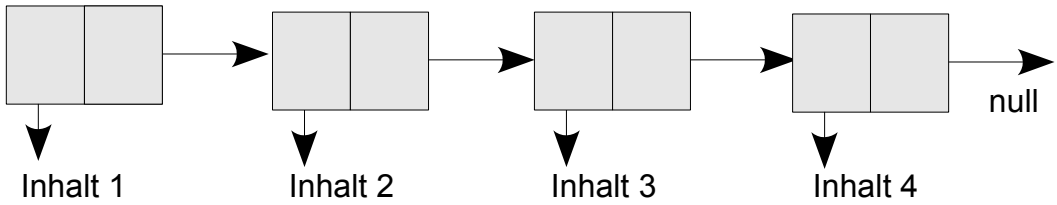
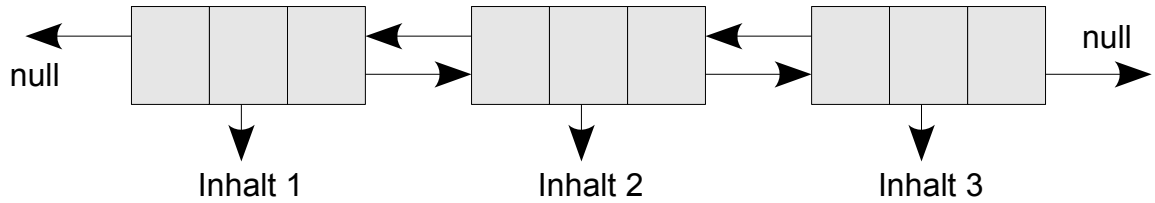
Verkettete Liste

Im Text zu den Sammlungsklassen ist das "Standardbild" zu verketteten Listen angegeben ...



… und nur aus der Zusatzaufgabe geht hervor, dass die **`LinkedList`** eigentlich ganz anders darzustellen ist, nämlich folgendermaßen:



Eine einfach verkettete Liste

Im jetzt folgenden Text soll es nun aber im Zusammenhang mit der selbst geschriebenen Klasse Warteschlange mit Iterator um die erste oben angegebene Struktur einer einfach verketteten Liste gehen, weil wir diese bei der Warteschlange als interne Datenstruktur verwendet haben. Sie ist nämlich äußerst einfach anzugeben und zu behandeln.

Definition eines Element – Objektes

Für die Definition einfacher Element – Objekte schreiben wir die minimale Form einer Klasse, die wir als interne Klasse realisieren und es daher auch zulassen, dass alle Regeln von Kapselung dabei verletzt werden.

```
/**
 * Minimalloesung der internen Klasse Element
 */
private class Element {
    Object inhalt=null;
    Element naechstes=null;
}
```

Die Klasse hat also keine Methoden und einfach nur zwei Attribute, die für die beiden Felder in der oben gewählten Darstellung stehen.
Wenn wir uns mit …

```
Element einElement = new Element();
```

… ein neues Elementobjekt verschaffen, ist also nichts weiter gewährleistet, als dass die beiden "Zeiger" für den Inhalt und für das nächste Elementobjekt zur Zeit noch "auf null zeigen".
"Per Hand" könnte man die oben angegebene verkettete Liste daher mit folgenden Programmzeilen erzeugen:

```
einElement = new Element();
einElement.inhalt = "Inhalt 1";
einElement.naechstes = new Element();
```

Jetzt bekommen wir ein Problem, wenn wir dem zweiten Element den Inhalt und das weitere nachfolgende Element anhängen wollen. Also müssen wir uns durch die Liste zum zweiten Element durchhangeln. Da wir dabei die Variablennamen verändern, müssen wir uns in einer neuen Variablen dringend den Kopf der Liste merken, anderenfalls sind die

Daten weg.

```
Element kopf = einElement;
einElement = new Element();
kopf.naechstes = einElement;
einElement.inhalt = "Inhalt 2";
```

Für das dritte Element wird es noch aufwändiger. Den Kopf brauchen wir nicht neu zu definieren, durchhangeln müssen wir uns aber weiter, wenn wir nicht vorher clever waren und uns das Ende der Warteschlange gemerkt haben. Machen wir das also noch. Die Zeile ...

```
Element schwanz = einElement;
```

… schließt dann also das Anhängen des zweiten Elementes ab und nun bekommen wir das dritte ganz einfach angehängt:

```
einElement = new Element();
einElement.inhalt = "Inhalt 3";
schwanz.naechstes = einElement;
```

Nun ist das neue Element einfach angehängt, es muss aber noch als der neue Schwanz ausgewiesen werden:

```
schwanz = einElement;
```

Beim vierten Element geht es nun genauso leicht.

Aufgabe

Zeichnen Sie einzelne Karten mit dem Elementsymbol und legen Sie diese entsprechend dem beschriebenen Vorgang aneinander und tragen Sie die Werte ein.