# Übung 14 infT

**1.**
Erstellen Sie eine abstrakte **`Shape`**-Klasse als Basisklasse. Von dieser Klasse werden die konkreten Klassen **`Circle`**, **`Square`** und **`Triangle`** abgeleitet. Stellen Sie für die Klasse sinnvolle Konstruktoren zur Verfügung. Definieren Sie in **`Shape`** eine pure virtuelle Funktion **`draw()`** und überschreiben Sie diese in den abgeleiteten Klassen.
Erstellen Sie nun ein Array mit Pointern auf **`Shape`**-Objekte. Solche Objekte (Upcasting!) erstellen Sie nun auf dem Heap und referenzieren Sie via die Array-Elemente. Rufen Sie nun **`draw()`** via die Basis-Klassen-Pointer auf und verifizieren Sie das Funktionieren des späten Bindens.

**2.**
Ergänzen Sie die **`Shape`**-Klassen aus Aufgabe **1.** mit virtuellen Destruktoren, die sich an der Konsole melden. Verifizieren Sie jetzt, dass die Destrukturen gegebenenfalls auch aktiviert werden.

**3.**
Schreiben Sie ein kleines Programm, das den Unterschied zeigt zwischen dem Aufruf einer virtuellen Funktion aus einem Konstruktor und dem Aufruf einer virtuellen Funktion einer normalen Member-Funktion. Erweitern Sie sodann das Programm, indem Sie auch eine virtuelle Funktion aus einem Destruktor heraus aktivieren.

**4.**
Schreiben Sie eine Klasse mit drei überladenen virtuellen Funktionen. Leiten Sie von der genannten Klasse eine weitere Klasse ab, in welcher Sie eine der überladenen Funktionen überschreiben. Instantiieren Sie dann ein Objekt der abgeleiteten Klasse. Können jetzt die Basisklassen-Funktionen noch aufgerufen werden?
Nehmen Sie ein Upcast der Adresse des Objekts vor. Können die Basisklassen-Funktionen aufgerufen werden.
Stellen Sie die Situation in einem UML-Klassen-Diagramm dar.

**5.**
Schreiben Sie eine abstrakte Basisklasse, die private Pointer auf Objekte als Daten-Member aufweist. Leiten Sie davon eine konkrete Klasse ab, die einen weiteren Pointer auf ein Objekt als Daten-Member hat. Stellen Sie nun Konstruktoren, Copy-Konstruktoren, Zuweisungs-Operatoren und Destruktoren bereit, so dass Klone erzeugt werden können.
Zeigen Sie in **`main`**, dass die Klasse korrekt funktioniert.

**6.**
Schreiben Sie eine Klasse, deren Objekte selbst Buch darüber führen, wie oft sie referenziert werden. Erstellen Sie ein solches Objekt auf dem Heap und referenzieren sie es mehrfach unter Verwendung des Copy-Konstruktors (der hier also nur *shallow copies* erzeugt). Verbieten Sie den Gebrauch des Zuweisungsoperators. Heben Sie die Referenz auf "Kopien" in geeigneter Weise auf. Beim Aufheben der letzten Referenz auf das Objekt werde dieses vom Heap entfernt. Testen Sie das Konstrukt.