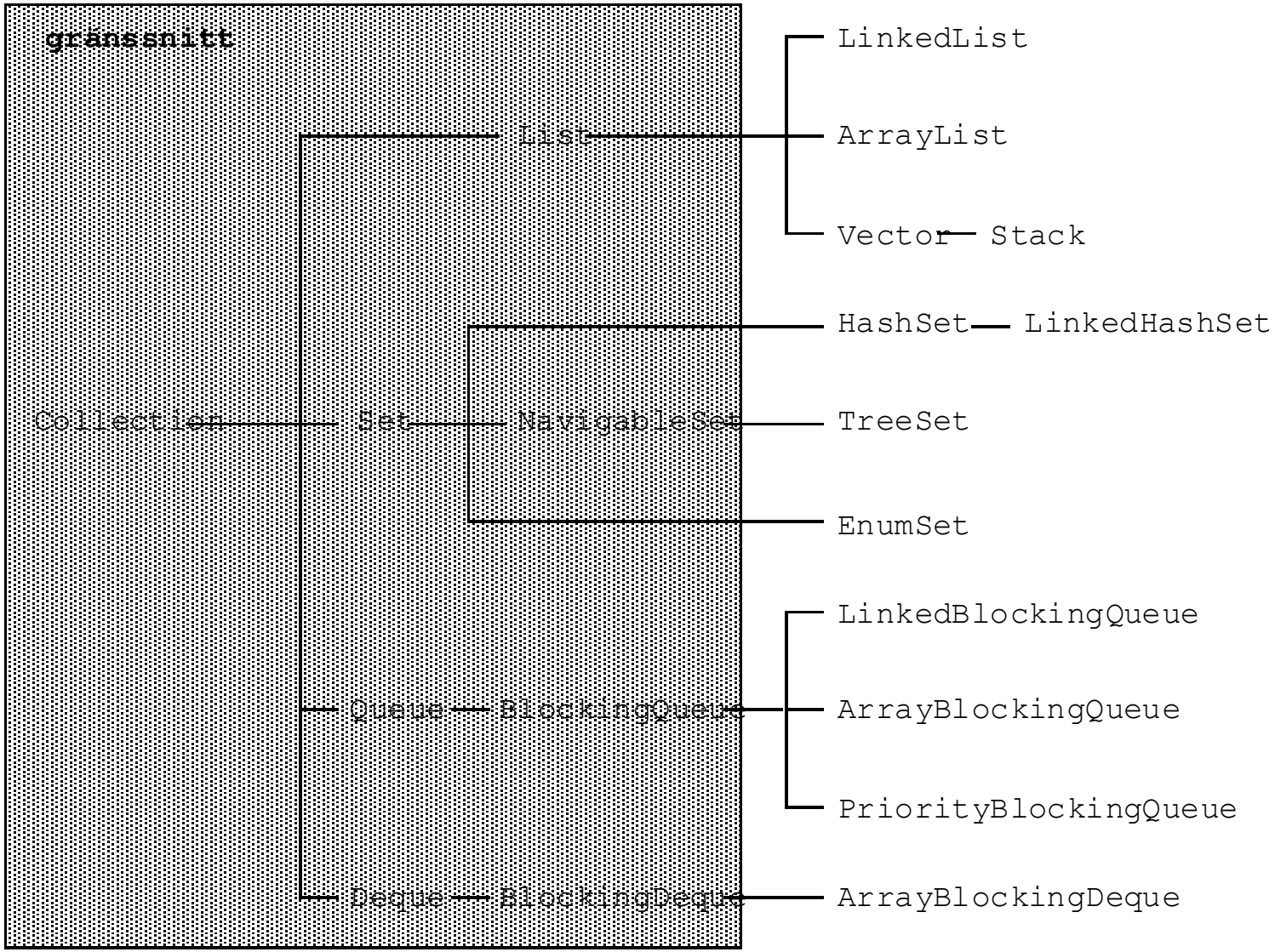

gränssnitt
Collection
List
Set
Queue
Deque
SortedSet
NavigableSet
BlockingQueue
BlockingDeque
LinkedList
ArrayList
Vector
Stack
HashSet
LinkedHashSet
TreeSet
EnumSet
LinkedBlockingQueue
ArrayBlockingQueue
PriorityBlockingQueue
ArrayBlockingDeque

Skapa samlingar

```
Set<typ> h = new HashSet<>();
... // placera element i mängden h
List<typ> l = new LinkedList<>(h); // kopia av h

l.add(new String("en text"));
```

Klassen Collections

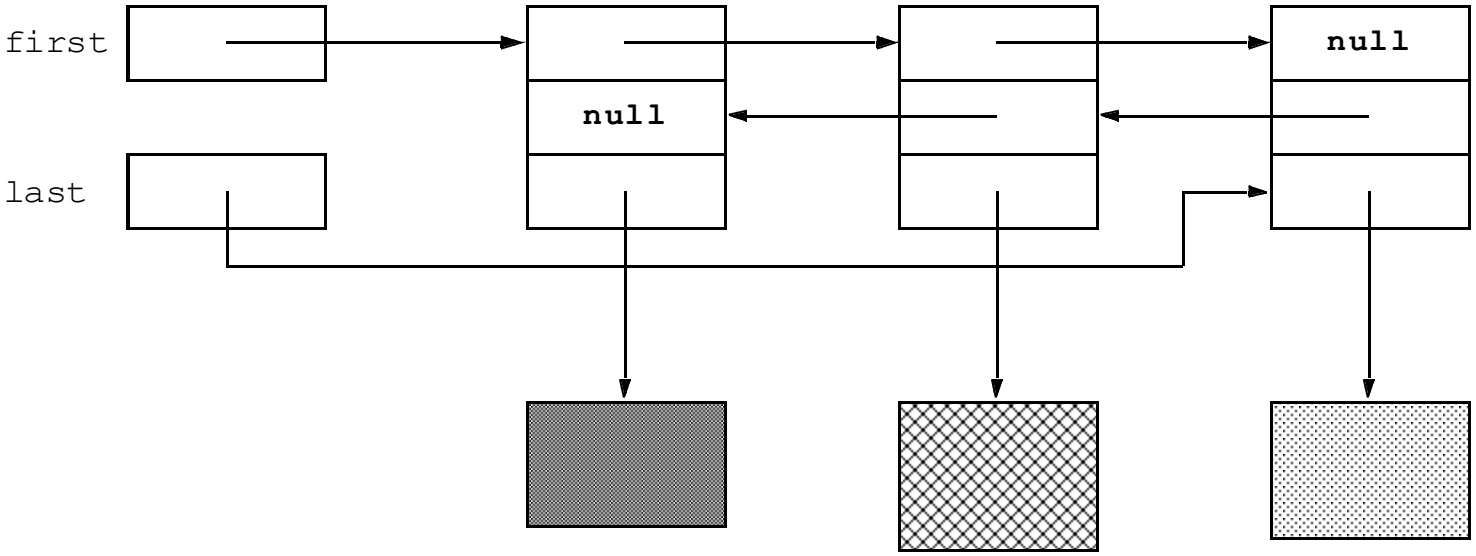
```
Integer imax = Collections.max(li);
Collections.fill(l, "Java");   // lägger "Java" i alla element i l

List<Integer> nollor = Collections.nCopies(100, 0);
Collections.copy(lTill, lFrån);

Collections.sort(l);
int k = Collections.binarySearch(l1, sökt);

Jämförare jfr = new Jämförare();
Collections.sort(l, jfr);
int k = Collections.binarySearch(l, sökt, jfr);
```

Implementering av listor



Mängder

```
Set<String> s0 = Collections.emptySet();
Set<String> s1 = Collections.singleton(new String("Ensam"));


TreeSet<E>()
TreeSet<E>(Comparator<? super E> comp)
TreeSet<E>(Collection<? extends E> coll)
TreeSet<E>(SortedSet<E> ss)
```

```
import java.util.*;
import java.text.*;
import java.io.*;

public class TextAnalys {
  public static void main(String[] arg) throws IOException {
    Collator co = Collator.getInstance();  // jämför texter
    co.setStrength(Collator.PRIMARY);
    NavigableSet<String> m = new TreeSet<String>(co);

    // koppla en scanner till filen
    Scanner sc = new Scanner(new File(arg[0]));

    // läs ett ord i taget och addera till mängden
    while(sc.hasNext())
      m.add(sc.next());

    // skriv ut alla orden
    for (String ord : m)
      System.out.println(ord);
  }
}
```

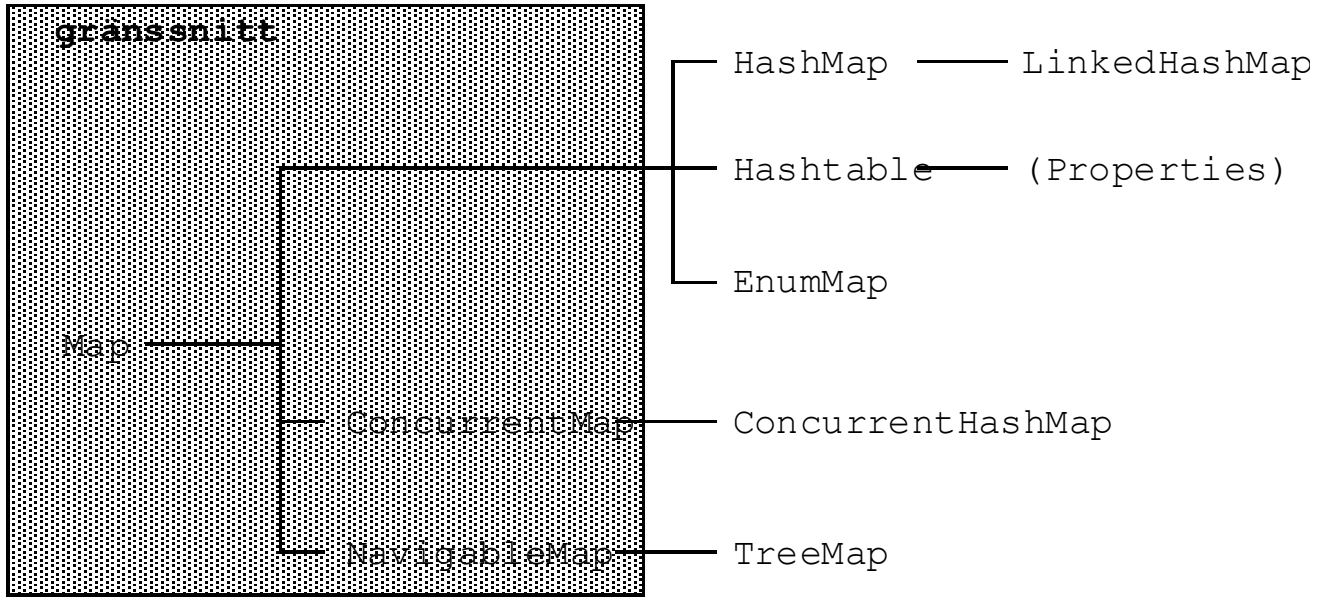
Iteratorer

```
List<Integer> li = new LinkedList<>();
for (Iterator<Integer> it=li.iterator(); it.hasNext(); )
  if (it.next() == 0)
    it.remove();
```

```
for (Integer i : li)
  System.out.println(i);
```

Samma som:

```
for (Iterator<Integer> it=li.iterator(); it.hasNext();) {
  Integer i = it.next();
  System.out.println(i);
}
```


gränssnitt
Map
SortedMap
ConcurrentMap
NavigableMap
HashMap
LinkedHashMap
Hashtable
(Properties)
EnumMap
ConcurrentHashMap
TreeMap

```
Map<key,value> tab1 = new TreeMap<>();
... // lägg in avbildningar i tab1
Map<key,value> tab2 = new HashMap<>(tab1);
```

```
Map<String,Integer> pertab  = new TreeMap<>();
Map<String,Motorfordon> reg = new HashMap<>();

pertab.put("David", new Integer(18));

Motorfordon f = reg.get("ABC123");
```

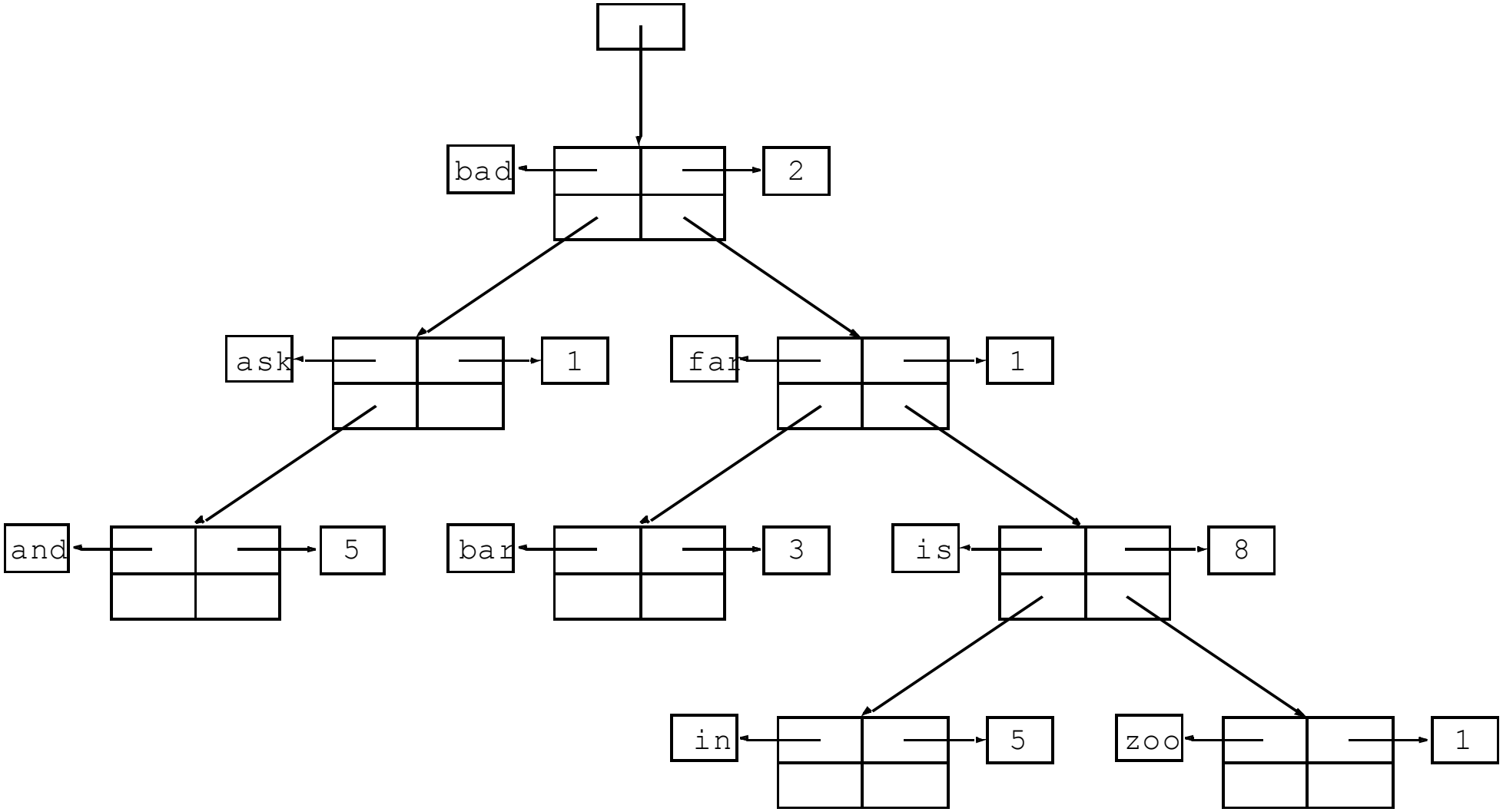
```
TreeMap()
TreeMap(Comparator<? super K> comp)
TreeMap(Map<? extends K,? extends V> m)
TreeMap(SortedMap<K,? extends V> sm)
```

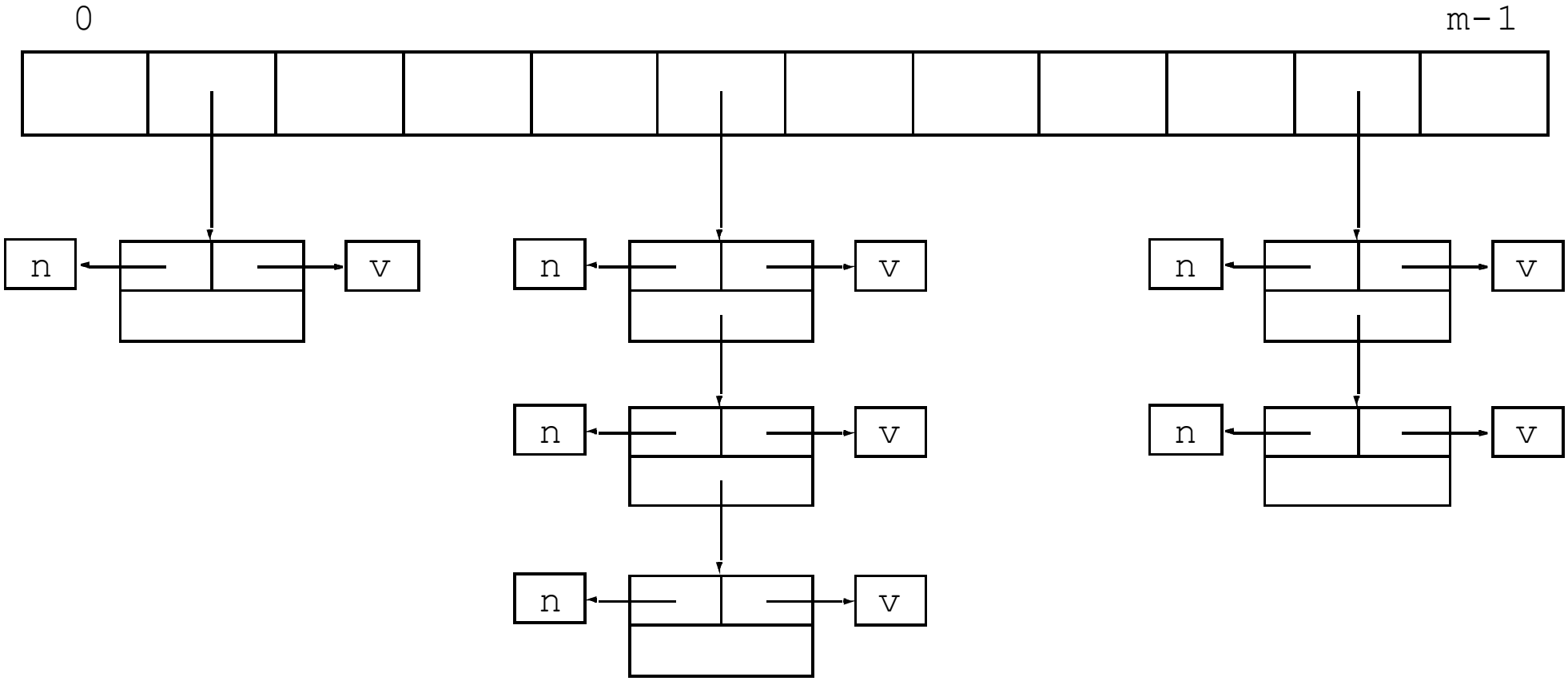
```
import java.util.*;
import java.text.*;
import java.io.*;

public class TextAnalys2  {
  public static void main(String[] arg) throws IOException {
    Collator co = Collator.getInstance();  // jämför texter
    co.setStrength(Collator.PRIMARY);
    // skapa avbildningstabellen
    NavigableMap<String,Integer> tab = new TreeMap<>(co);
    // koppla en scanner till filen
    Scanner sc = new Scanner(new File(arg[0]));
    // läs ett ord i taget och addera till mängden
    while(sc.hasNext()) {
      String ord = sc.next();
      Integer antal = tab.get(ord);  // slå upp ordet i tabellen
      if (antal == null)
        antal = new Integer(0);      // ordet fanns inte tidigare
      // öka antalet förekomster av ordet
      tab.put(ord, new Integer(antal+1));
    }
```

```
    // skapa en sorterad mängd med alla par av ord och antal
    Set<Map.Entry<String,Integer>> m = tab.entrySet();

    // skriv ut alla paren
    for (Map.Entry<String,Integer> a : m)
      System.out.println(a.getKey() + "  " + a.getValue());
  }
}
```


bad
2
ask
1
far
1
and
5
bar
3
is
8
in
5
zoo
1


0
m-1
n
v
n
v
n
v
n
v
n
v
n
v