# Algorithmen & Datenstrukturen 2

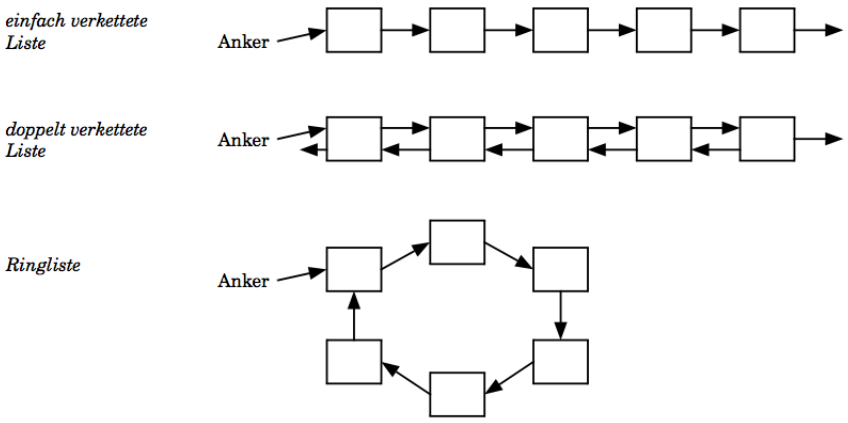
Jan Fässler

3. Semester (HS 2012)

# Inhaltsverzeichnis

# 1 Listen

Eine verkettete Liste (linked list) ist eine dynamische Datenstruktur zur Speicherung von Objekten. Sie eignen sich für das speichern einer unbekannten Anzahl von Objekten, sofern kein direkter Zugriff auf die einzelnen Objekte benötigt wird. Jedes Element in einer Liste muss neben den Nutzinformationen auch die notwendigen Referenzen zur Verkettung enthalten.
Es gibt drei verschiedene Arten von Listen:




Listing 1: Einfache verkettete Liste

```
1  public class LinkedList<T> {
2    private Element<T> head = null;
3    private Element<T> last = null;
4    public void add(T data) {
5      last.next = new Element<T>(data);
6      last = last.next;
7    }
8    public void remove(T data) {
9      Element<T> current = head;
10     while (current != null && current.data != data) {
11       current = current.next;
12       if (current != null && current.data == data) {
13         current.last = current.next;
14         current = null;
15       }
16     }
17   }
18   public T getFirst() {
19     return head.data;
20   }
21   public T getLast() {
22     return last.data;
23   }
24   public class Element<E> {
25     public Element<E> next;
26     public Element<E> last;
27     public E data;
28     public Element(E input) {
29       data = input;
30     }
31   }
32 }
```

## 1.1 Stack

Der Stack ist eine dynamische Datanstruktur bei der man nur auf das oberste Element des Stabels zugreifen (top), ein neues Element auf den Stabel legen (push) oder das oberste Element des Stapels entfernen (pop) kann.

Listing 2: Implementierung eines Stacks

```
1  public class Stack<T> extends LinkedList<T> {
2    public T top() {
3      return this.getLast();
4    }
5    public void push(T data) {
6      this.add(data);
7    }
8    public T pop() {
9      T last = this.getLast();
10     this.remove(last);
11     return last;
12   }
13   public boolean isEmpty() {
14     return this.getFirst() == null ? true : false;
15   }
16 }
```