12-2

(e)

clone: Einfaches Umhängen führt zur neuen Liste. Schnelle Operation. Allerdings wenn nachträgliche Änderungen an der angehängten Liste erfolgen, kann die ungewollte Effekte haben.
concatClone: erzeugt neue Liste aus Kopien anderer. Änderungen an den Ursprungslisten haben keinen Einfluss auf neue Liste. Dafür ist es langsamer als clone, weil man um die Kopien zu erzeugen durch die Listen iterieren muss.

(h)

ja, terminiert. Nur das letzte Listenelement hat als Nachfolger = null. Da durch alle Listenelemente iteriert wird, endet die Schleife nach endlichen Schritten. Wenn die Liste leer ist, wird Schleife gar nicht betreten.

(i)

Listen haben eine variable Länge, können ohne Probleme erweitert werden, indem man Einträge anhängt - Arrays nicht. Bei Arrays kann direkt auf ein Element zugegriffen werden. Bei Listenelementen muss man zunächst Vorgänger (bzw. Folgeelemente) durchlaufen, um zu einem Element zu gelangen.

12-3

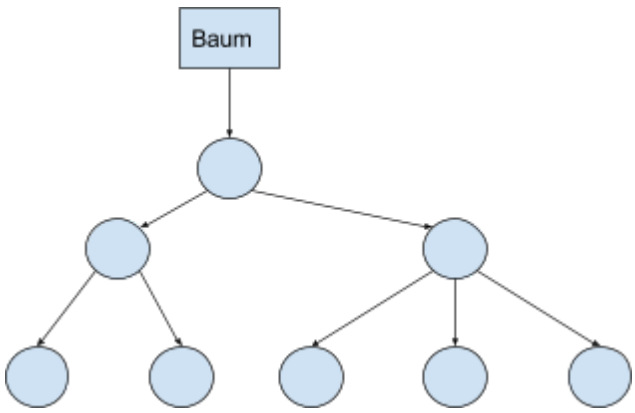
(a)

```
public class List<T> {                  // unverändert
        public Entry <T> head;
}

public class Entry<T> {                 // statt Nachfolger werden 2 Kinder angegeben
        public T element;
        public Entry <T> leftChild;
        public Entry <T> rightChild;
}
```

(b)

Unter variablem Knotengrad verstehen wir z.B. folgendes Konstrukt:



Um einen binären Baum mit variablem Knotengrad zu realisieren, kann das Attribut Entry<T> next in der Klasse Entry zu einer Liste List<T> next abgewandelt werden, wobei die Elemente der Listenentries die Referenzen auf die Unterknoten enthalten.
Um den gewünschten Knoten aus der Liste auszuwählen kann z.B die Methode getNext() zu getNext(int Index) abgeändert werden.