Aufgabe 1 Schreiben Sie eine Schnittstellenbeschreibung in Form einer Interface-Klasse namens **Unerledigtes**, die die gemeinsamen Merkmale von **LIFO** und **FIFO** aus der Aufgabenstellung von Practical 2 wiedergibt. Schreiben Sie Klassen **MeinLIFO** und **MeinFIFO**, die exakt dieselbe Funktionalität haben wie die auf der WWW-Seite vorgegebenen Klassen **LIFO** und **FIFO**, und deren Objekte zusätzlich den Typ **Unerledigtes** annehmen können. Testen Sie dies aus. Wieviel muß man an einer Lösung von Practical 2 ändern, um dem Benutzer interaktiv die Wahl zwischen LIFO und FIFO zu geben?

Aufgabe 2 Erweitern Sie die Schnittstellenbeschreibung von Aufgabe 1 um eine Methode **int getAnzahl()**. Verwenden Sie dabei die Klasse **Unerledigtes**, nicht aber deren Code! Erweitern Sie nun die Klasse **LIFO** um einen Zähler für die auf dem Stapel liegenden Elemente so, daß die Methode **int getAnzahl()** immer die gegenwärtige Anzahl der Elemente ausgibt. Dabei dürfen Sie wiederum nicht den Quellcode von **LIFO** benutzen, müssen aber die Klasse **LIFO.class** bei der Programmausführung verwenden.

Aufgabe 3 Erzeugen Sie mittels Vererbung aus Ihrer Lösung für Aufgabe 2 eine Implementierung von **Unerledigtes** (deren Erweiterung!) so, daß **empty()** nun einfach testet, ob **getAnzahl()** den Wert 0 liefert. Weisen Sie mit einem einfachen Programm nach, daß auch die Aufrufe von **empty()** durch andere Methoden in der Klasse **LIFO** immer diese neue Implementierung benutzen. Der Effekt heißt “late binding”, d. h., daß zur Laufzeit des Programms für das jeweilige Objekt festgelegt wird, auf welcher Ebene der Klassenhierarchie die Methode aufgerufen wird.