VL03, Aufgabe 1 (Übung)

In dieser Aufgabe betrachten wir ausschließlich die Klasse `Link` aus dem Beispielprogramm zur Vorlesung (`BSP03-ListeMitVerkettung.zip`):

```
public class Link<T>
{
    public T daten;
    public Link<T> naechster;

    public Link(T daten, Link<T> naechster)
    {
        assert(daten != null);

        this.daten = daten;
        this.naechster = naechster;
    }
}
```

Erstellen Sie ein Java-Programm mit `main`-Methode, das nacheinander folgende Anweisungen ausführt. Zeichnen Sie nach jedem Schritt ein Objekt-Diagramm:

1. Erstellen Sie ein neues `Link`-Objekt, das `mittleresElement` heißt, und den `String` `"Test"` als Nutzdaten enthält.
2. Geben Sie `mittleresElement` einen Nachfolger mit den Daten `"Letzter"`.
3. Erzeugen Sie einen Vorgänger von `mittleresElement`, der `anfang` heißt und die Nutzdaten `"Erster"` enthält.

VL03, Aufgabe 2 (Übung)

Gegeben sei das folgende kurze Java-Programm, welches die Klassen und Methoden des Beispielprogramms zur Vorlesung nutzt (siehe Beispielprogramm zur Vorlesung, `BSP03-ListeMitVerkettung.zip`).

Zeichnen Sie nach jedem Methodenaufruf ein Objektdiagramm der verketteten Liste (entsprechend der Vorlesung).

```
public static void main(String[] args)
{
    Liste<String> eineListe = new Liste<String>();

    eineListe.einfuegen("Pascal");
    eineListe.anfuegen("Java");
    String s = eineListe.entfernen();
}
```

VL03, Aufgabe 3 (Übung)

a) Erläutern Sie anhand eines Diagramms, das die Objektzustände darstellt, den Ablauf bei der Verkettung von zwei Listen `A` und `B` zu einer Liste.

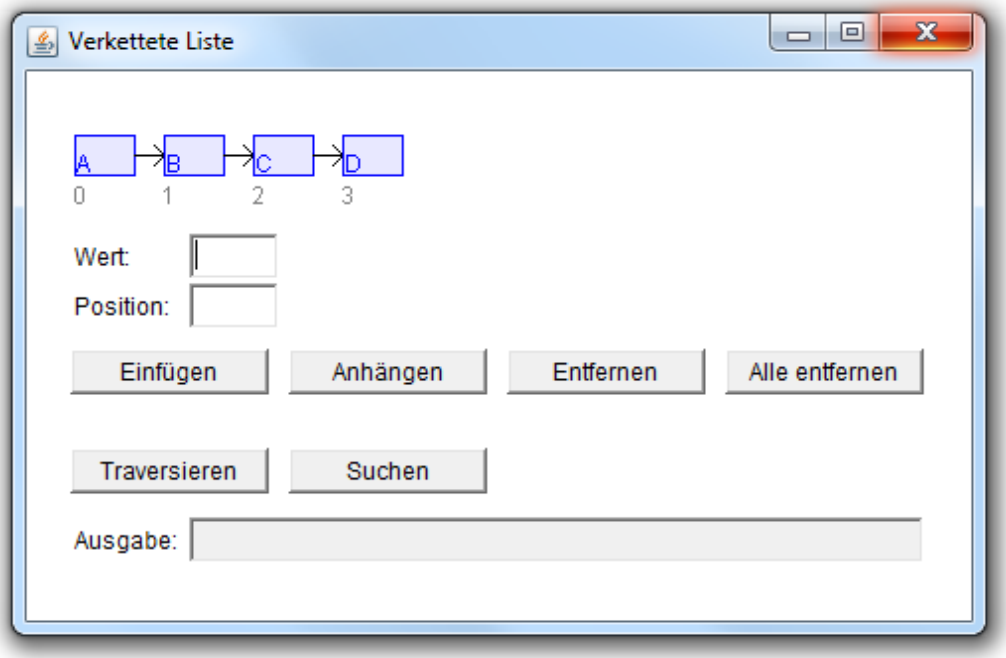
b) Warum ist das Verketten von zwei Listen eine potenziell gefährliche Operation? Was kann man tun, um das entstehende Problem zu beheben?

VL03, Aufgabe 4 (Praktikum)

Benutzen Sie für diese Aufgabe das Beispielprogramm zur Vorlesung aus dem ZIP-Archiv `BSP03-ListeMitVerkettung.zip`. **Ohne Änderungen kann das Programm zunächst nicht compiliert werden!**

a) Implementieren Sie nun in der Klasse `Liste` die Methode `istLeer()`, die `true` zurückgibt, wenn die Liste leer ist. Diese Methode wird auch von den vorgegebenen Methoden verwendet und muss daher korrekt implementiert werden! Sie können das Programm nun compilieren und starten.

b) Ergänzen Sie in der Klasse `Liste` die Methode `verketten`, die zwei Listen wie in Aufgabe 3 verketten soll. Sie können davon ausgehen, dass beide Listen mindestens ein Element enthalten. Wenn die Verkettung korrekt arbeitet, muss beim Start des Programms eine Liste mit den Werten `A` bis `D` zu sehen sein:



Die verschiedenen Operationen (Buttons) greifen auf die Eingabefelder „Wert“ und „Position“ zurück:

	Einfügen	Anhängen	Entfernen	Alle entf.	Travers.	Suchen
Wert	✓	✓		✓		✓
Position	✓		✓			

VL03-Aufgabe 5 (Praktikum)

Ergänzen Sie in der Klasse `Liste` die Methode `entferneWerte(final T opfer)`, welche die Liste durchläuft und alle Nutzelemente aus der Liste löscht, die `opfer` als Daten enthalten. Die Methode gibt die Anzahl der gelöschten Elemente zurück.

Der zu löschende Wert wird im Textfeld „Wert“ eingegeben. Die vorgegebenen Methoden `einfuegen`, `anfuegen`, `suchen` und `entfernen` dürfen von Ihnen nicht aufgerufen werden!

Sie können Ihre Methode testen, indem Sie auf den Button „Alle entfernen“ klicken. Die Anzahl der gelöschten Elemente wird im Textfeld angezeigt. Prüfen Sie auch Sonderfälle, beispielsweise das Löschen des ersten und letzten Elements mit anschließendem Einfügen bzw. Anfügen neuer Elemente.