig von dessen `ShapeType` ein `Rectangle` (für `ShapeType::Rectangle`) bzw. ein `Circle` (für `ShapeType::Circle`) erstellt und dem Ergebnisvektor hinzugefügt werden.
Für ein `Rectangle` sollen das zweite und dritte Element des Tuples als Kantenlängen verwendet werden. Für einen `Circle` gibt das zweite Element des Tuples den Radius an und das dritte ist unbenutzt. Kommentieren Sie zum Testen ihrer Lösung die Defintion der Präprozessorkonstante `A3` ein!

Hinweise:

- Erinnern Sie sich, dass Polymorphie in C++ nur über Pointer oder Referenzen möglich ist und Referenzen nicht in Containern gespeichert werden können!
- Verwenden Sie keine manuelle Speicherverwaltung (`new`/`delete`)!