Quicksort
Et eksempel på rekursjon

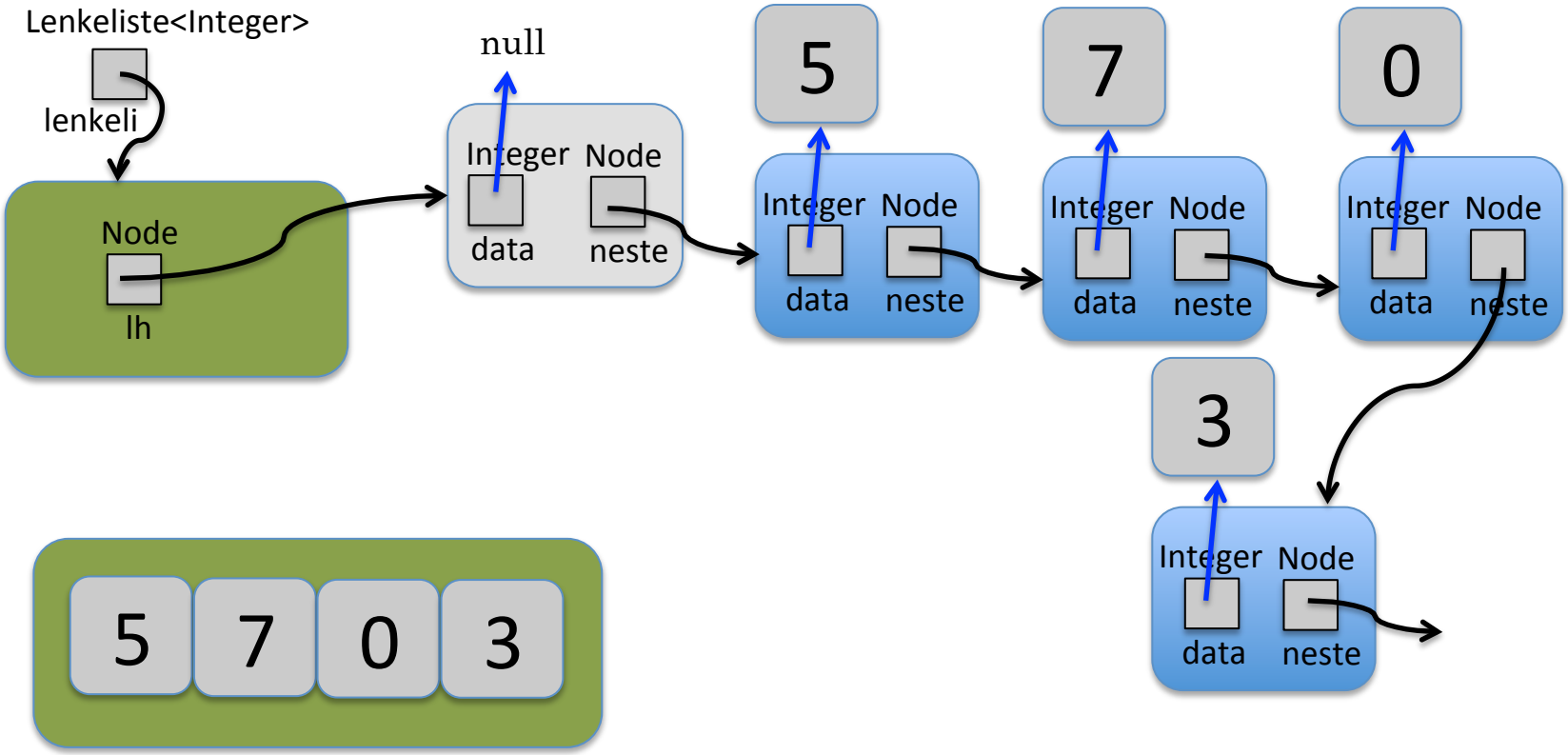
INF1010

Stein Michael Storleer

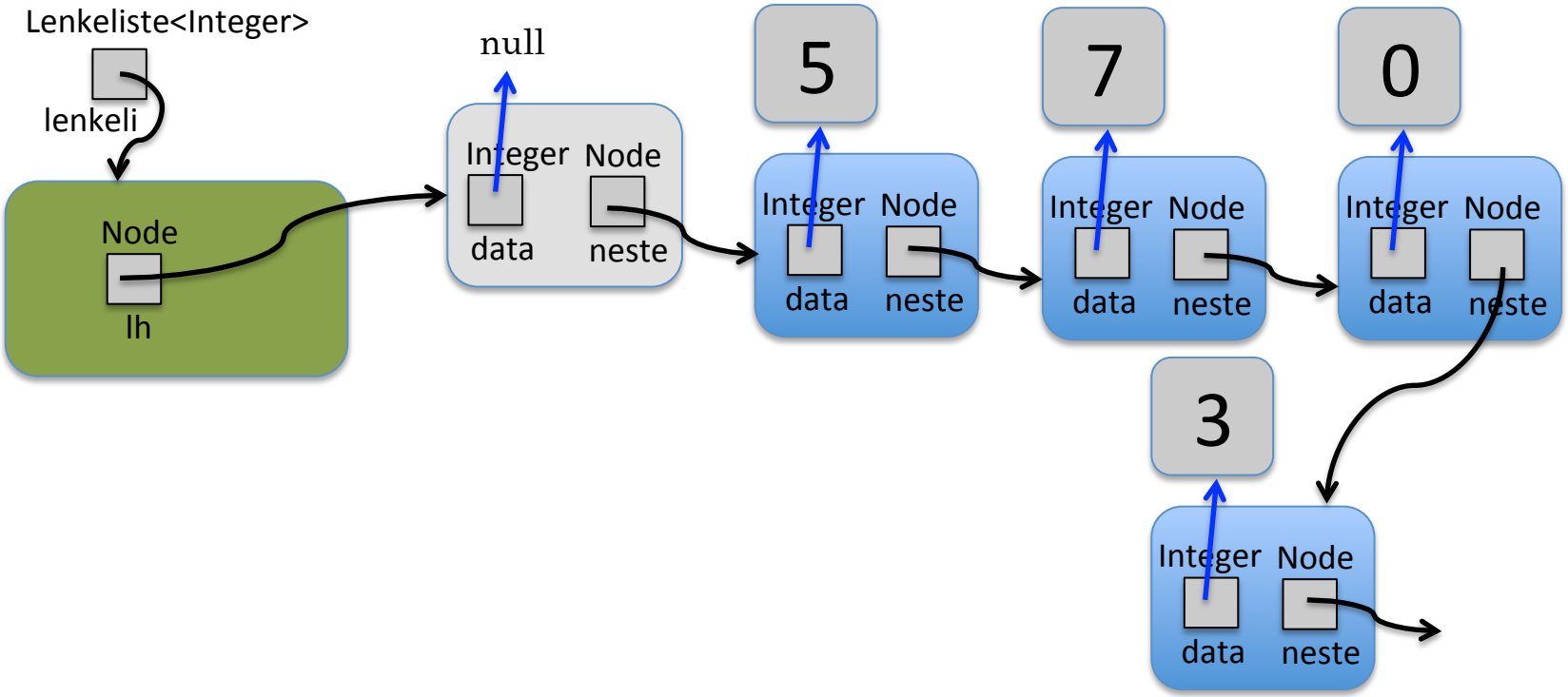

public static void quicksort(char[] items, int left, int right)
{
int i, j;
char x, y;
i = left; j = right;
x = items[(left + right) / 2];
do
{
while ((items[i] < x) && (i < right)) i++;
while ((x < items[j]) && (j > left)) j--;
if (i <= j)
{
y = items[i];
items[i] = items[j];
items[j] = y;
i++; j--;
}
} while (i <= j);
if (left < j) quicksort(items, left, j);
if (i < right) quicksort(items, i, right);
}

```
private static char[] quickSort(char[] items, int left, int right) {
        int L = left;
        int R = right;
        // we pick the middle item as comparand; other methods can be used as well.
        char comparand = items[(L + R) / 2];
        char comparated;

        do {
            while(items[L] < comparand && L < right) { L++; }
            while(comparand < items[R] && R > L) { R--; }

            if(L <= R) {
                comparated = items[L];
                items[L] = items[R];
                items[R] = comparated;
                L++;
                R--;
            }

        } while(L <= R);

        if(left < R) {
            quickSort(items, left, R);
        }
        if(L < right) {
            quickSort(items, L, right);
        }
        return items;
    }
}
```

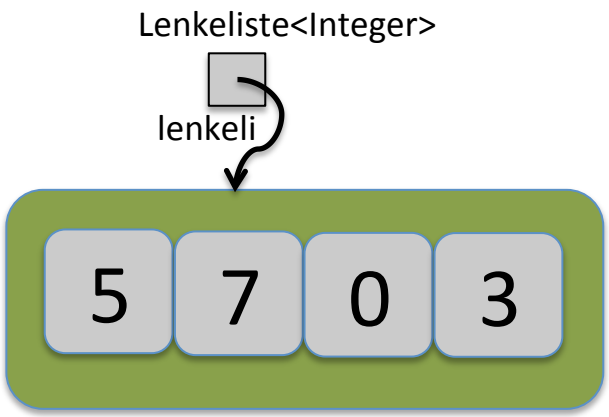
https://en.wikibooks.org/wiki/Algorithm_Implementation/Sorting/Quicksort#Java



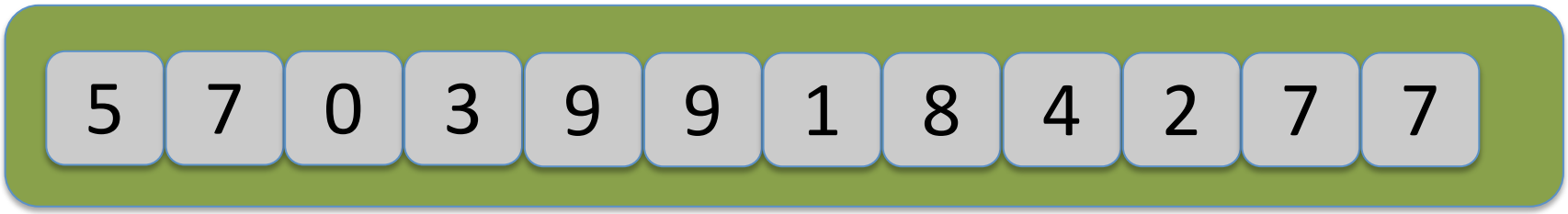

```
class Lenkeliste<T extends Comparable<T> > implements Iterable<T> {

    private Node lh = null ; // listehode
    private class Node { … }
    public void settInnForan(T o) { … }
    public T taUtForan() { … }

    public void quickSort() { … }
}
```




```
class Lenkeliste<T extends Comparable<T> > implements Iterable<T> {

    private Node lh = null ; // listehode
    private class Node { … }
    public void settInnForan(T o) { … }
    public T taUtForan() { … }

    public void quickSort() { … }
}
```

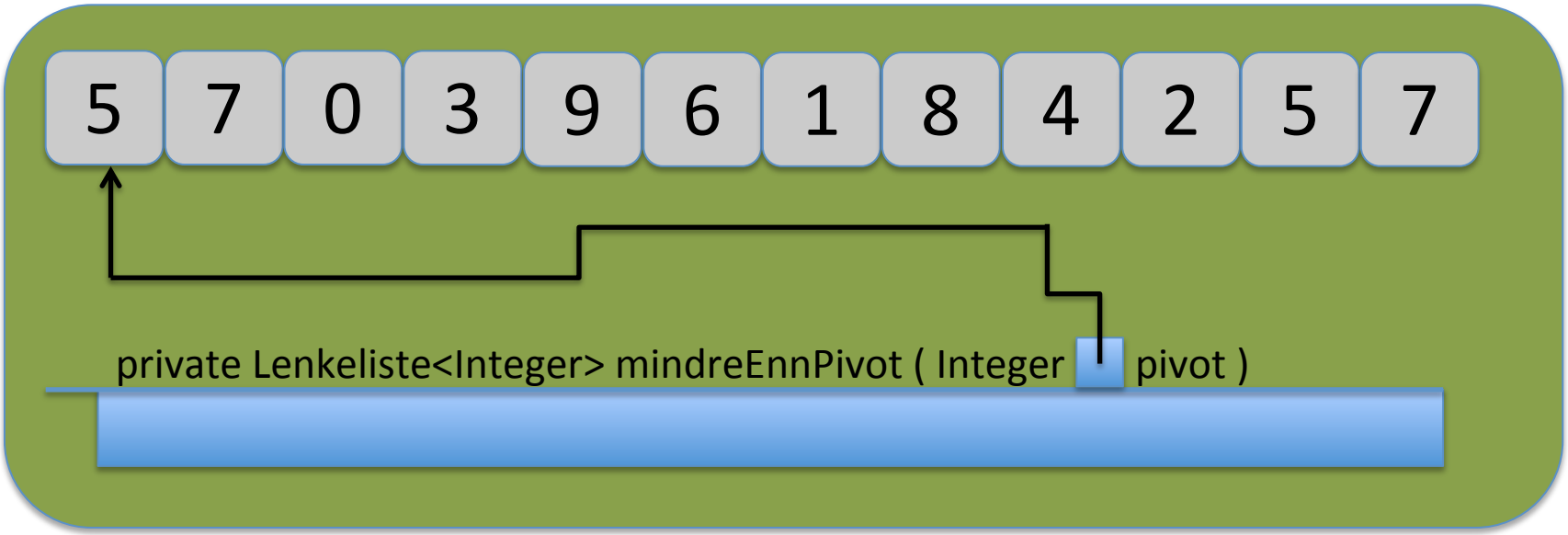


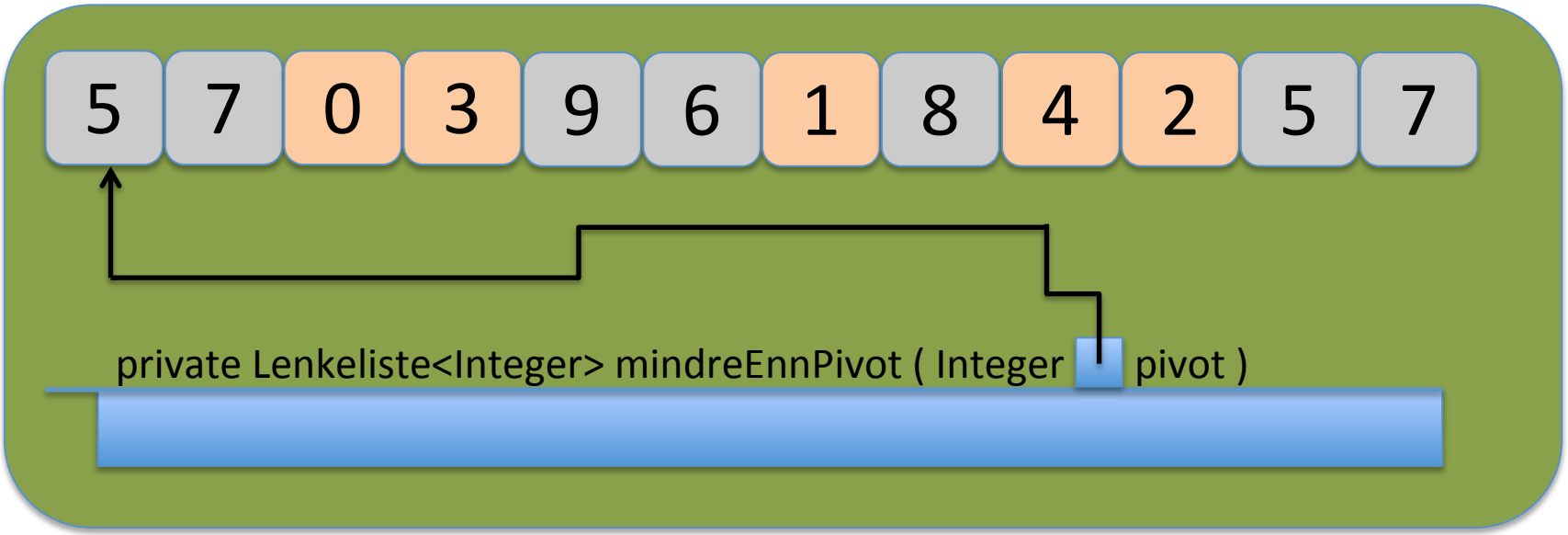

```
class Lenkeliste<T extends Comparable<T> > implements Iterable<T> {

    private Node lh = null ; // listehode
    private class Node { … }
    public void settInnForan(T o) { … }
    public T taUtForan() { … }

    public void quickSort() { … }
}
```

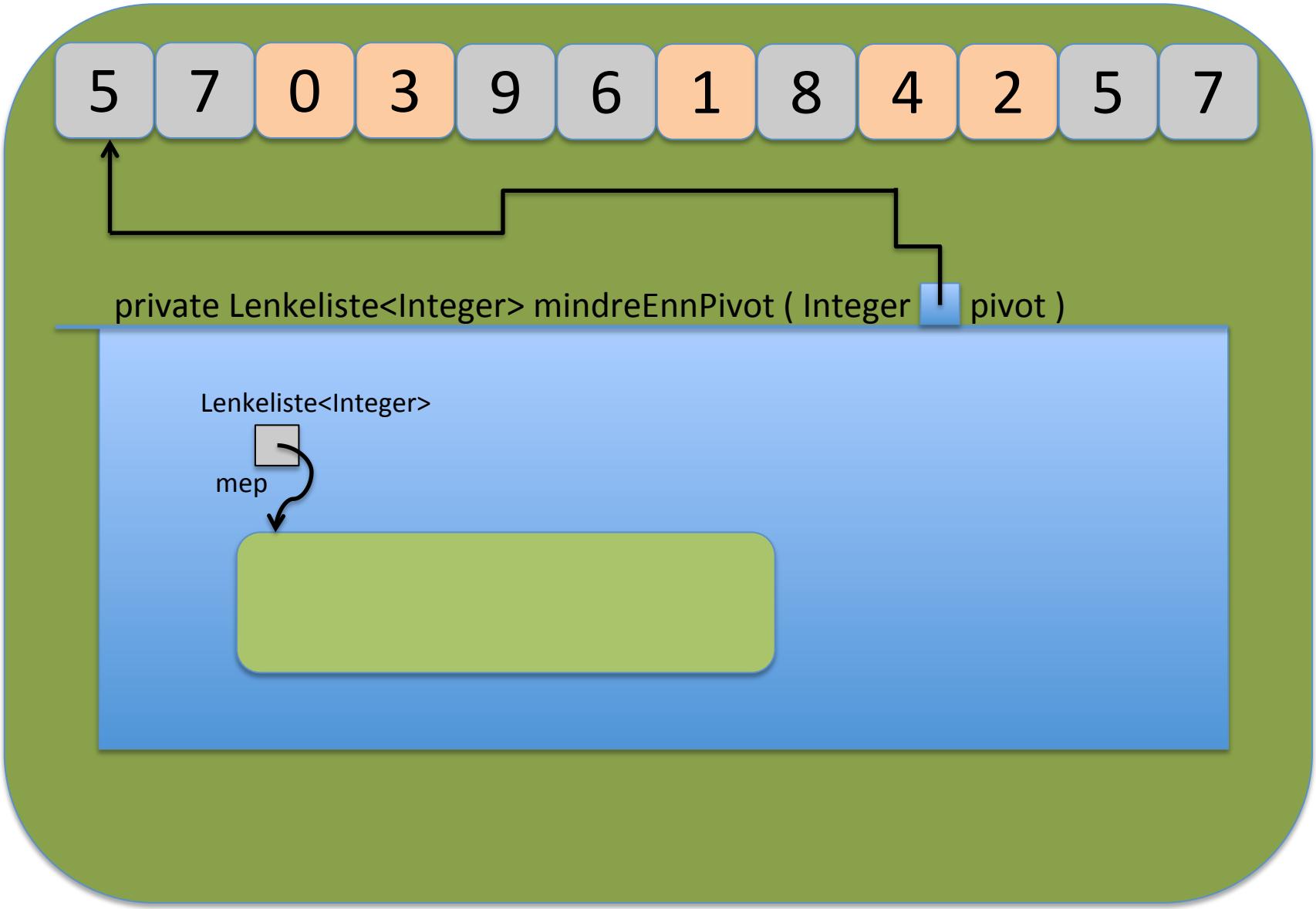

5 7 0 3 9 9 1 8 4 2 7 7

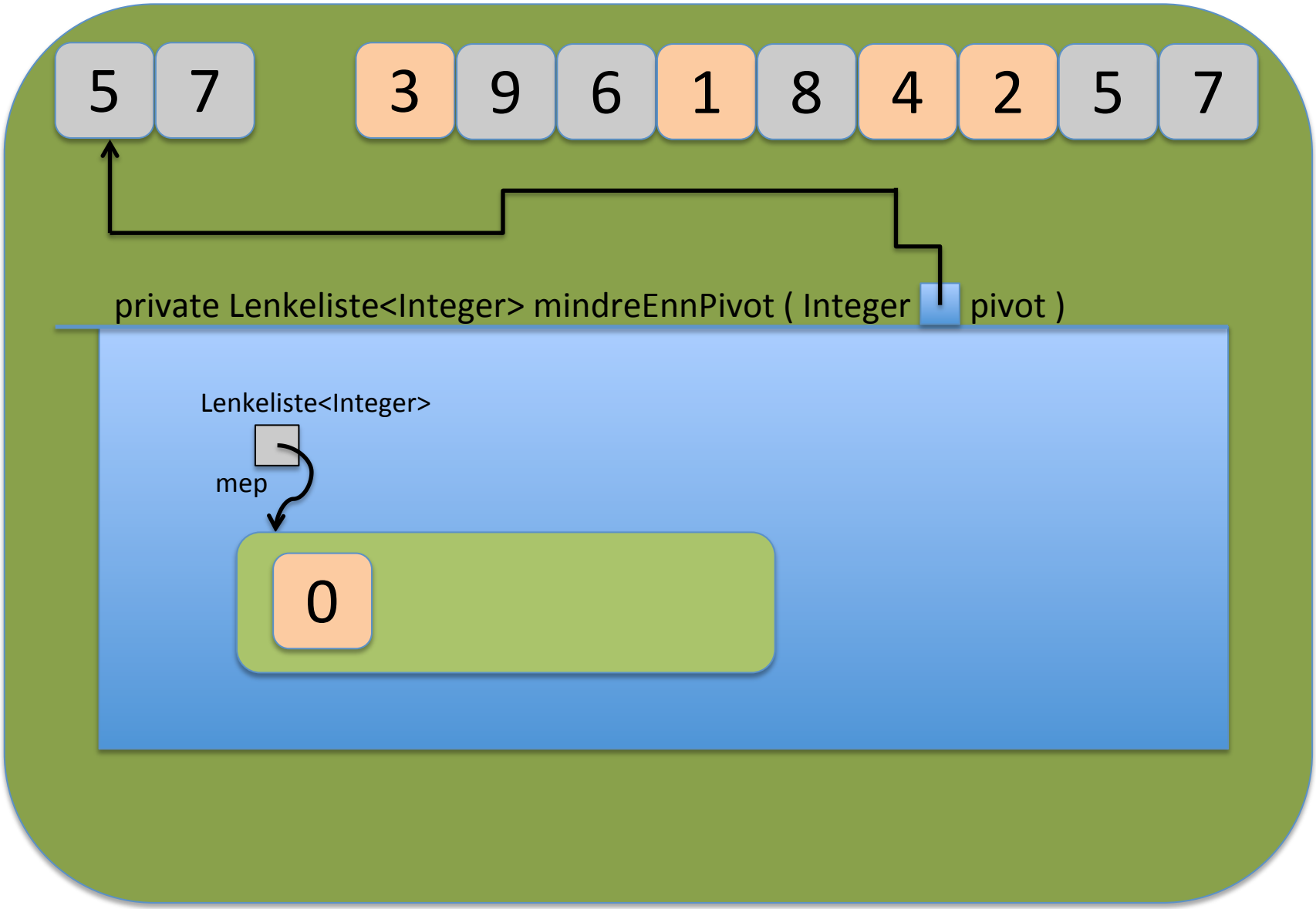

```
public void quickSort() {
    if (! tom() ) {
        T pivot = this.taUtForan();
        Lenkeliste<T> mep = mindreEnnPivot(pivot);
        …
}
```

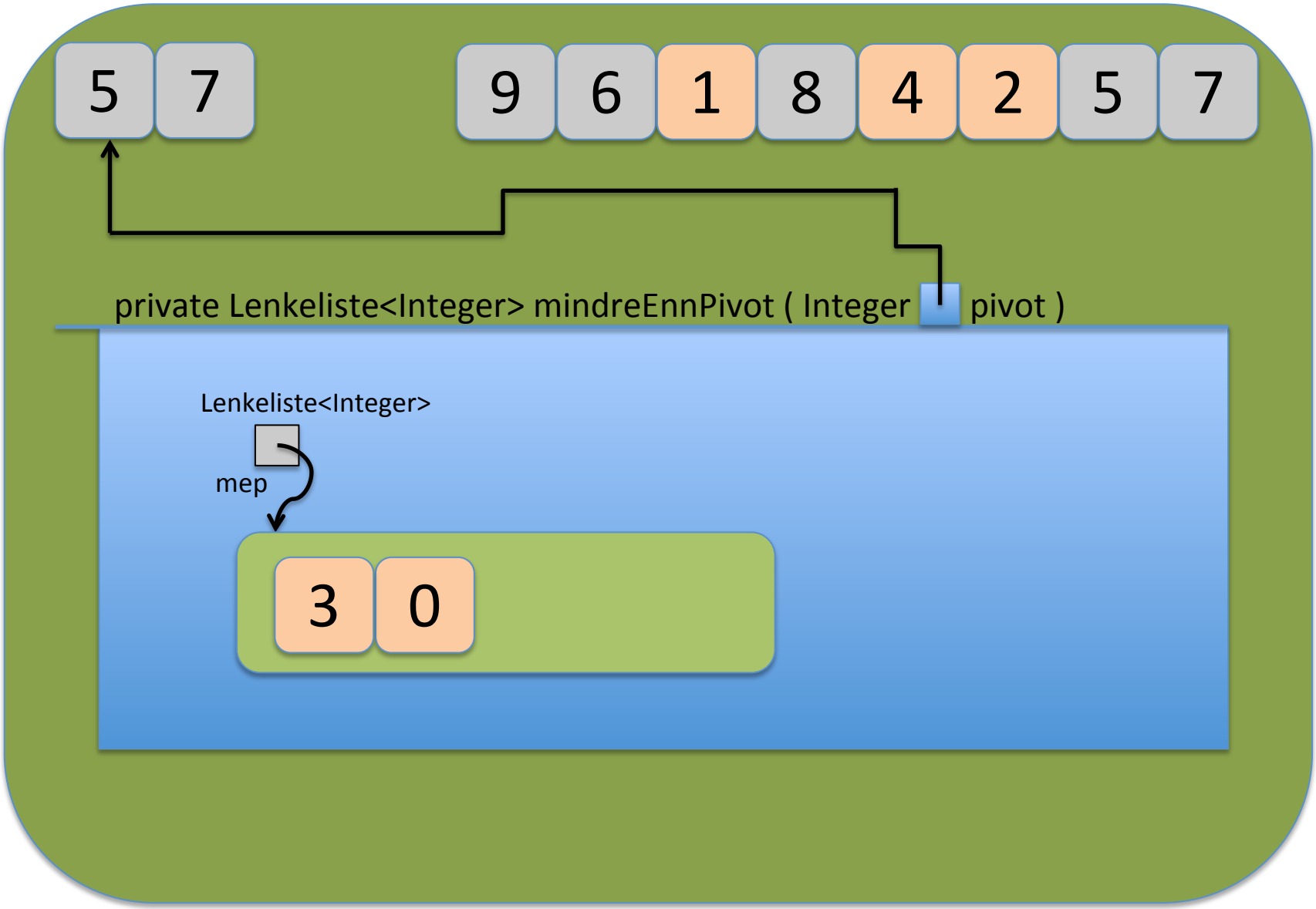
```
private Lenkeliste<Integer> mindreEnnPivot ( Integer  pivot )
```

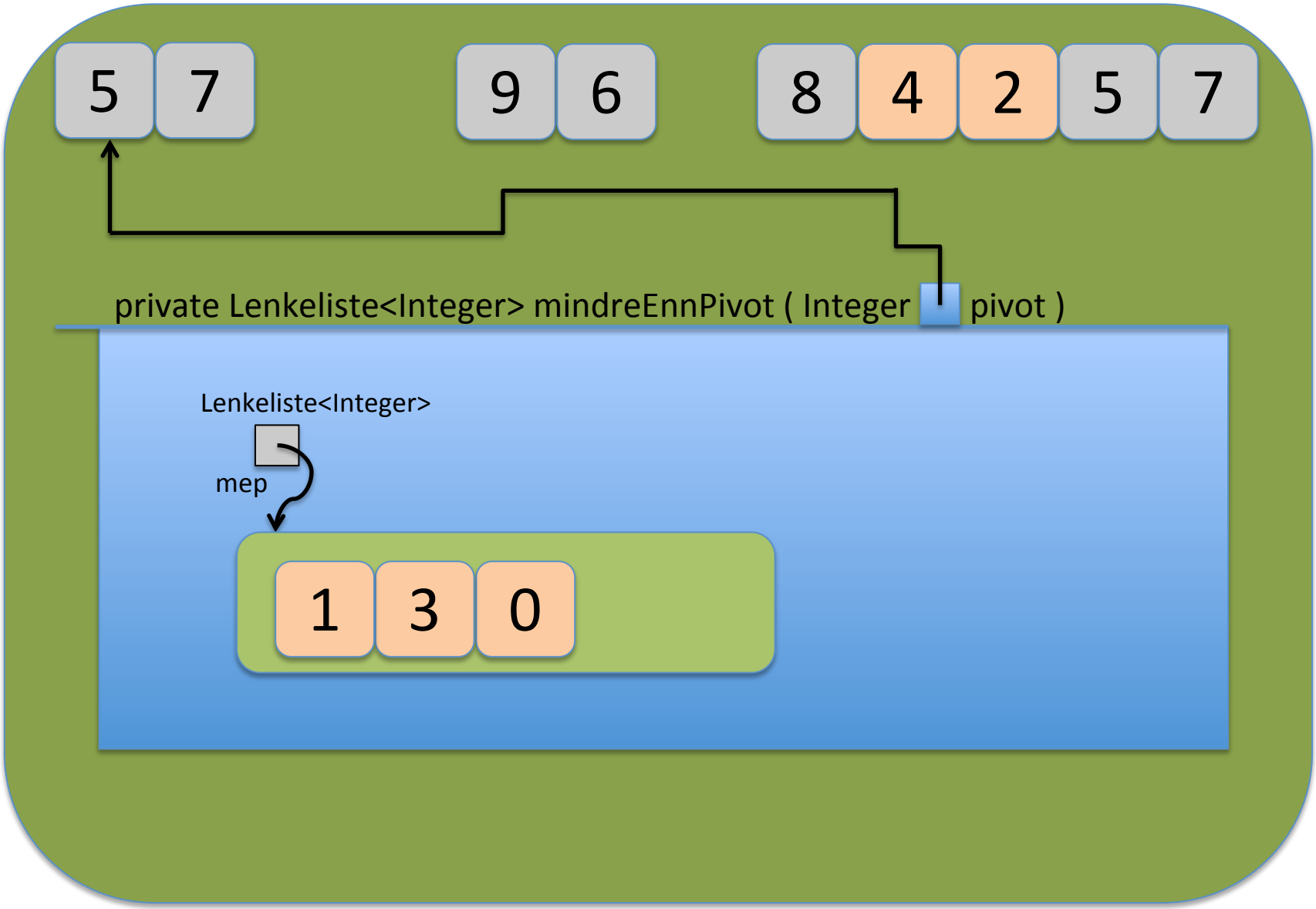

5 7 0 3 9 6 1 8 4 2 5 7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)

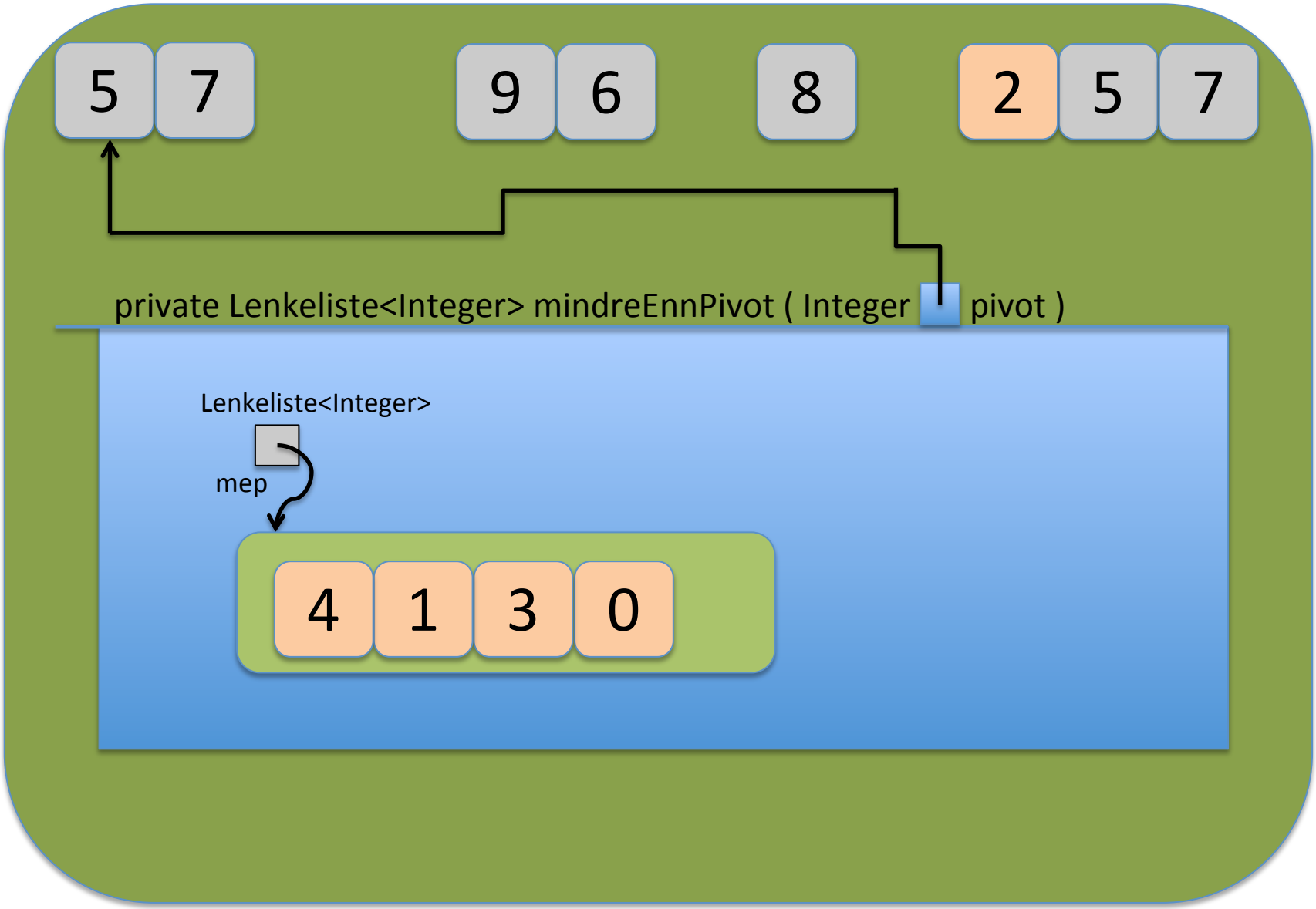

5 7 0 3 9 6 1 8 4 2 5 7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)

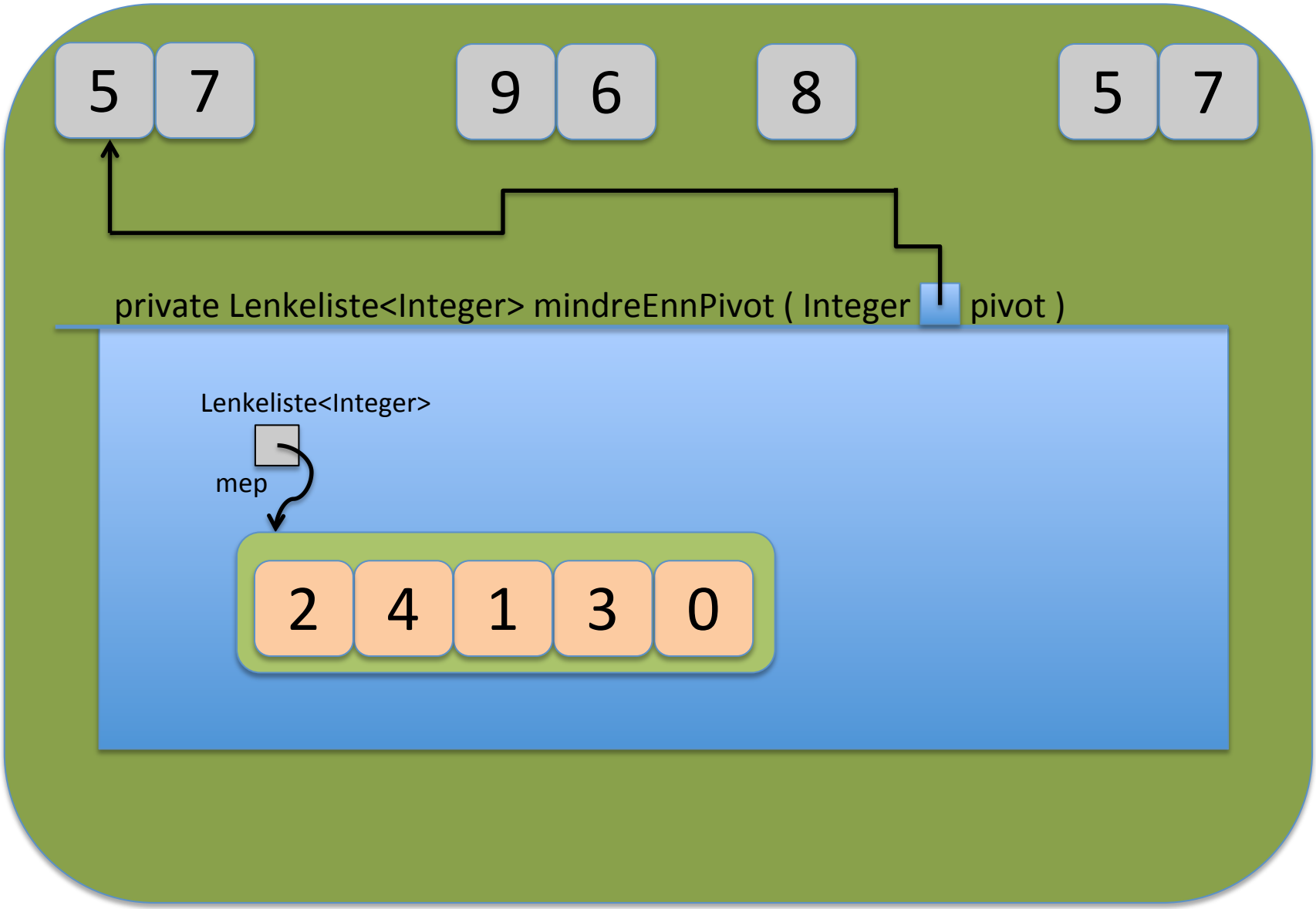

5 7 0 3 9 6 1 8 4 2 5 7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep

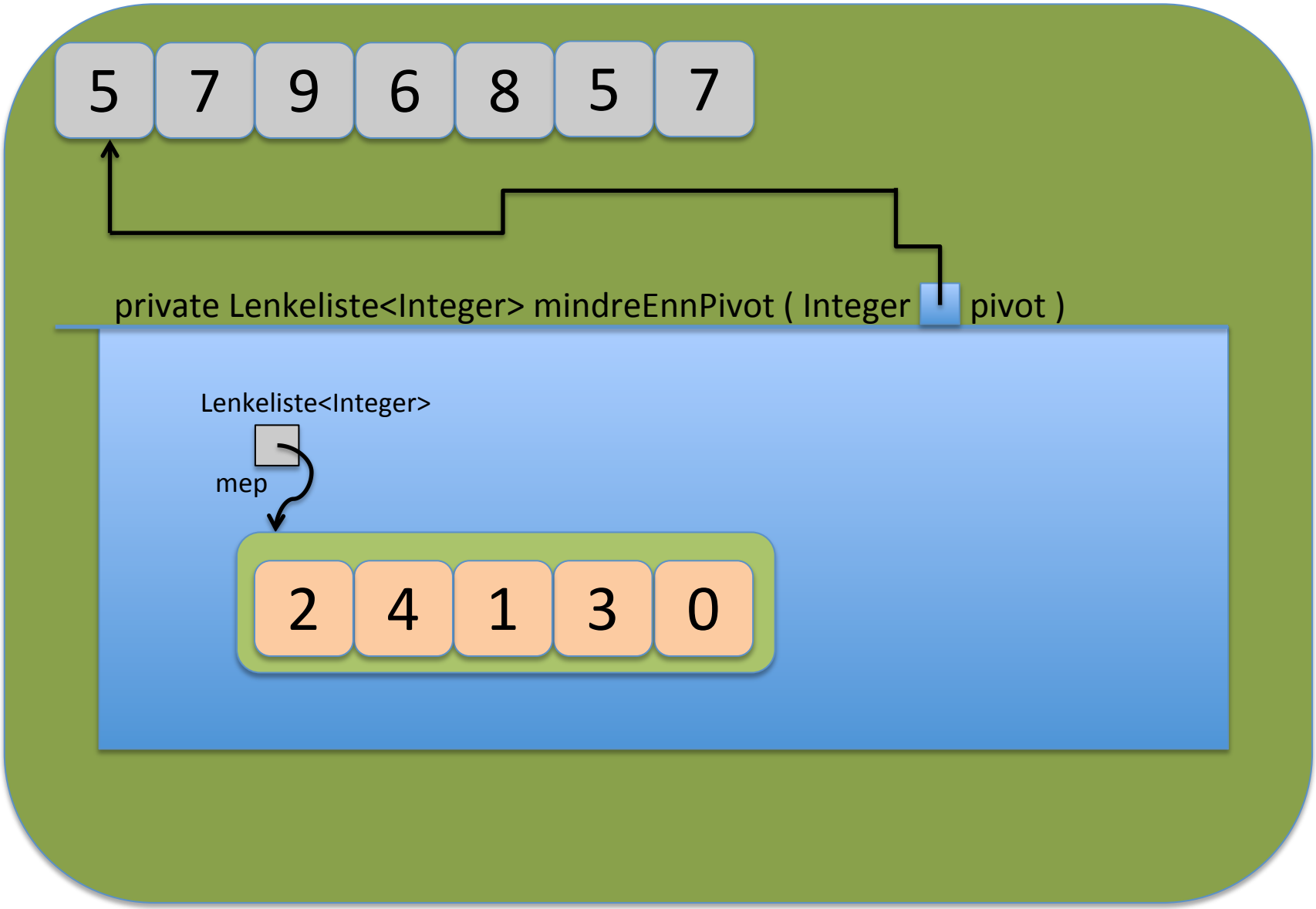

5
7
3
9
6
1
8
4
2
5
7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep
0

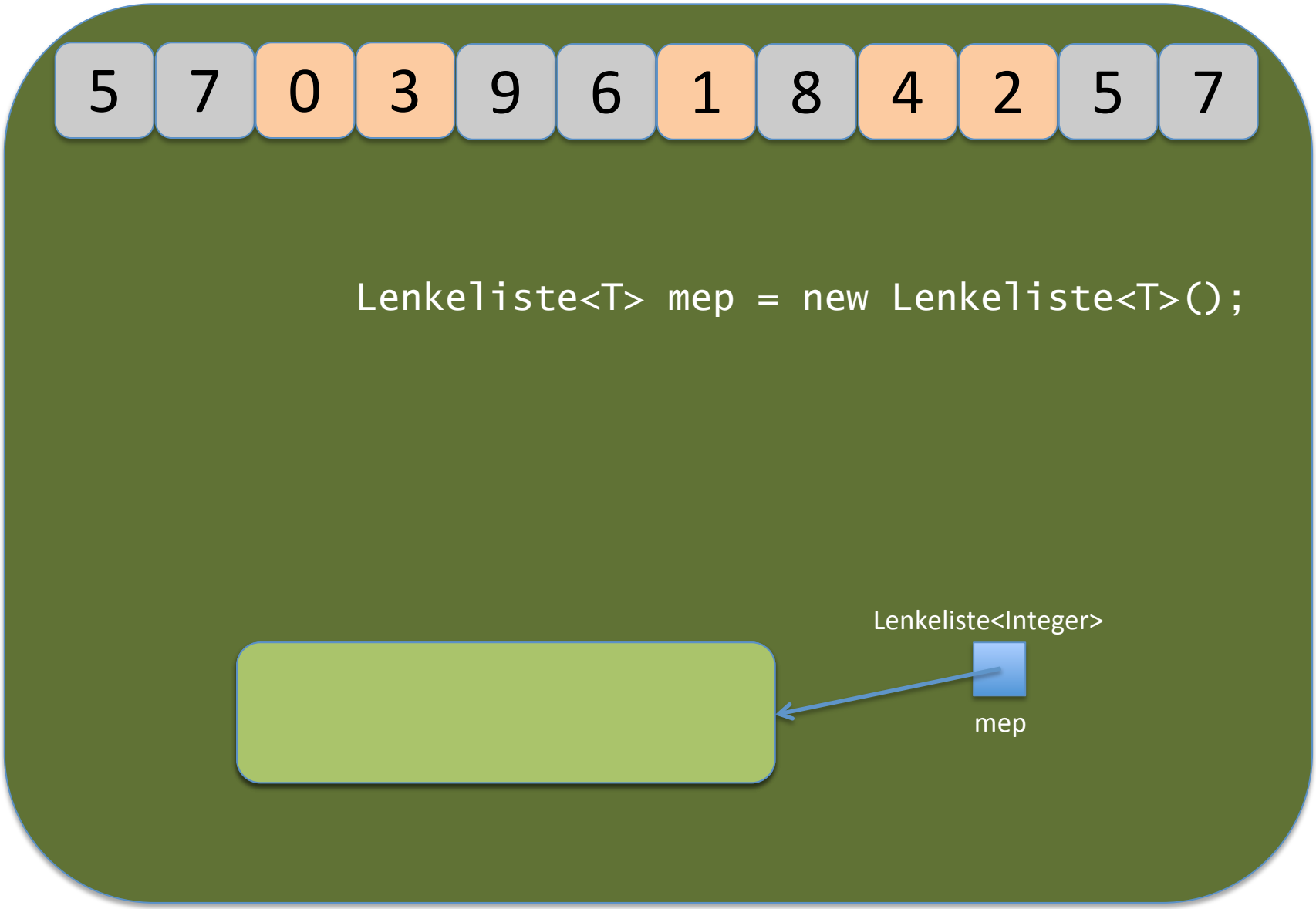

5
7
9
6
1
8
4
2
5
7
private Lenkeliste<Integer> mindreEnnPivot (Integer
pivot)
Lenkeliste<Integer>
mep
3
0

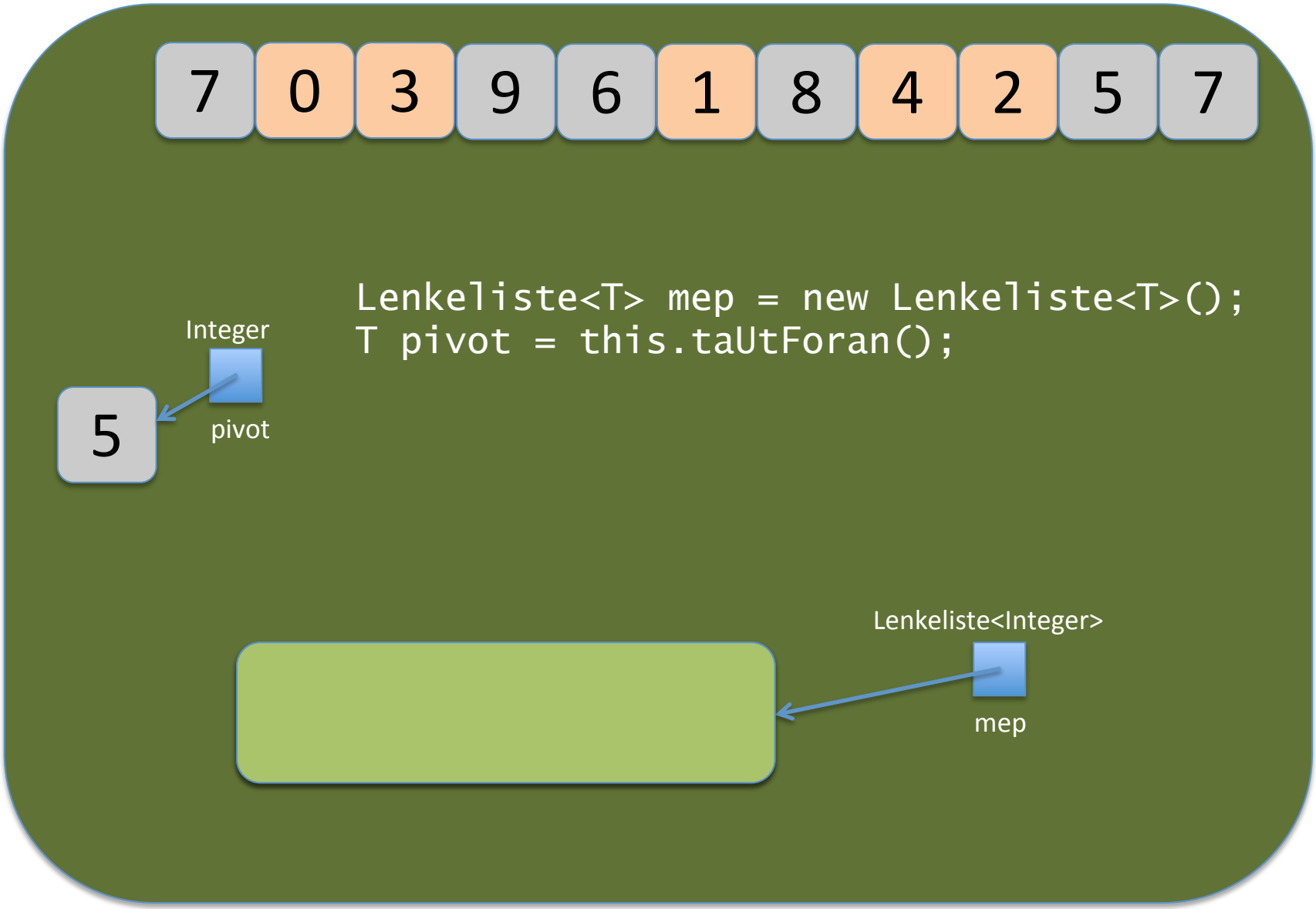

5
7
9
6
8
4
2
5
7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep
1
3
0

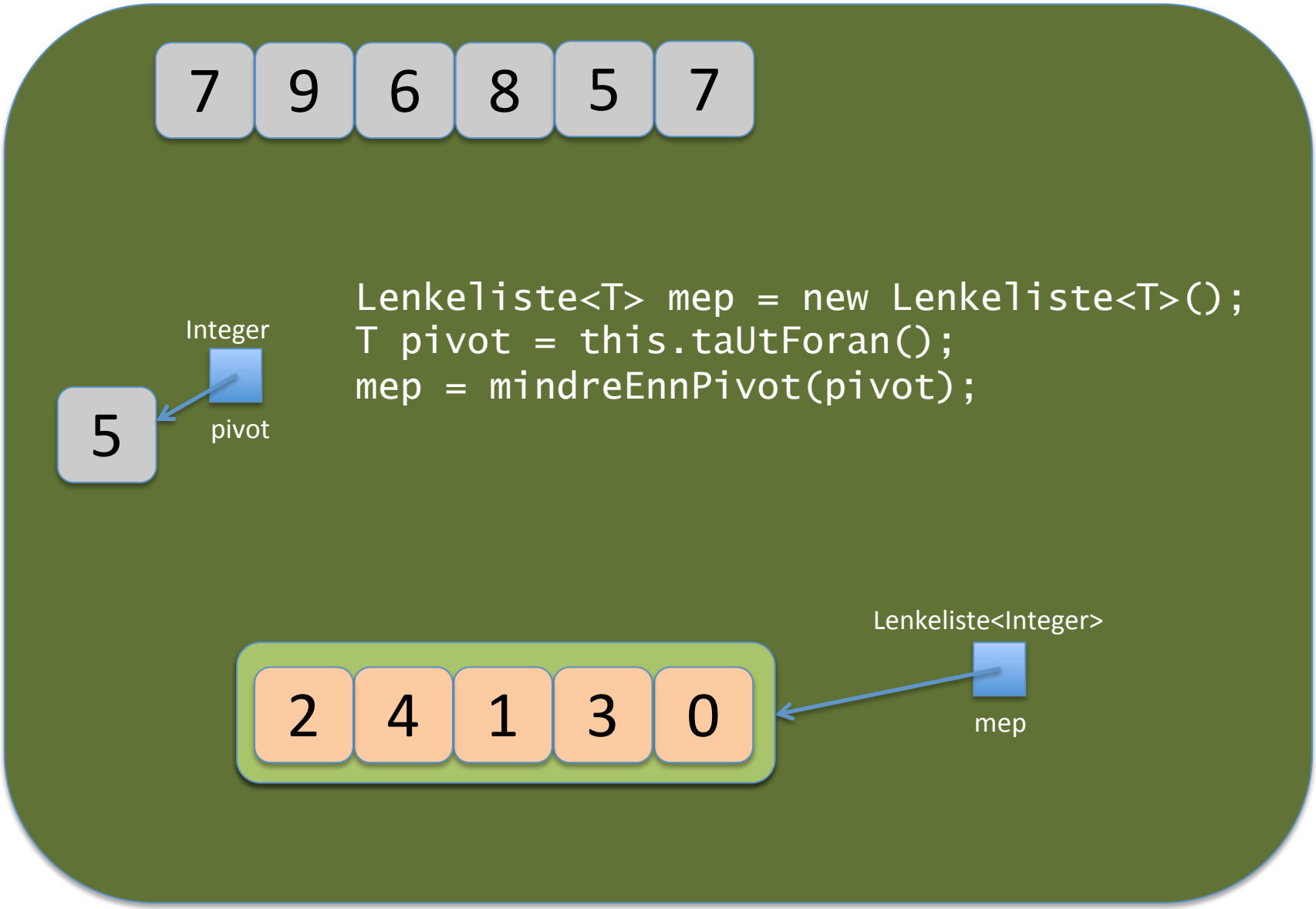

5
7
9
6
8
2
5
7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep
4
1
3
0

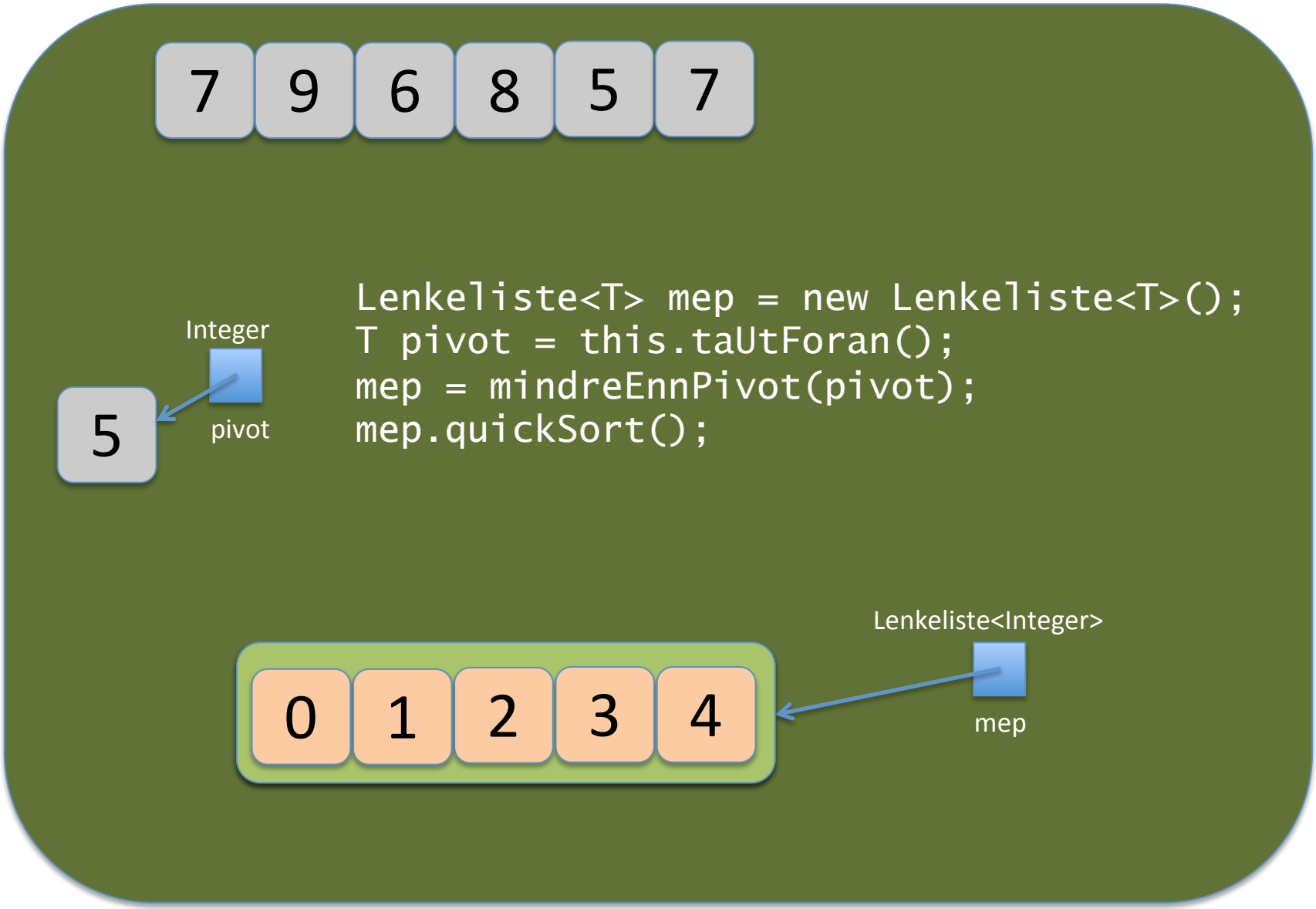

5
7
9
6
8
5
7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep
2
4
1
3
0

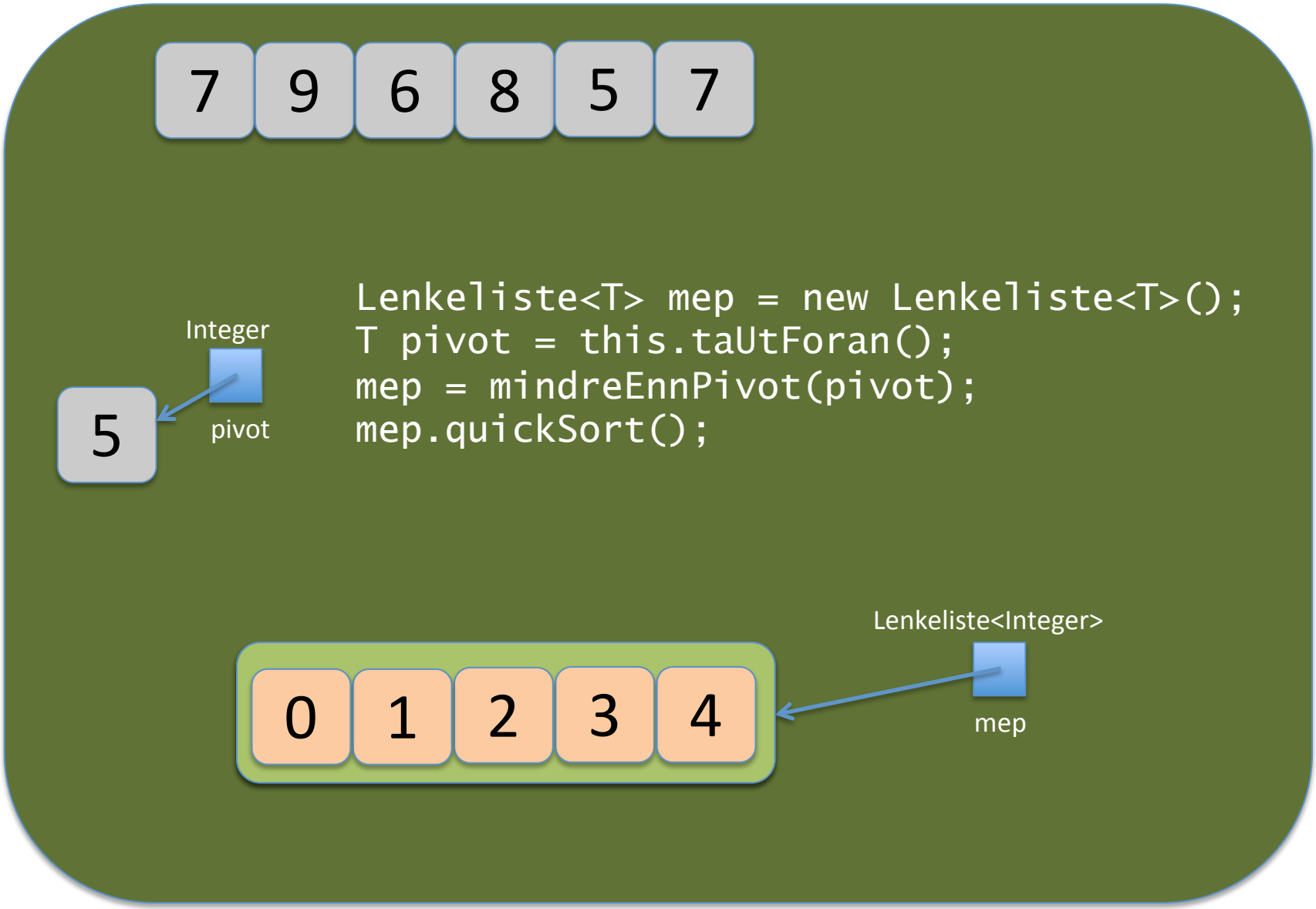

5 7 9 6 8 5 7
private Lenkeliste<Integer> mindreEnnPivot (Integer pivot)
Lenkeliste<Integer>
mep
2 4 1 3 0

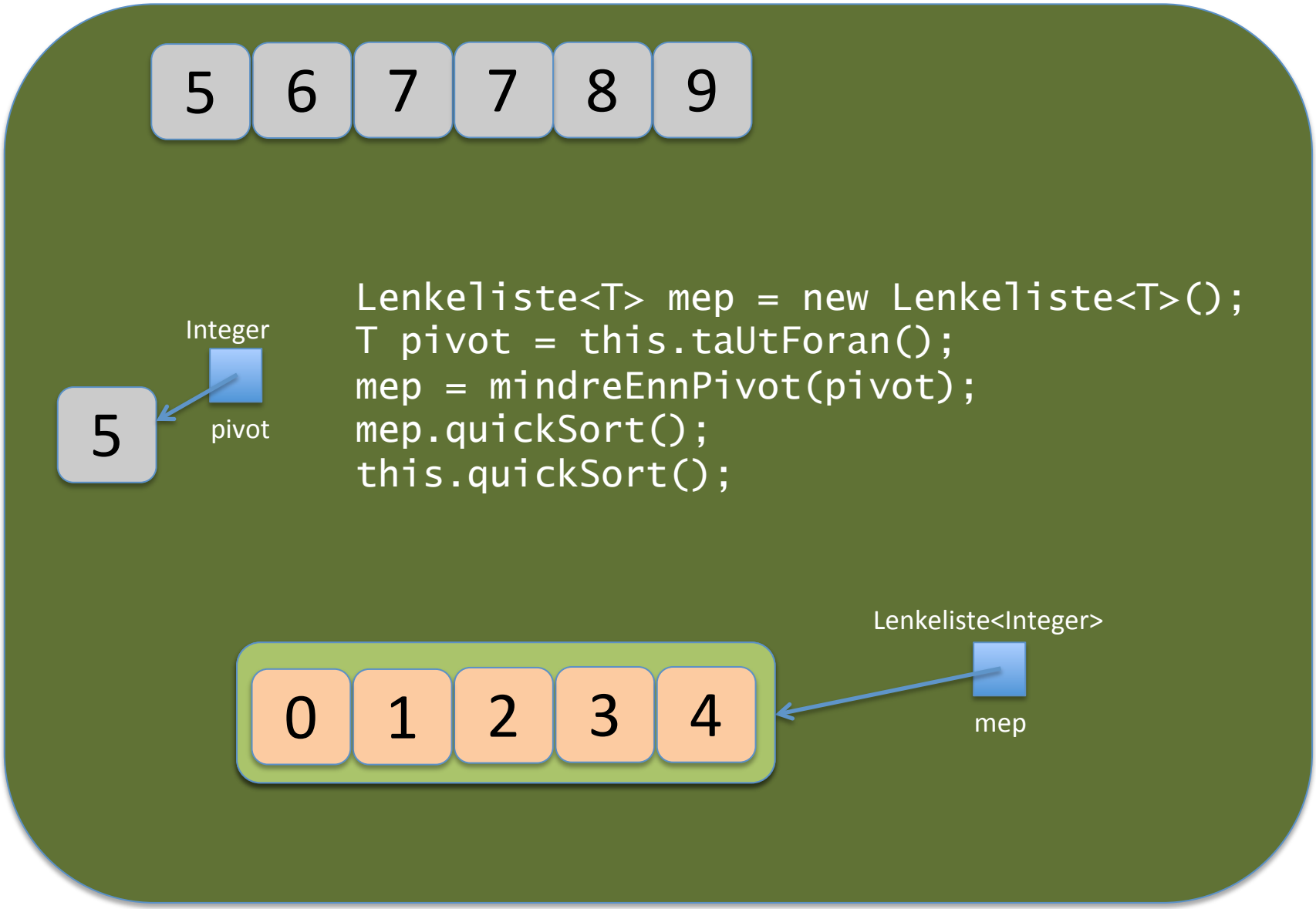

5 7 0 3 9 6 1 8 4 2 5 7
Lenkeliste<T> mep = new Lenkeliste<T>();
Lenkeliste<Integer>
mep

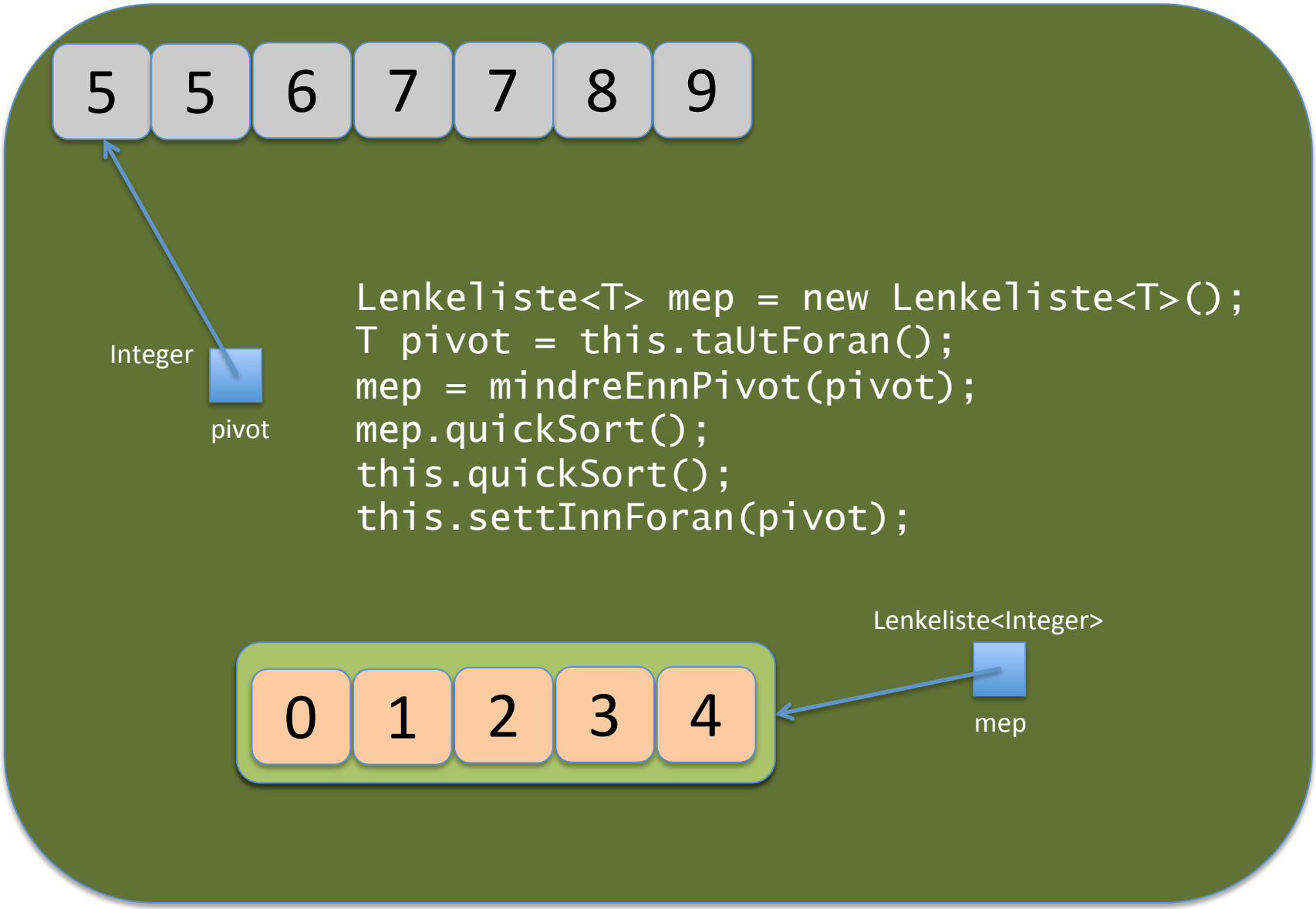

7 0 3 9 6 1 8 4 2 5 7
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
Integer
5
pivot
Lenkeliste<Integer>
mep

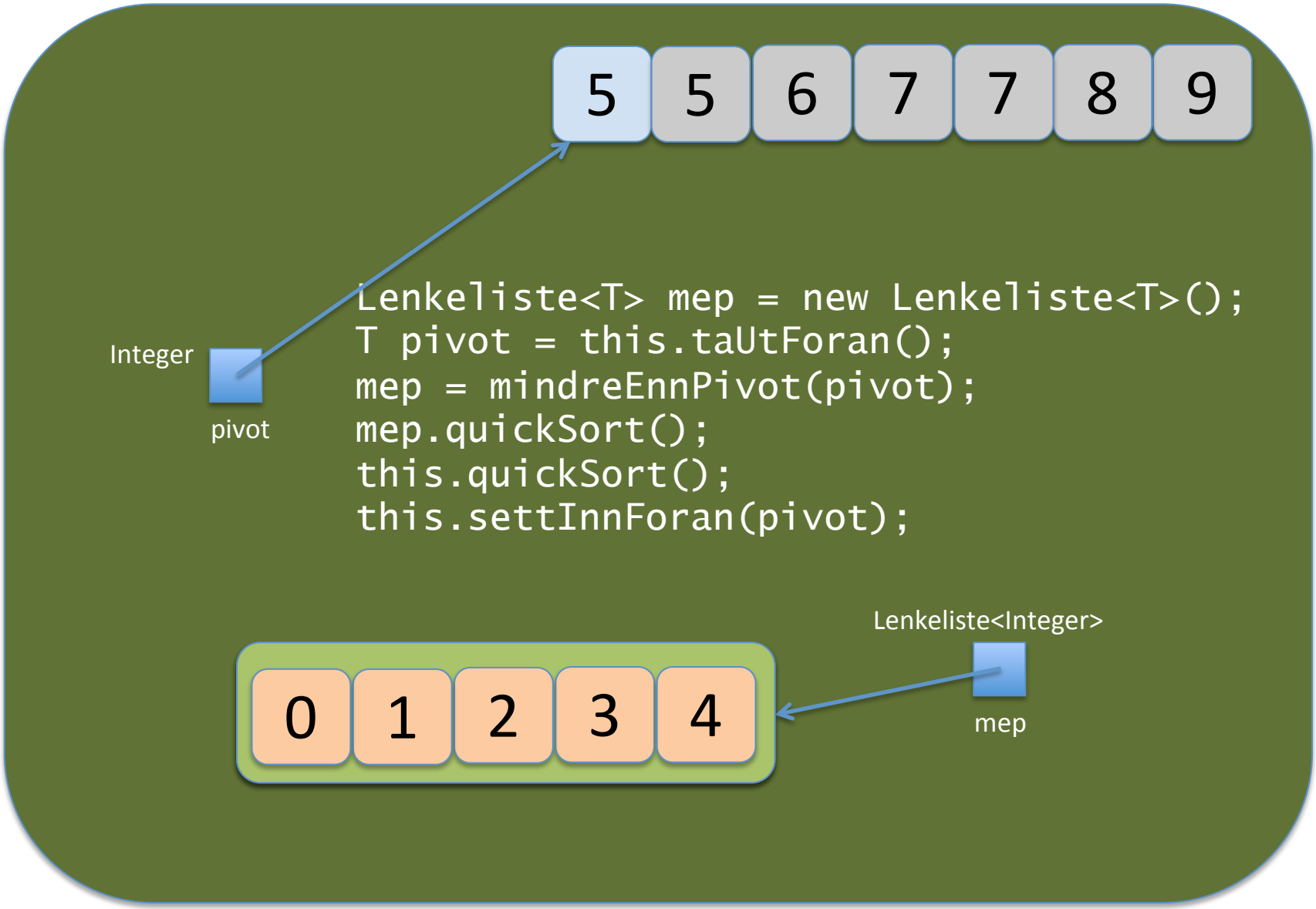

7 9 6 8 5 7
Integer
5
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
Lenkeliste<Integer>
2 4 1 3 0
mep

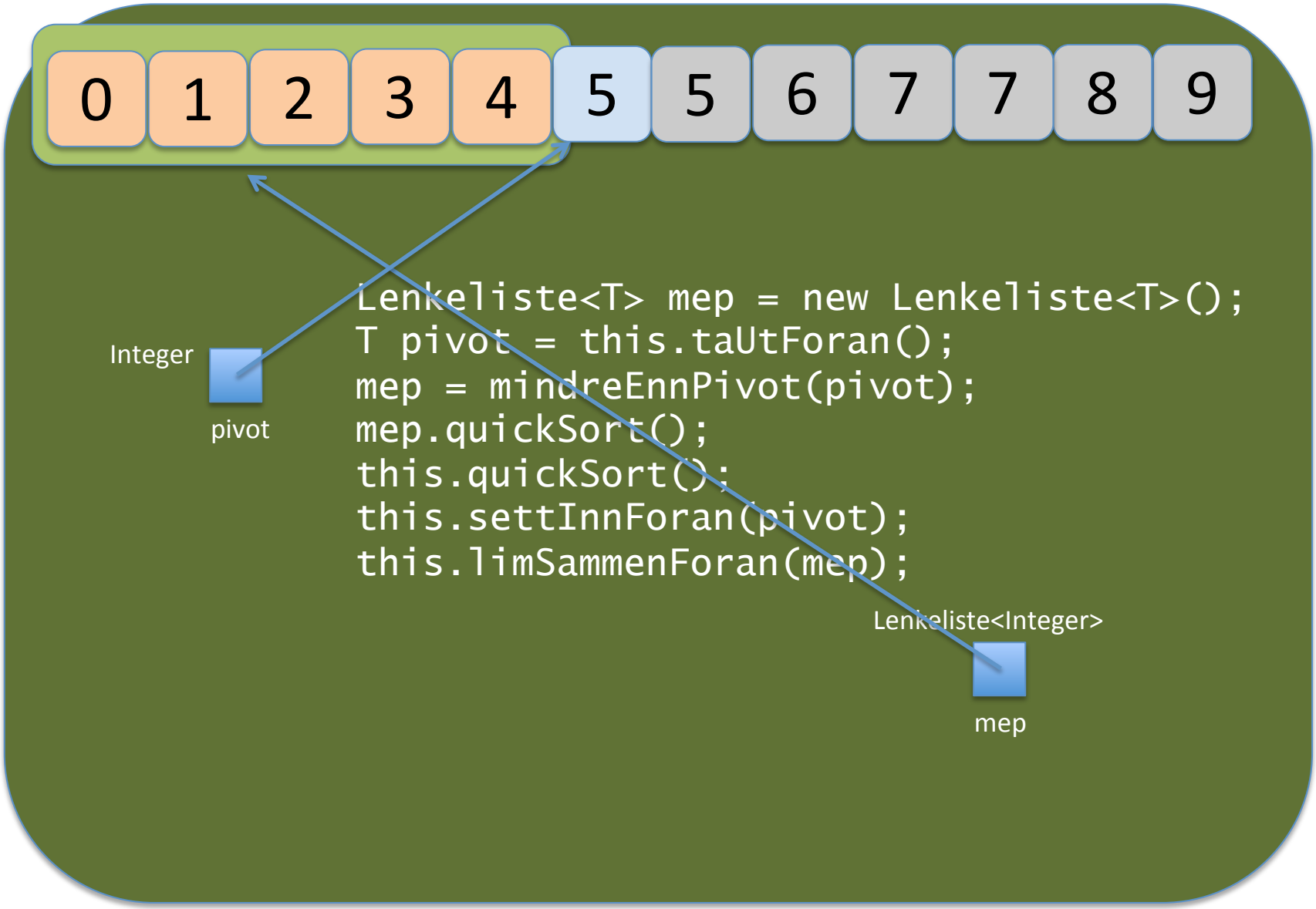

7 9 6 8 5 7
Integer
5
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
Lenkeliste<Integer>
0 1 2 3 4
mep

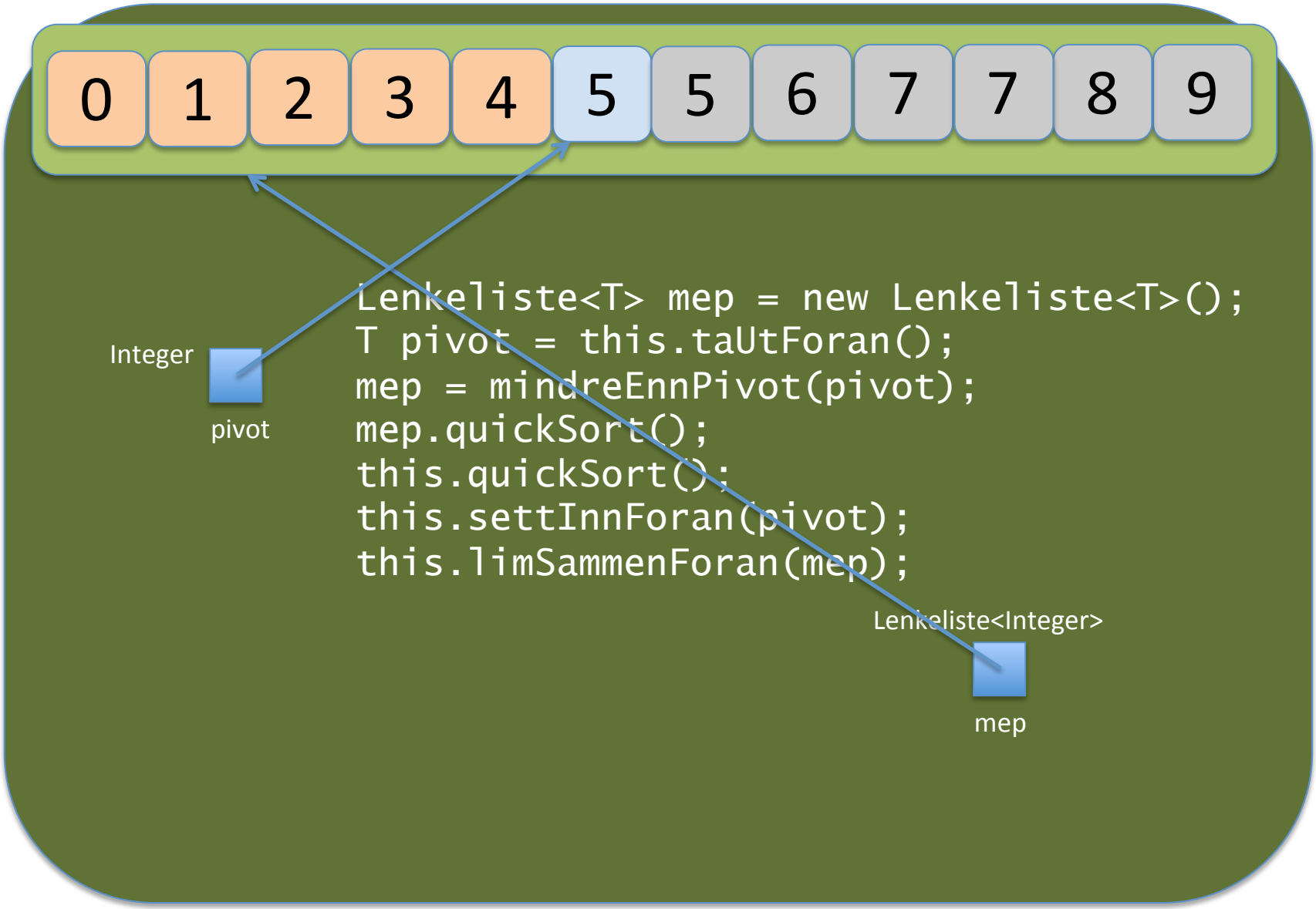

7 9 6 8 5 7
Integer
5
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
Lenkeliste<Integer>
0 1 2 3 4
mep

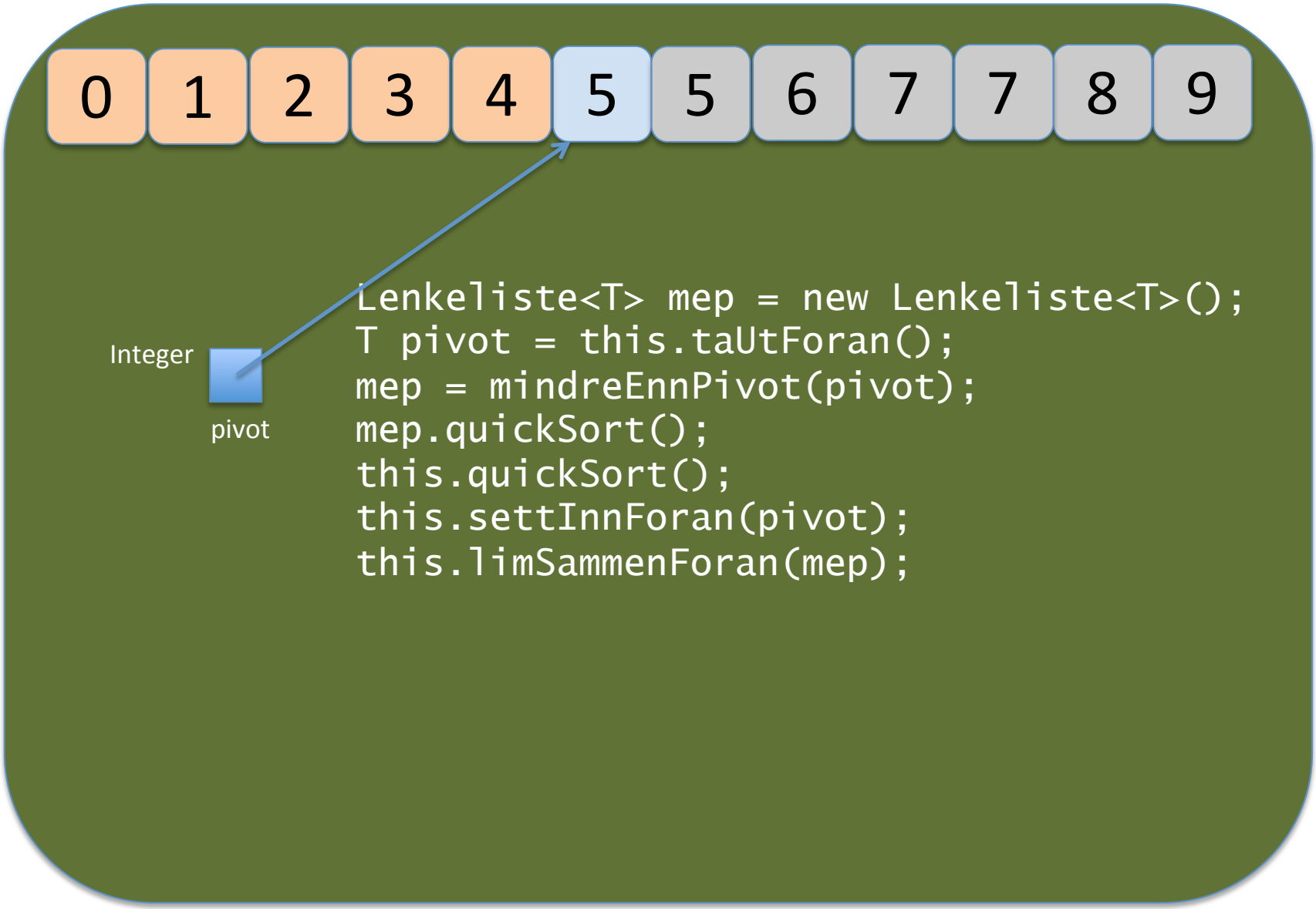

5 6 7 7 8 9
Integer
5
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
Lenkeliste<Integer>
0 1 2 3 4
mep


5 5 6 7 7 8 9
Integer
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
Lenkeliste<Integer>
0 1 2 3 4
mep


5 5 6 7 7 8 9
Integer
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
Lenkeliste<Integer>
0 1 2 3 4
mep

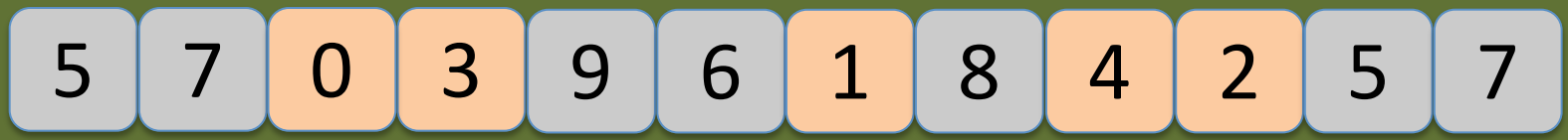

0 1 2 3 4 5 5 6 7 7 8 9
Integer
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
this.limSammenForan(mep);
Lenkeliste<Integer>
mep


0 1 2 3 4 5 5 6 7 7 8 9
Integer
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
this.limSammenForan(mep);
Lenkeliste<Integer>
mep


0
1
2
3
4
5
5
6
7
7
8
9
Integer
pivot
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
this.limSammenForan(mep);

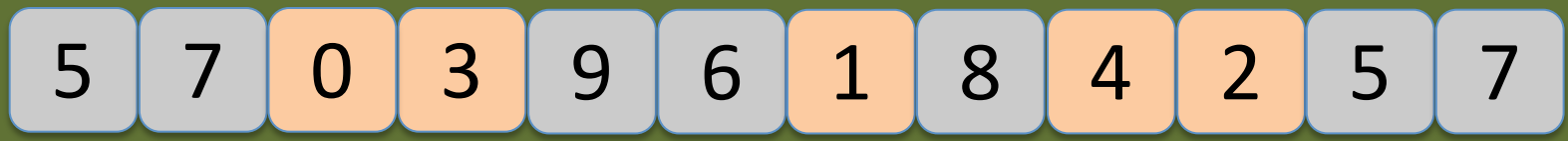

```
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
this.limSammenForan(mep);
```


```
public void quickSort() {
    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort();
        this.quickSort();
        this.settInnForan(pivot);
        this.limSammenForan(mep);
    }
}
```



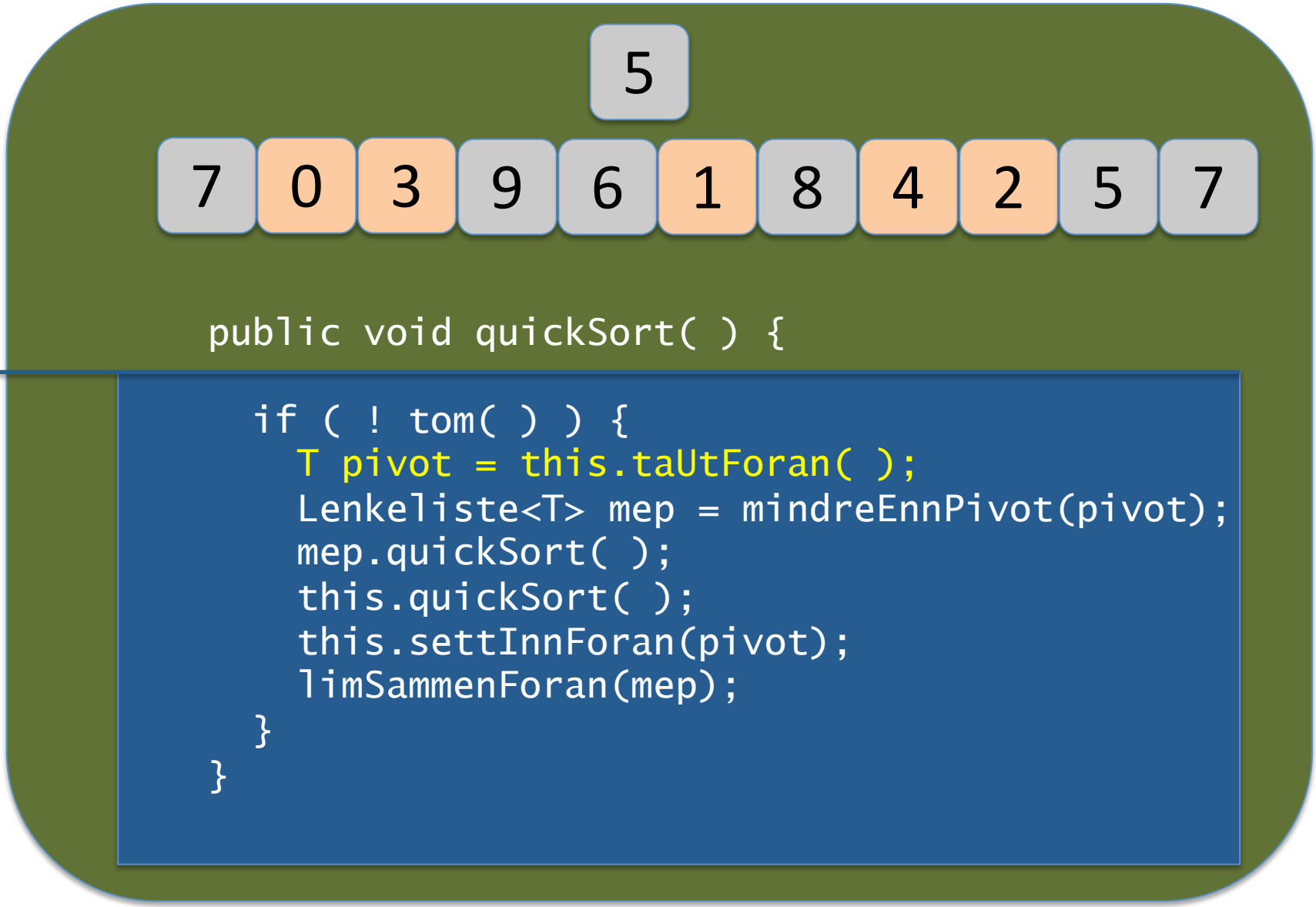

```
public void quickSort( ) {
    if ( ! tom( ) ) {
      T pivot = this.taUtForan( );
      Lenkeliste<T> mep = mindreEnnPivot(pivot);
      mep.quickSort( );
      this.quickSort( );
      this.settInnForan(pivot);
      limSammenForan(mep);
    }
}
```

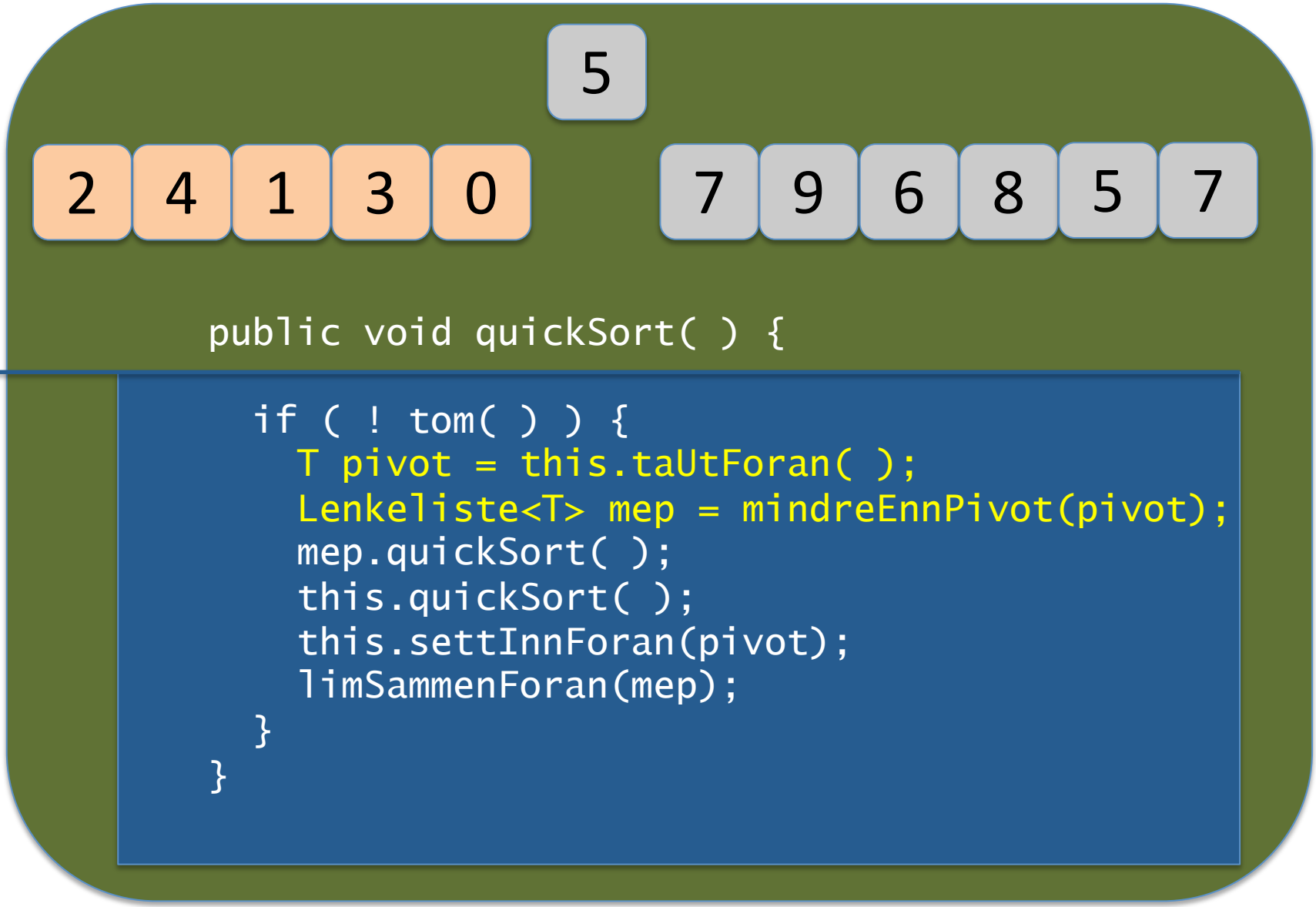
Hvis lista er tom, returner uten å gjøre noe. Basistilfelle: ingen rekursive kall.



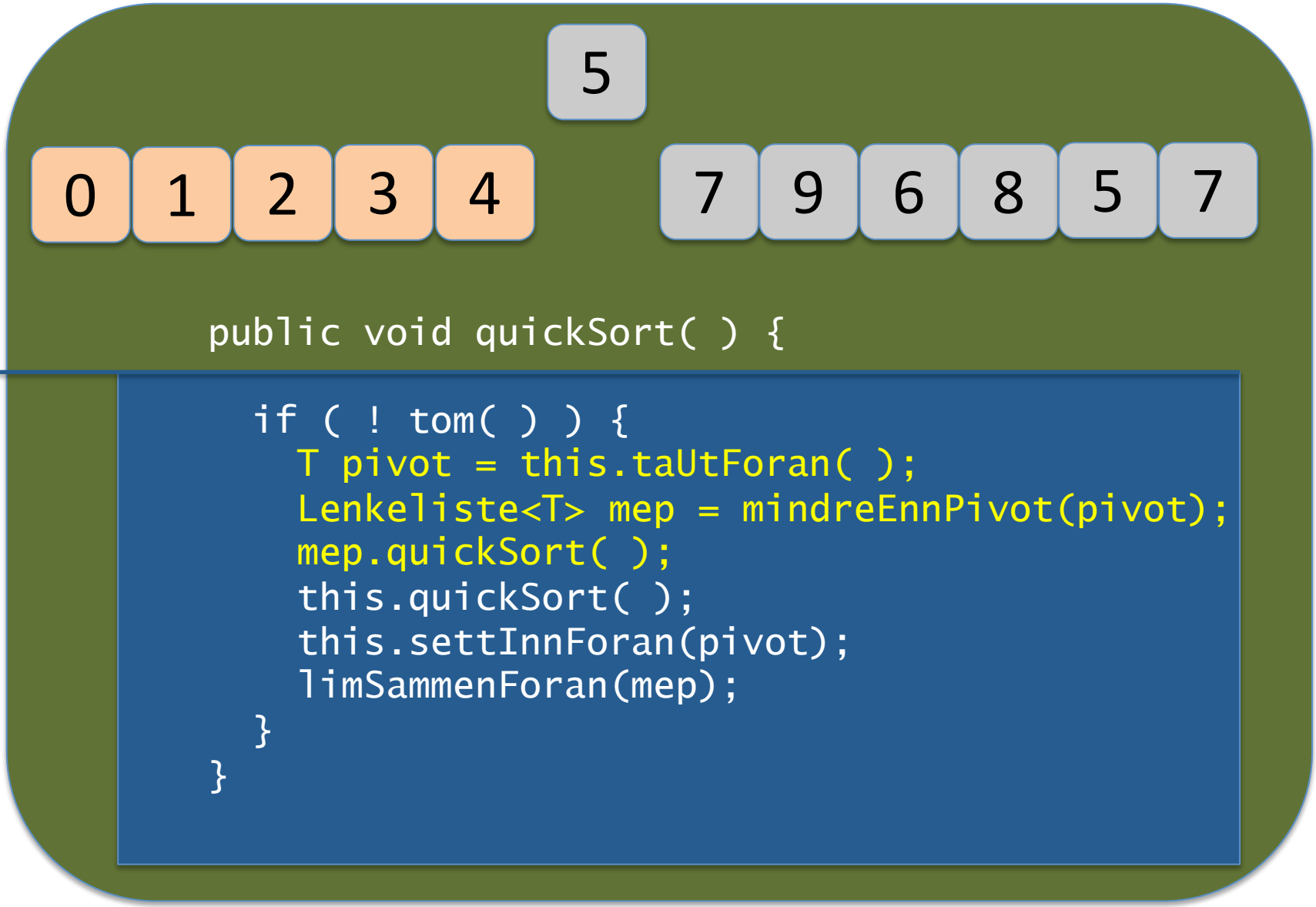

```
public void quickSort( ) {
    if ( ! tom( ) ) {
      T pivot = this.taUtForan( );
      Lenkeliste<T> mep = mindreEnnPivot(pivot);
      mep.quickSort( );
      this.quickSort( );
      this.settInnForan(pivot);
      limSammenForan(mep);
    }
}
```

Hvis lista ikke er tom, ta først ut objektet foran --- pivotobjektet

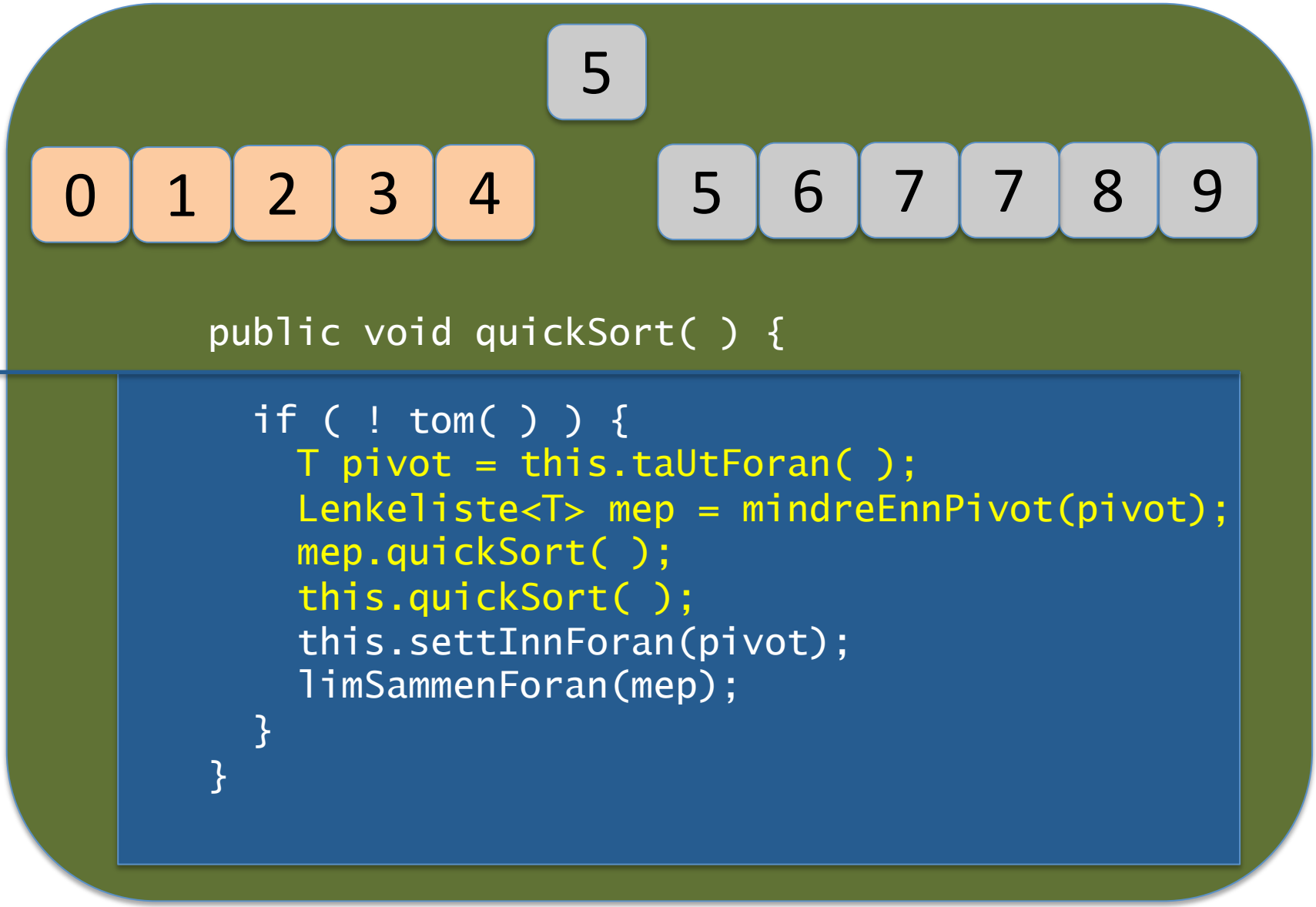

5
7 0 3 9 6 1 8 4 2 5 7
public void quickSort( ) {
if (! tom( )) {
T pivot = this.taUtForan( );
Lenkeliste<T> mep = mindreEnnPivot(pivot);
mep.quickSort( );
this.quickSort( );
this.settInnForan(pivot);
limSammenForan(mep);
}
}



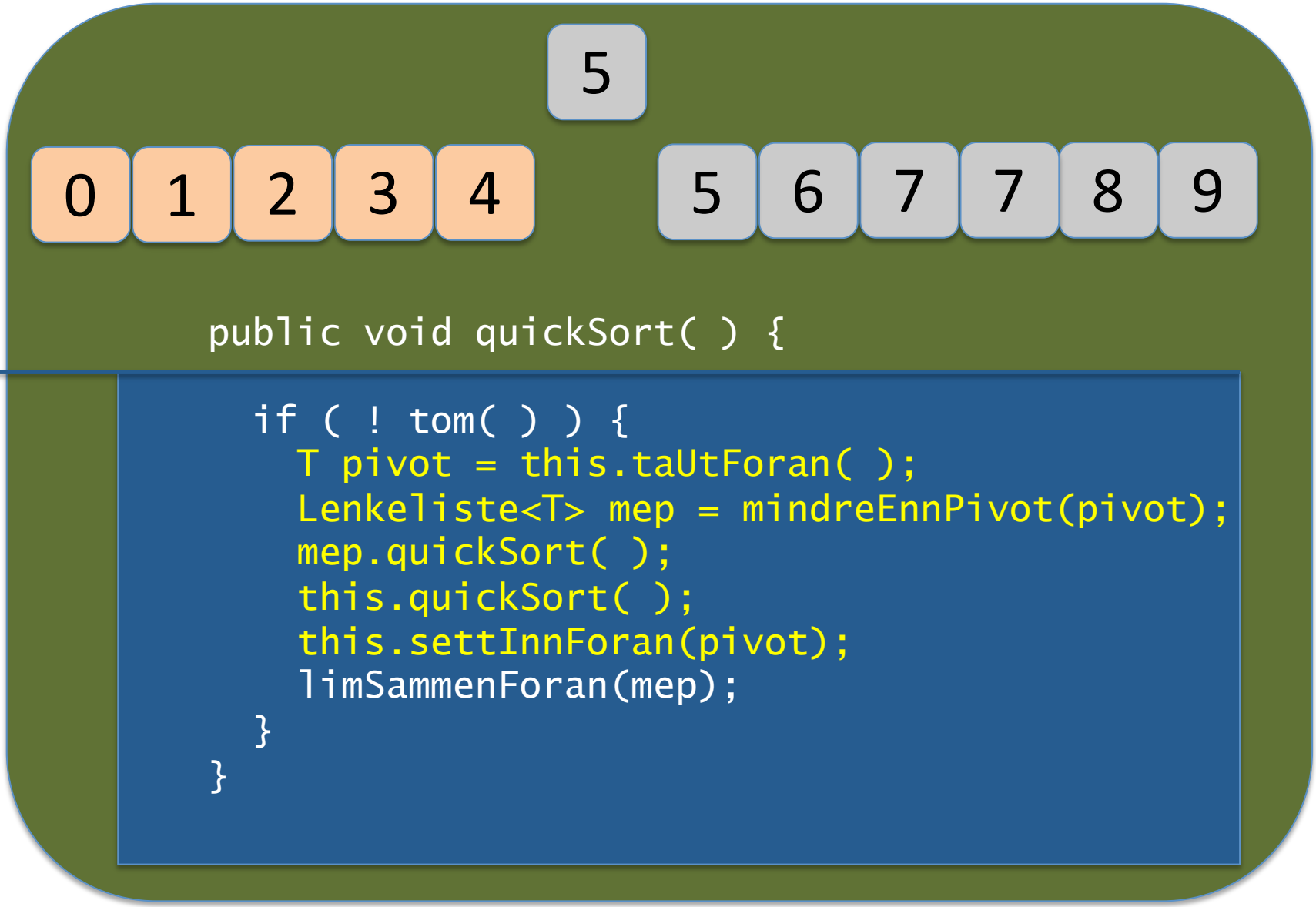
Skill ut objekter som er mindre enn pivot fra lista og legg dem inn i ei ny lenkeliste (mep)



Sorter lista med små elementer (mep) vha. quicksort (rekursivt kall)

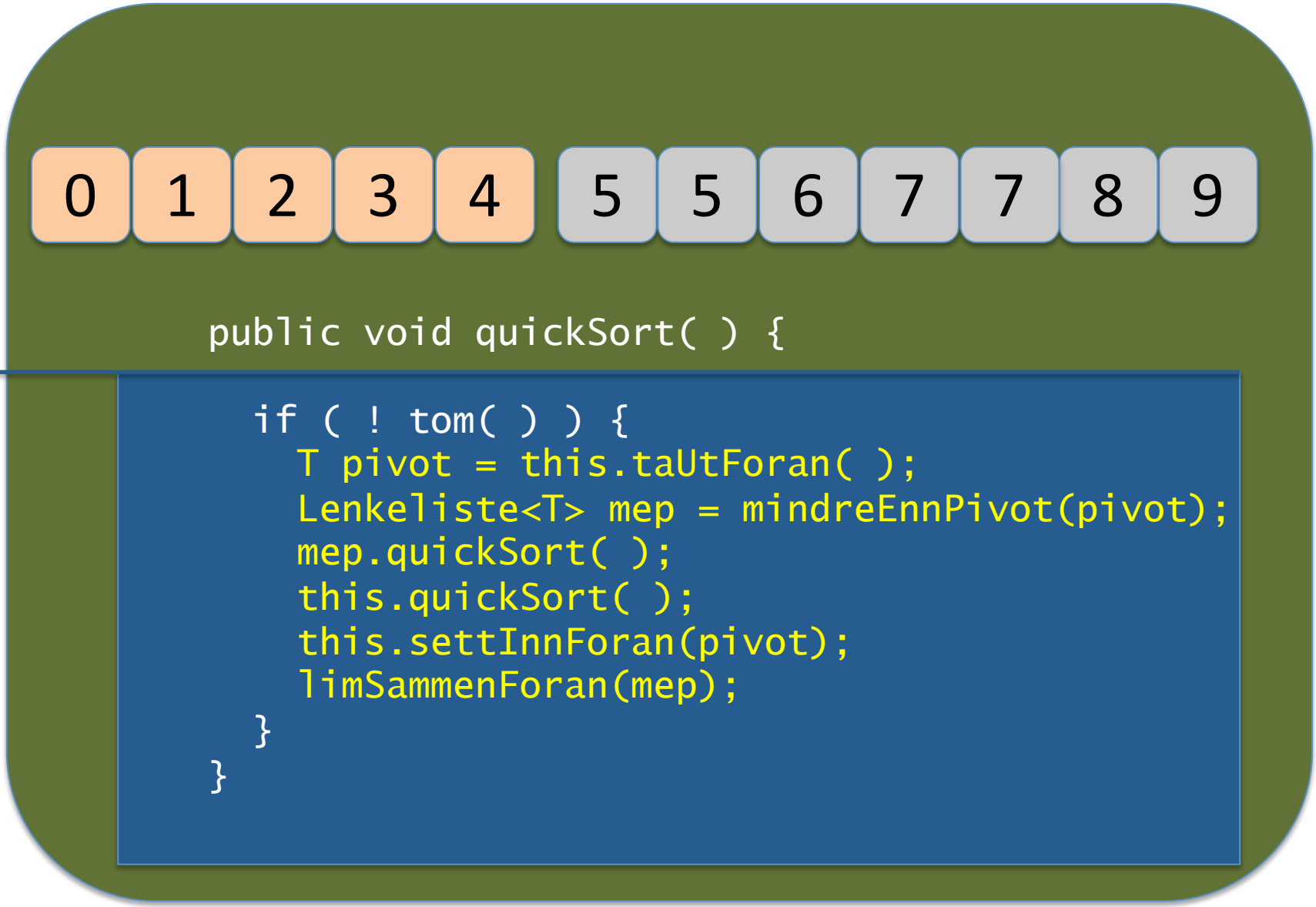


Sorter restlista med store elementer (this) vha. quicksort (rekursivt kall)



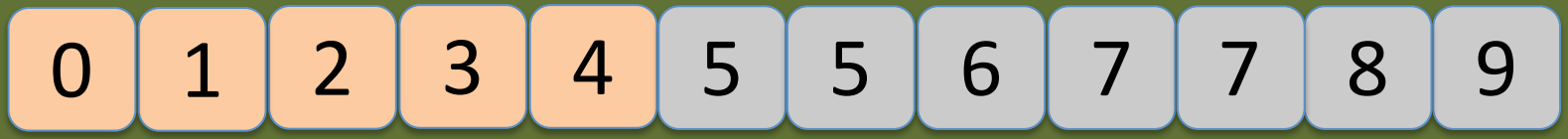
Sett pivotobjektet foran lista med de store verdiene


```
public void quickSort( ) {

    if ( ! tom( ) ) {
      T pivot = this.taUtForan( );
      Lenkeliste<T> mep = mindreEnnPivot(pivot);
      mep.quickSort( );
      this.quickSort( );
      this.settInnForan(pivot);
      limSammenForan(mep);
    }
}
```

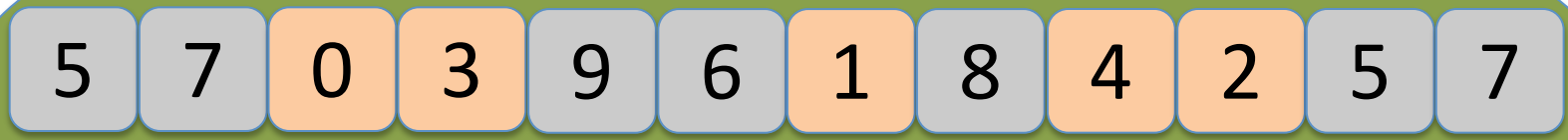


Skjøt de to listene sammen, den med små verdier foran de store …

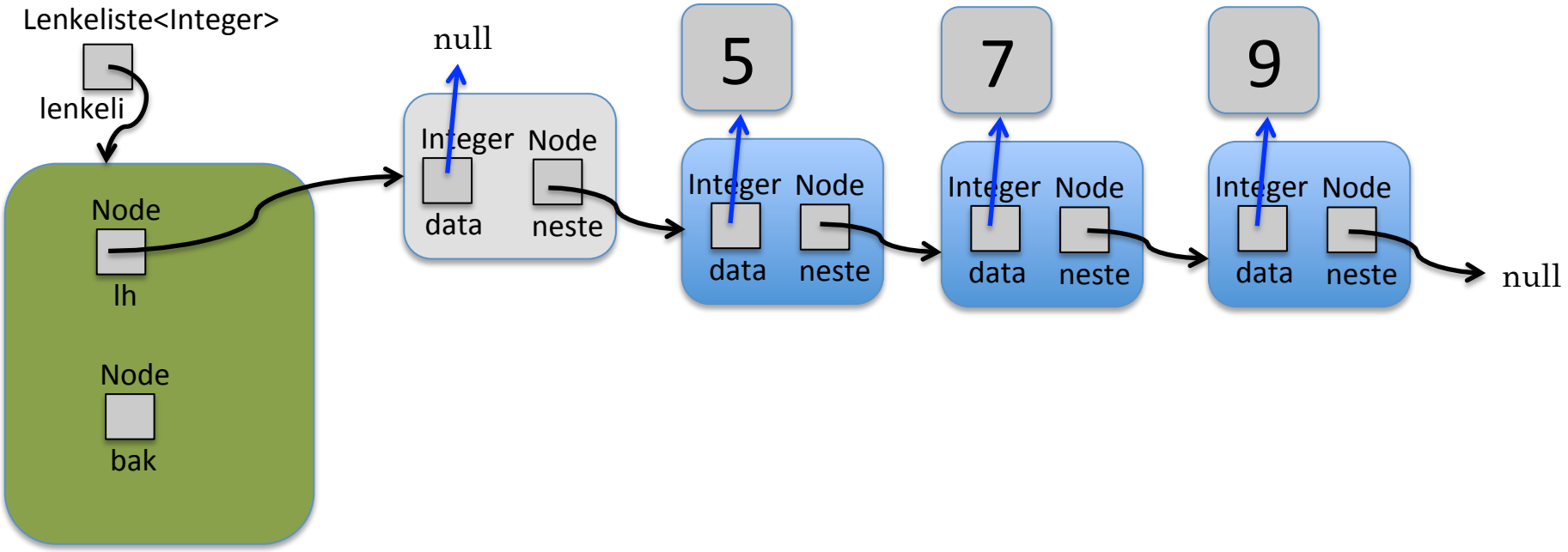

```
public void quickSort( ) {
    if ( ! tom( ) ) {
      T pivot = this.taUtForan( );
      Lenkeliste<T> mep = mindreEnnPivot(pivot);
      mep.quickSort( );
      this.quickSort( );
      this.settInnForan(pivot);
      limSammenForan(mep);
    }
}
```

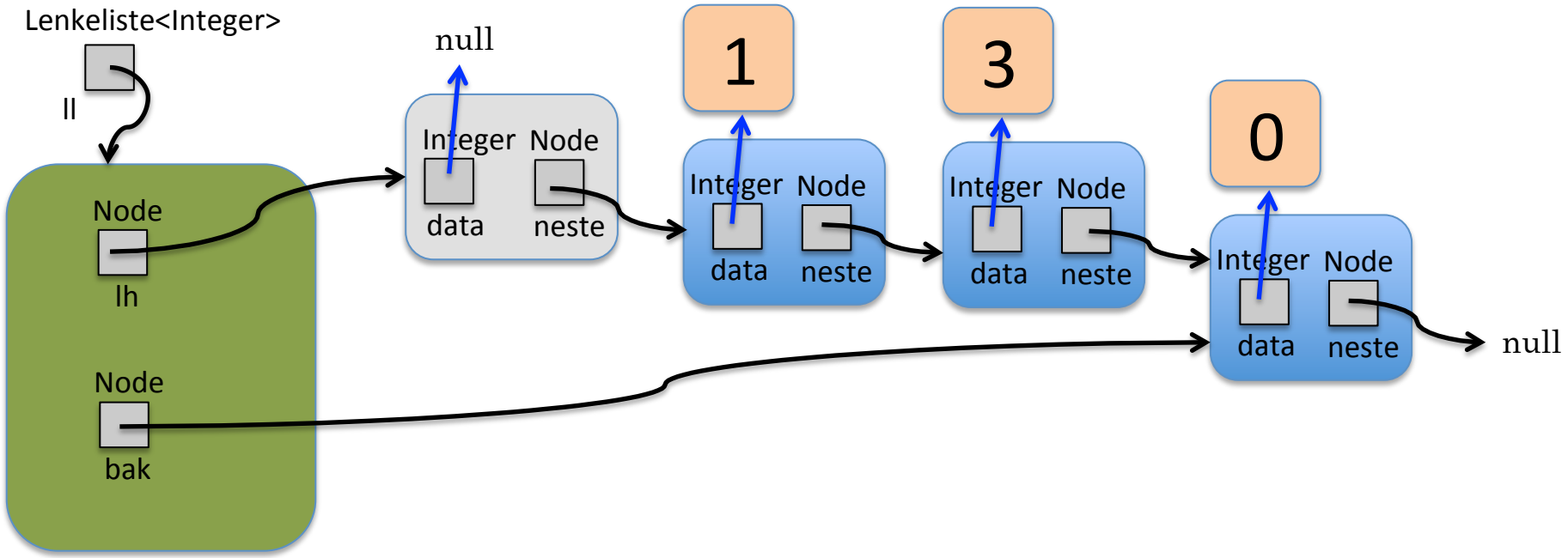


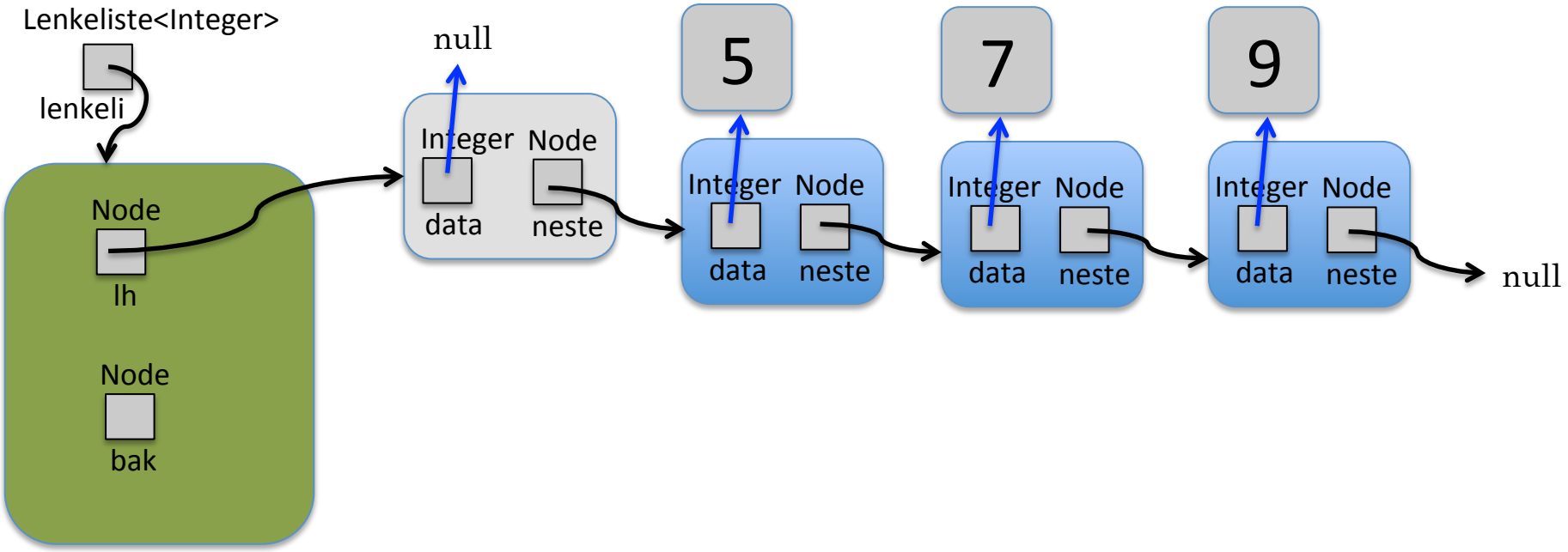
…et voilà !

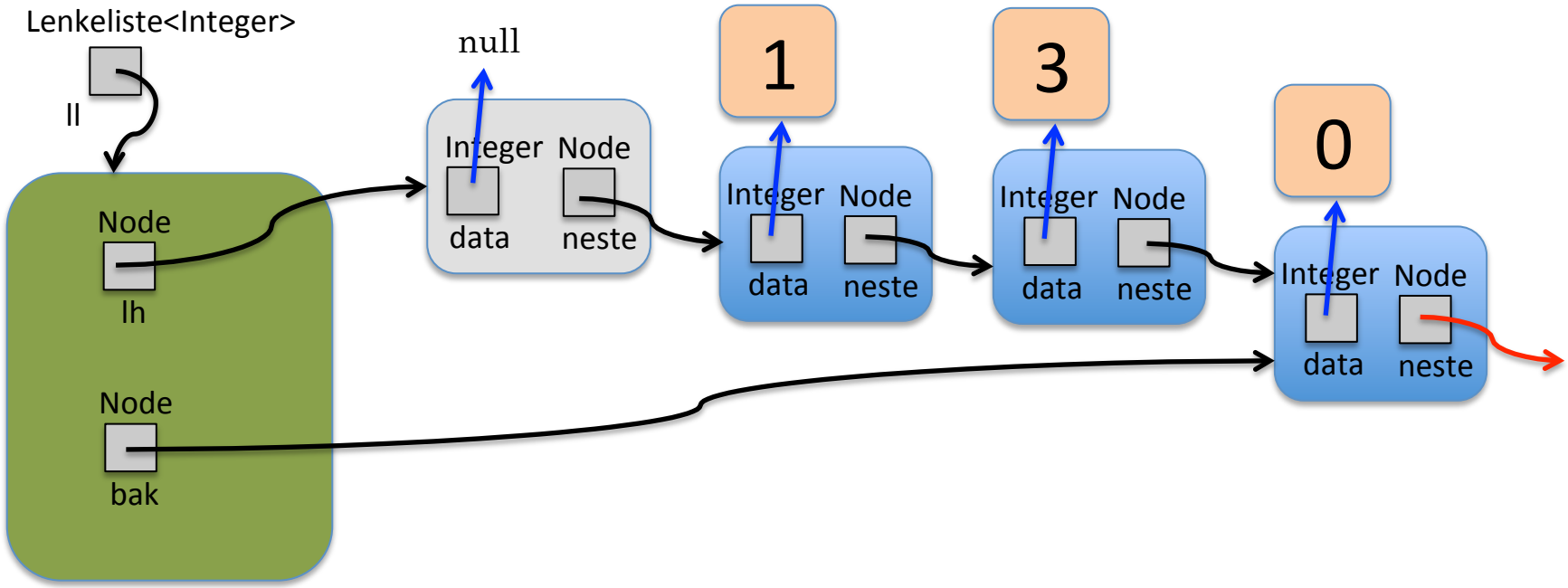


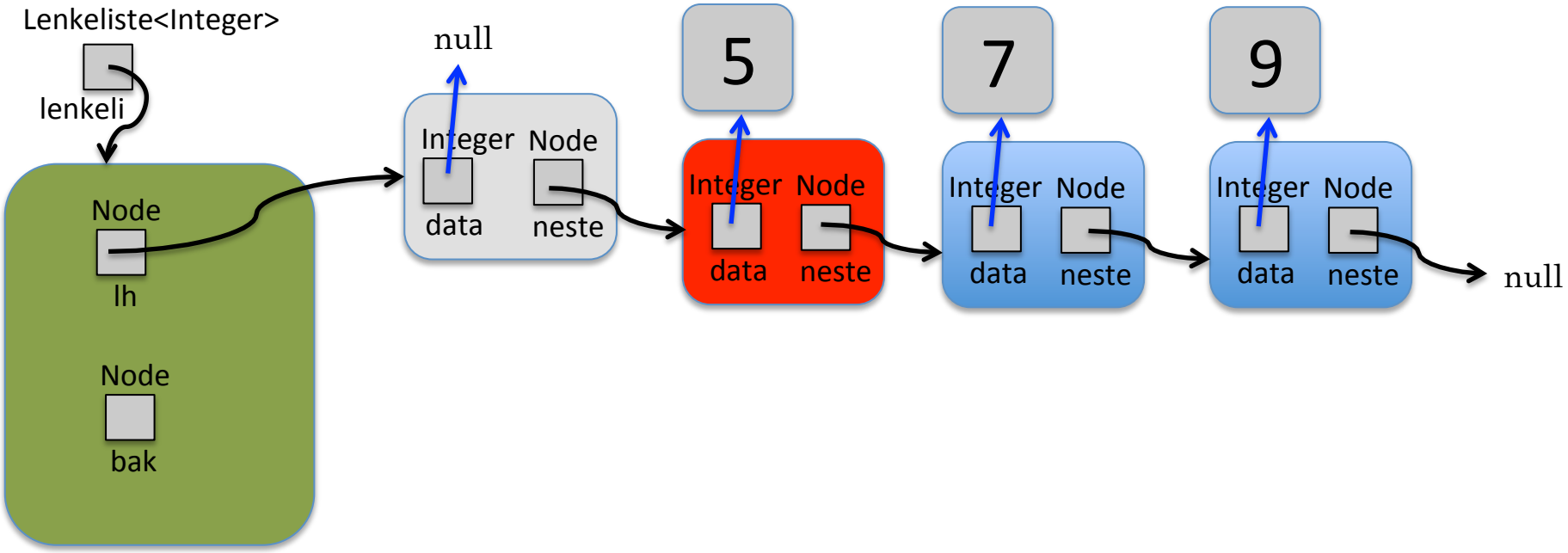

```
private Lenkeliste<T> mindreEnnPivot(T pivot) {
    Lenkeliste<T> mep = new Lenkeliste<T>();
    Iterator<T> it = iterator();
    while (it.hasNext()) {
        T t = it.next();
        if ( t.compareTo(pivot) <= 0 ) {
           mep.settInnForan(t);
           it.remove();
        }
    }
    return mep;
}
```

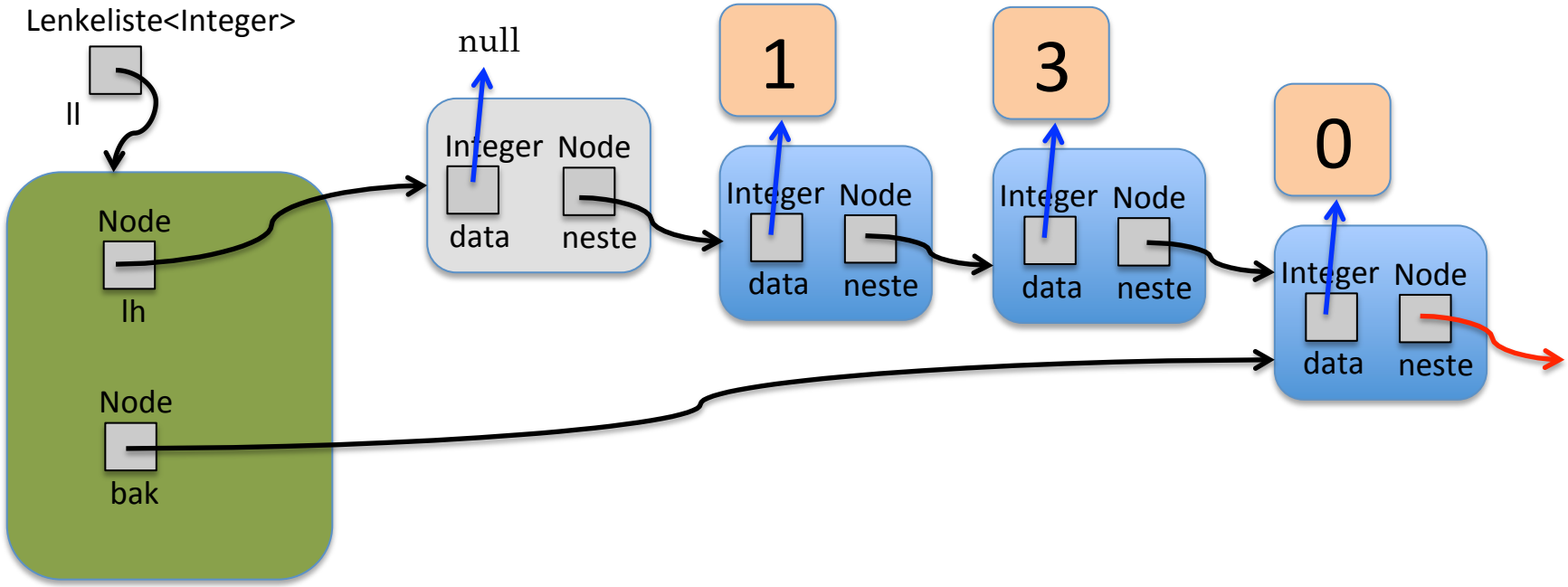

Lenkeliste<Integer>
lenkeli
Node
lh
Node
bak
null
Integer
Node
data
neste
5
7
9
null

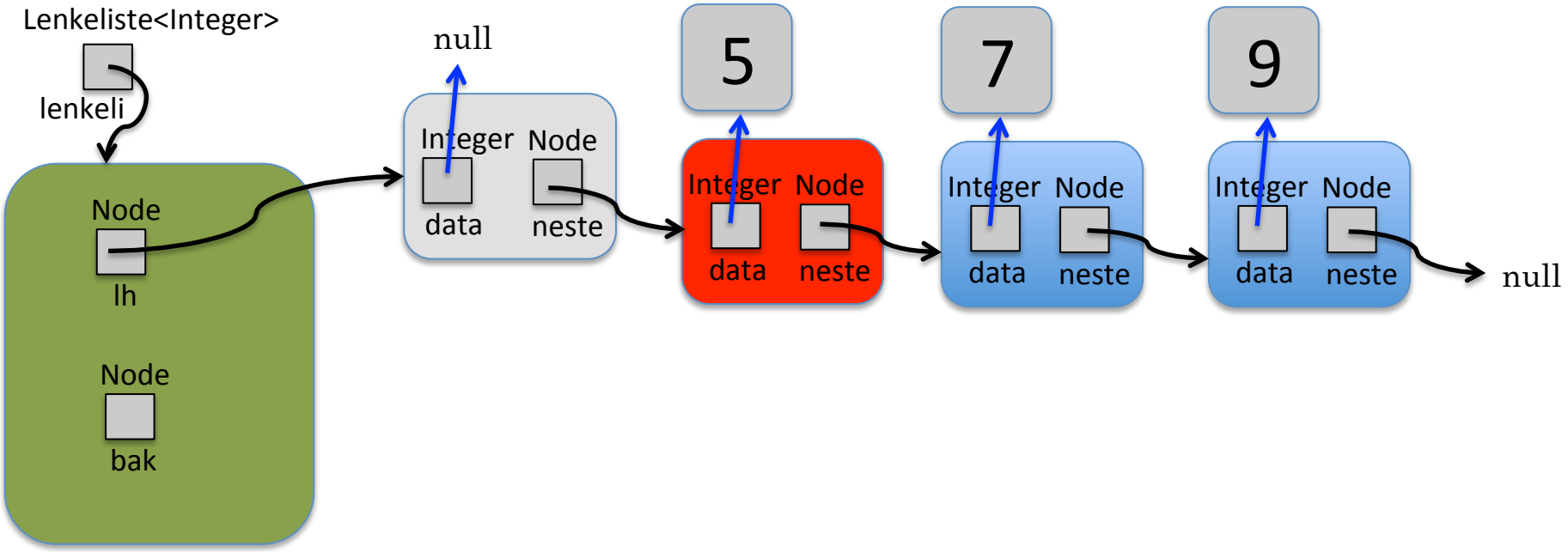

Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer
Node
data
neste
1
3
0
null

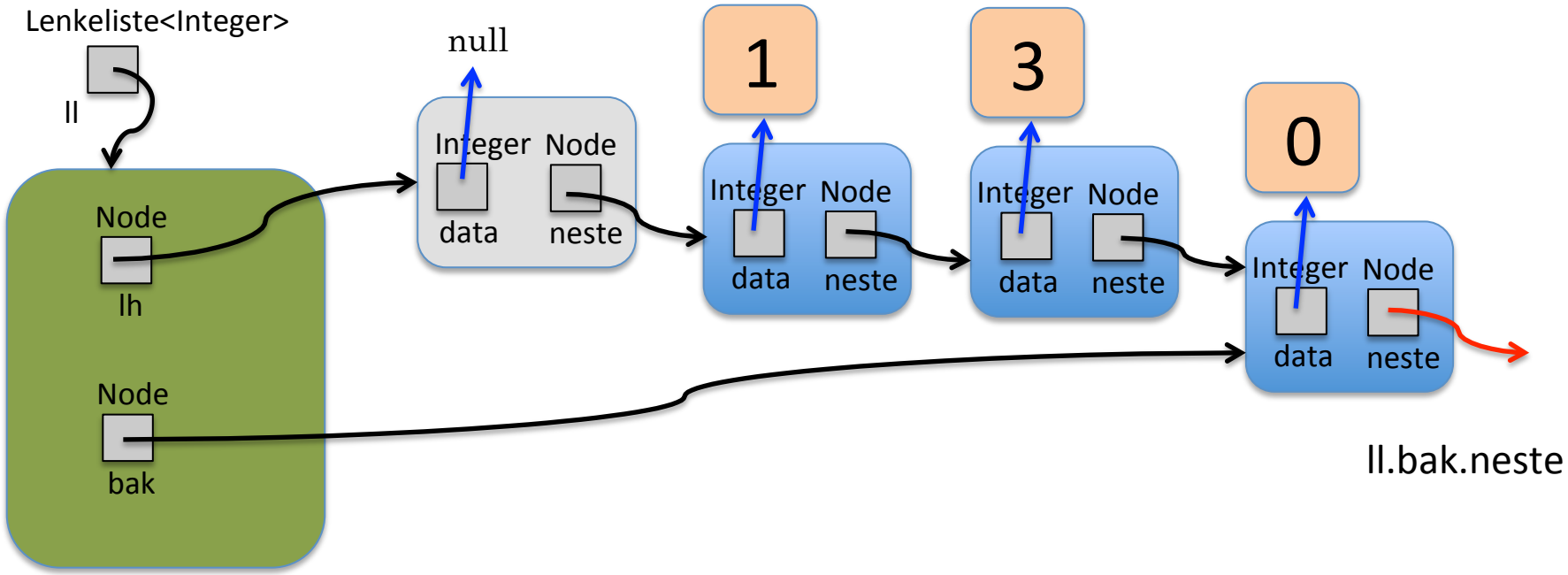

Lenkeliste<Integer>
lenkeli
Node
lh
Node
bak
null
Integer
Node
data
neste
5
7
9
null

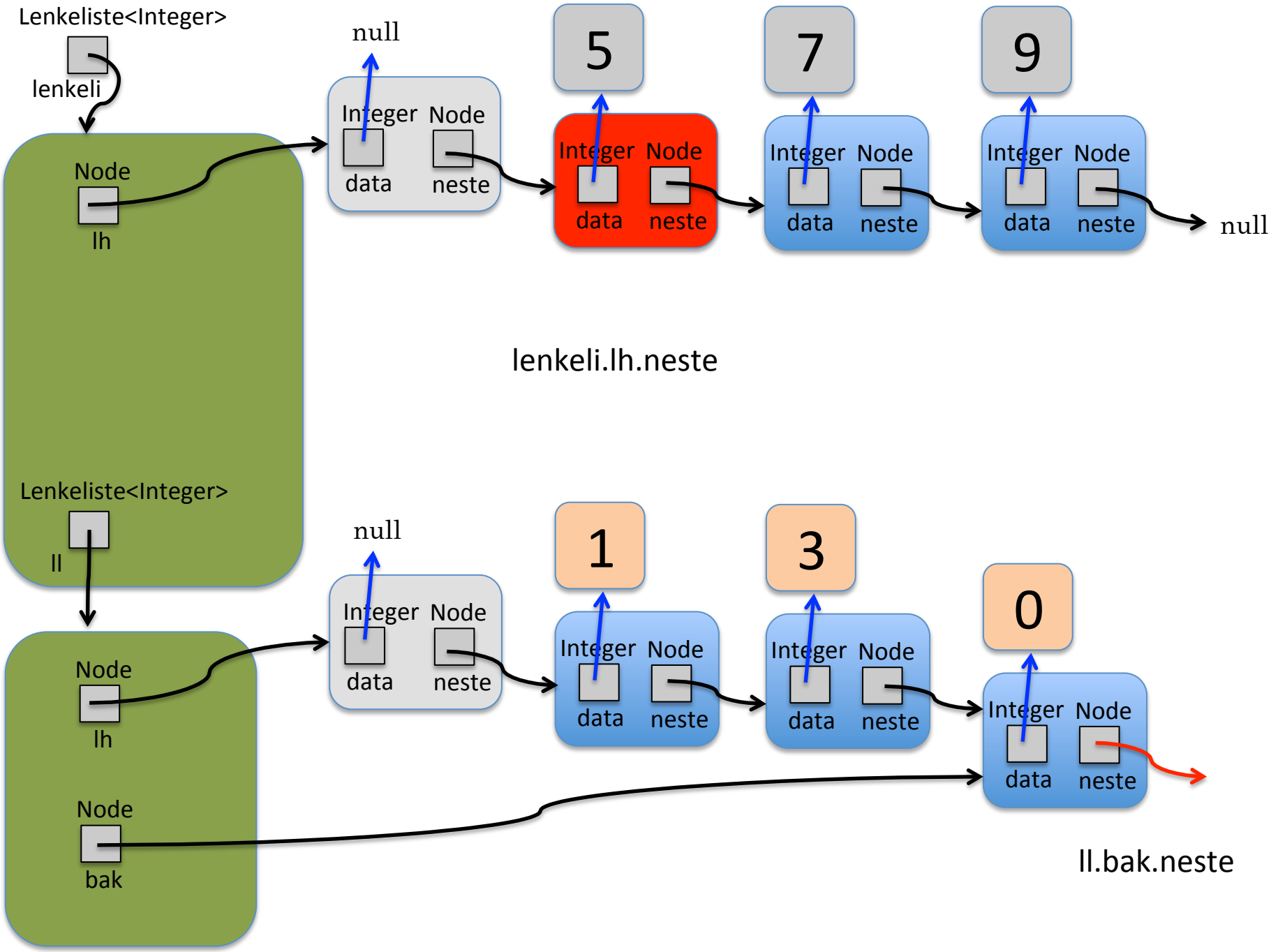

Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer
Node
data
neste
1
3
0

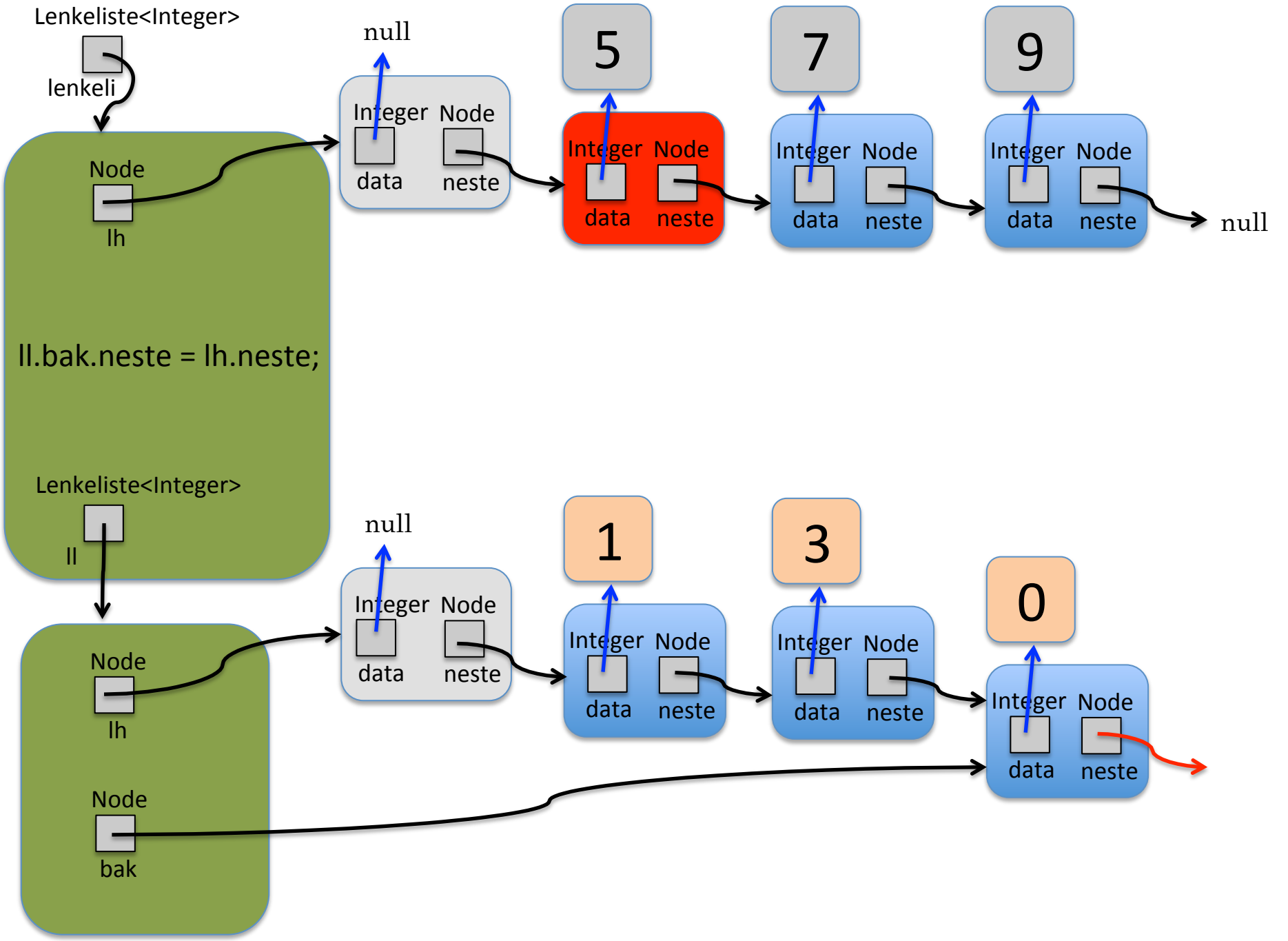

Lenkeliste<Integer>
lenkeli
Node
lh
Node
bak
null
Integer
Node
data
neste
5
7
9
null

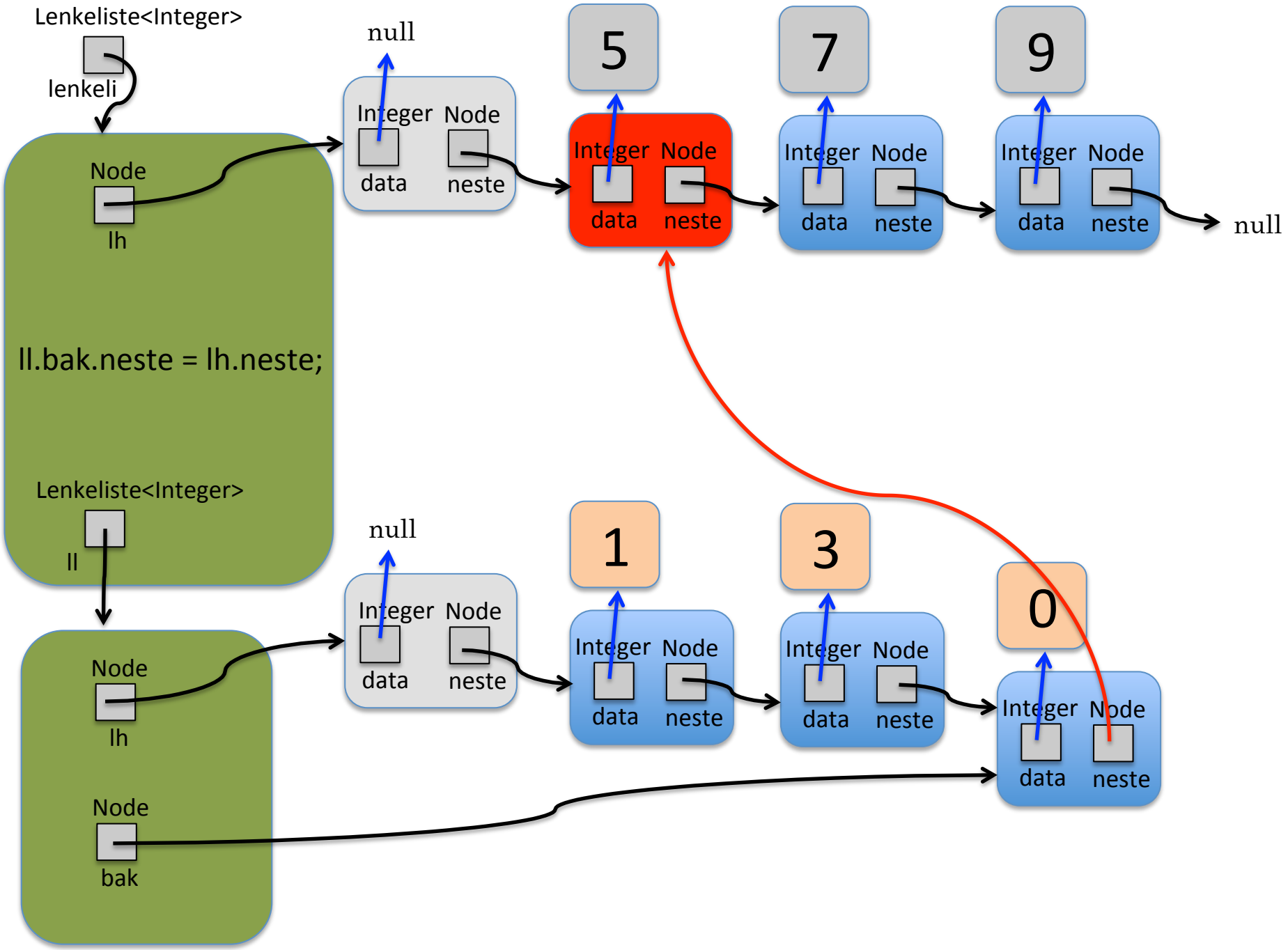

Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer
Node
data
neste
1
3
0

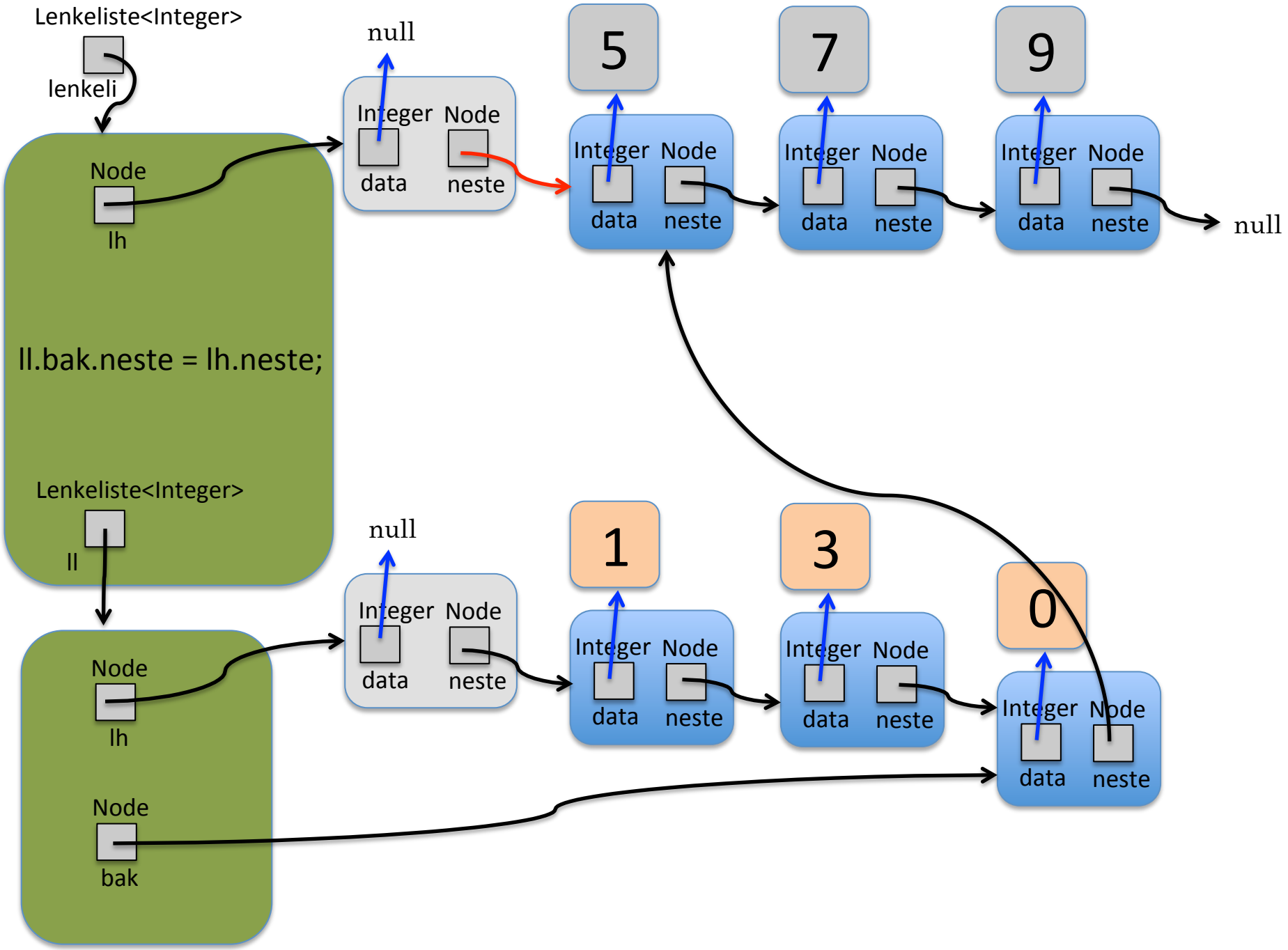

Lenkeliste<Integer>
lenkeli
Node
lh
Node
bak
null
Integer
Node
data
neste
5
7
9
null

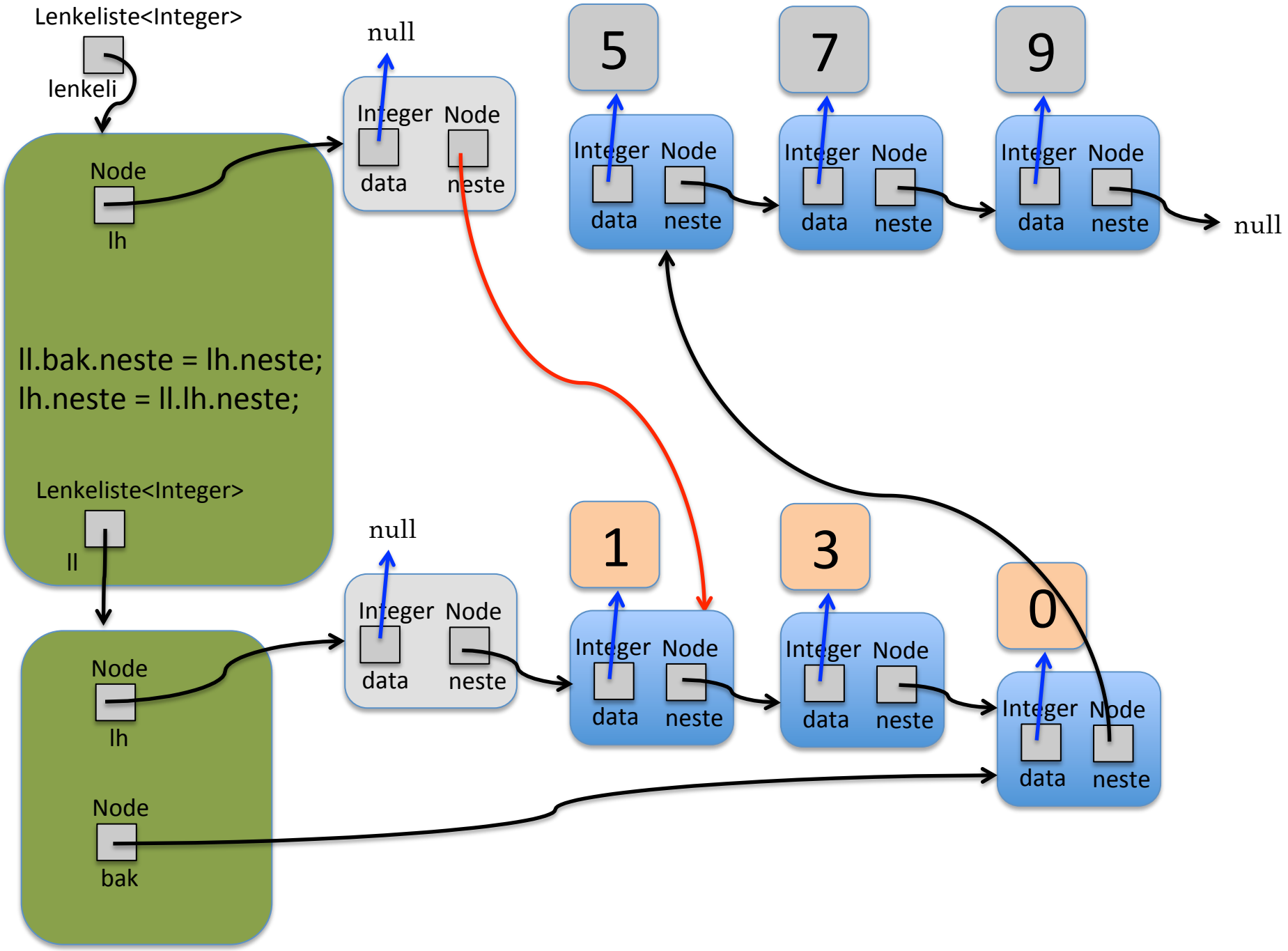

Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer
Node
data
neste
1
3
0
ll.bak.neste

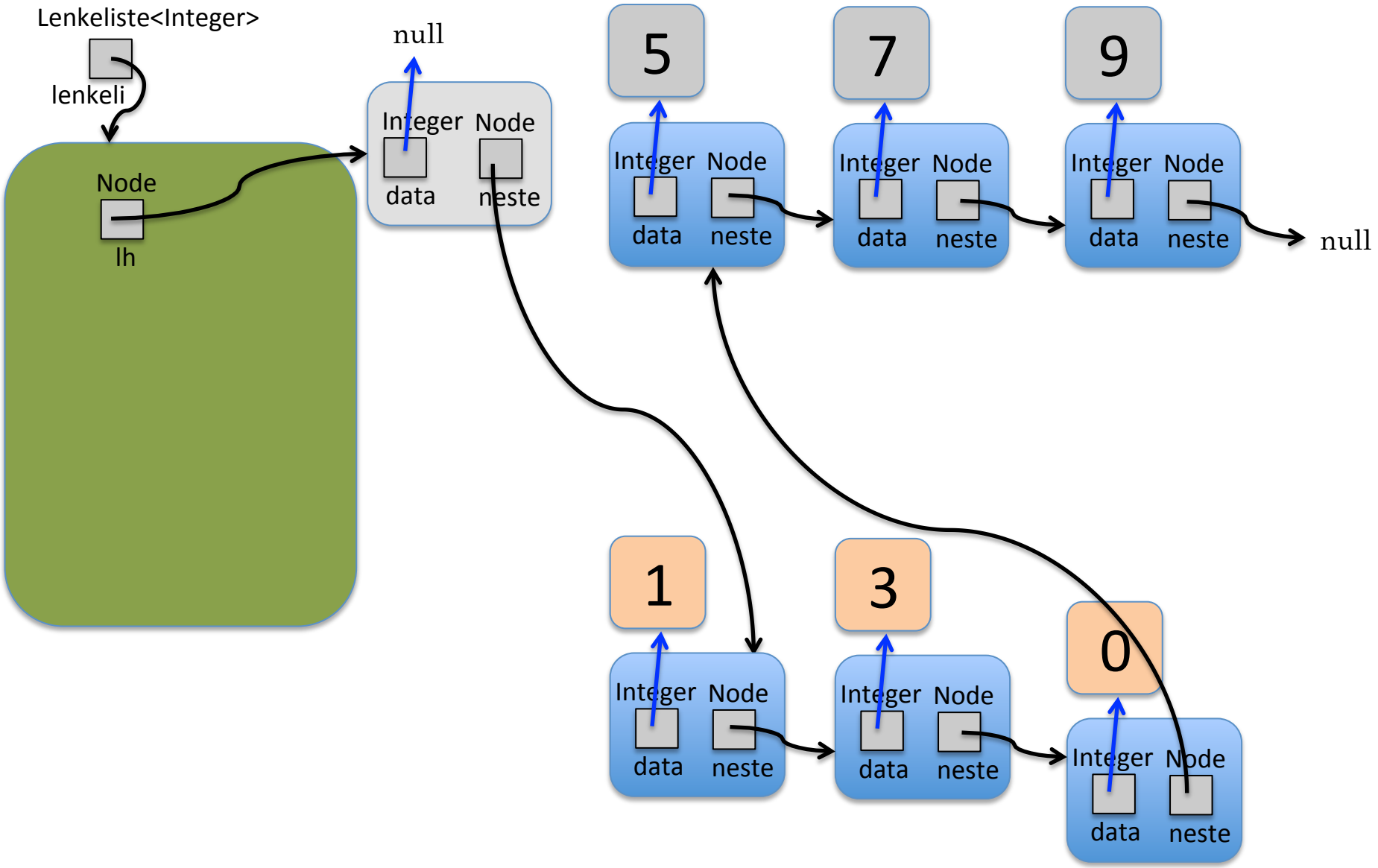

Lenkeliste<Integer>
lenkeli
Node
lh
null
Integer
Node
data
neste
5
7
9
null
lenkeli.lh.neste
Lenkeliste<Integer>
ll
Node
lh
Node
bak
1
3
0
ll.bak.neste

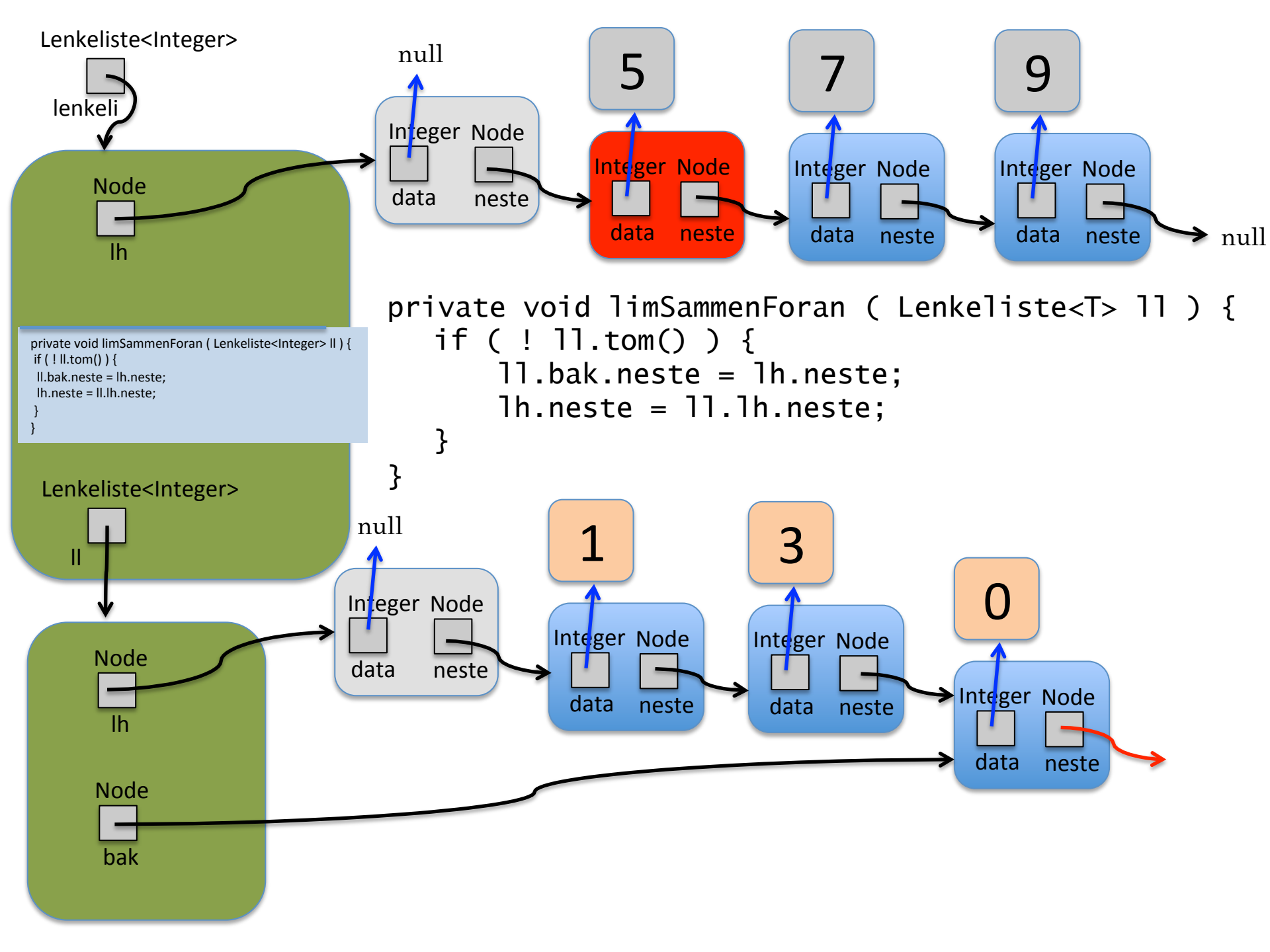

Lenkeliste<Integer>
lenkeli
Node
lh
ll.bak.neste = lh.neste;
Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer Node
data neste
5
7
9
null
1
3
0

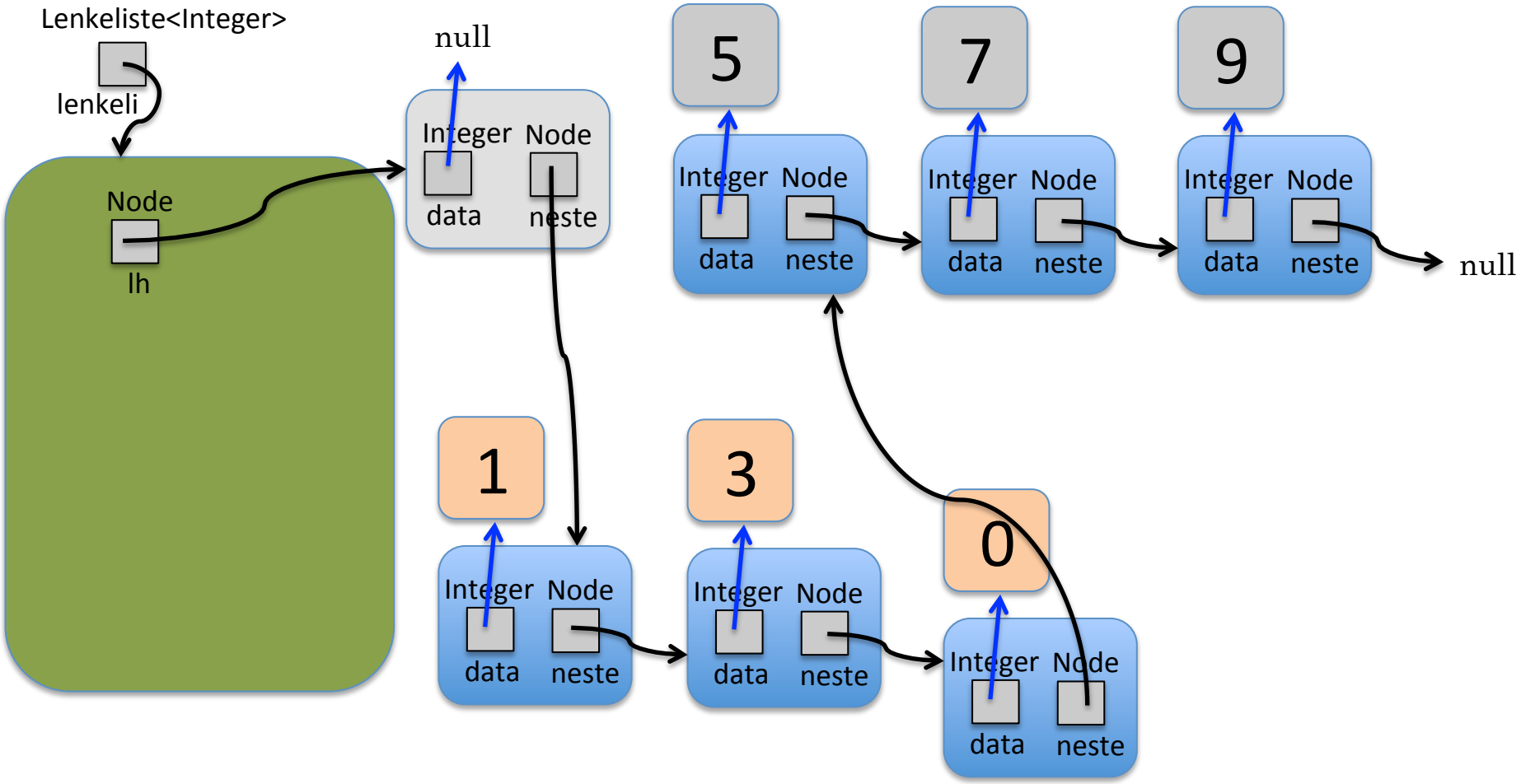

Lenkeliste<Integer>
lenkeli
Node
lh
ll.bak.neste = lh.neste;
Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer Node
data neste
5
7
9
null
1
3
0


Lenkeliste<Integer>
lenkeli
Node
lh
ll.bak.neste = lh.neste;
Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer Node
data neste
5
7
9
null
1
3
0


Lenkeliste<Integer>
lenkeli
Node
lh
ll.bak.neste = lh.neste;
lh.neste = ll.lh.neste;
Lenkeliste<Integer>
ll
Node
lh
Node
bak
null
Integer Node
data neste
5
7
9
null
1
3
0

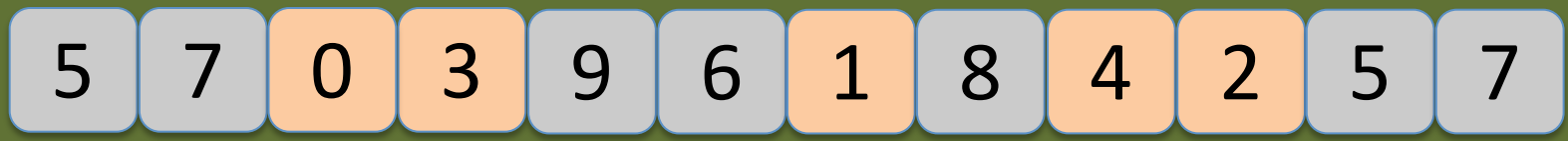

Lenkeliste<Integer>
lenkeli
Node
lh
null
Integer
Node
data
neste
5
7
9
null
1
3
0


Lenkeliste<Integer>
lenkeli
Node
lh
null
Integer Node
data neste
5
7
9
null
private void limSammenForan (Lenkeliste<Integer> ll) {
if (! ll.tom()) {
ll.bak.neste = lh.neste;
lh.neste = ll.lh.neste;
}
}
Lenkeliste<Integer>
ll
Node
lh
Node
bak
1
3
0
private void limSammenForan (Lenkeliste<T> ll) {
if (! ll.tom()) {
ll.bak.neste = lh.neste;
lh.neste = ll.lh.neste;
}
}

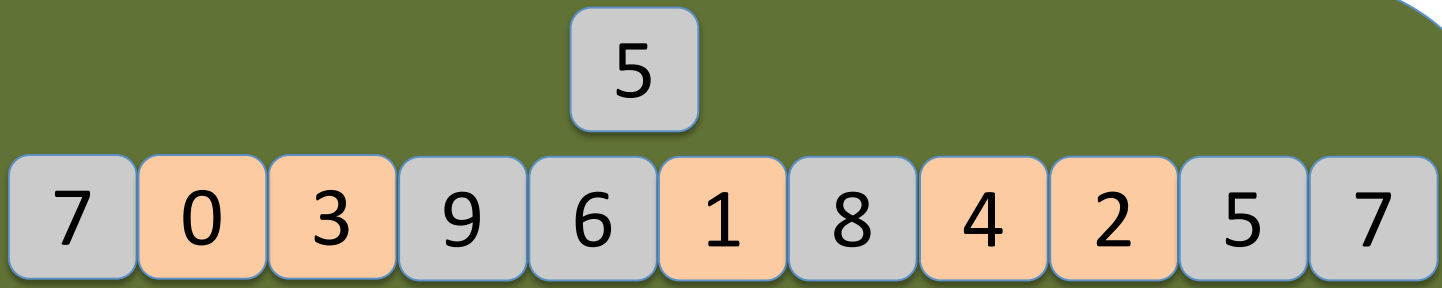



```
private void limSammenForan ( Lenkeliste<T> ll ) {
   if ( ! ll.tom() ) {
       ll.bak.neste = lh.neste;
       lh.neste = ll.lh.neste;
   }
}
```

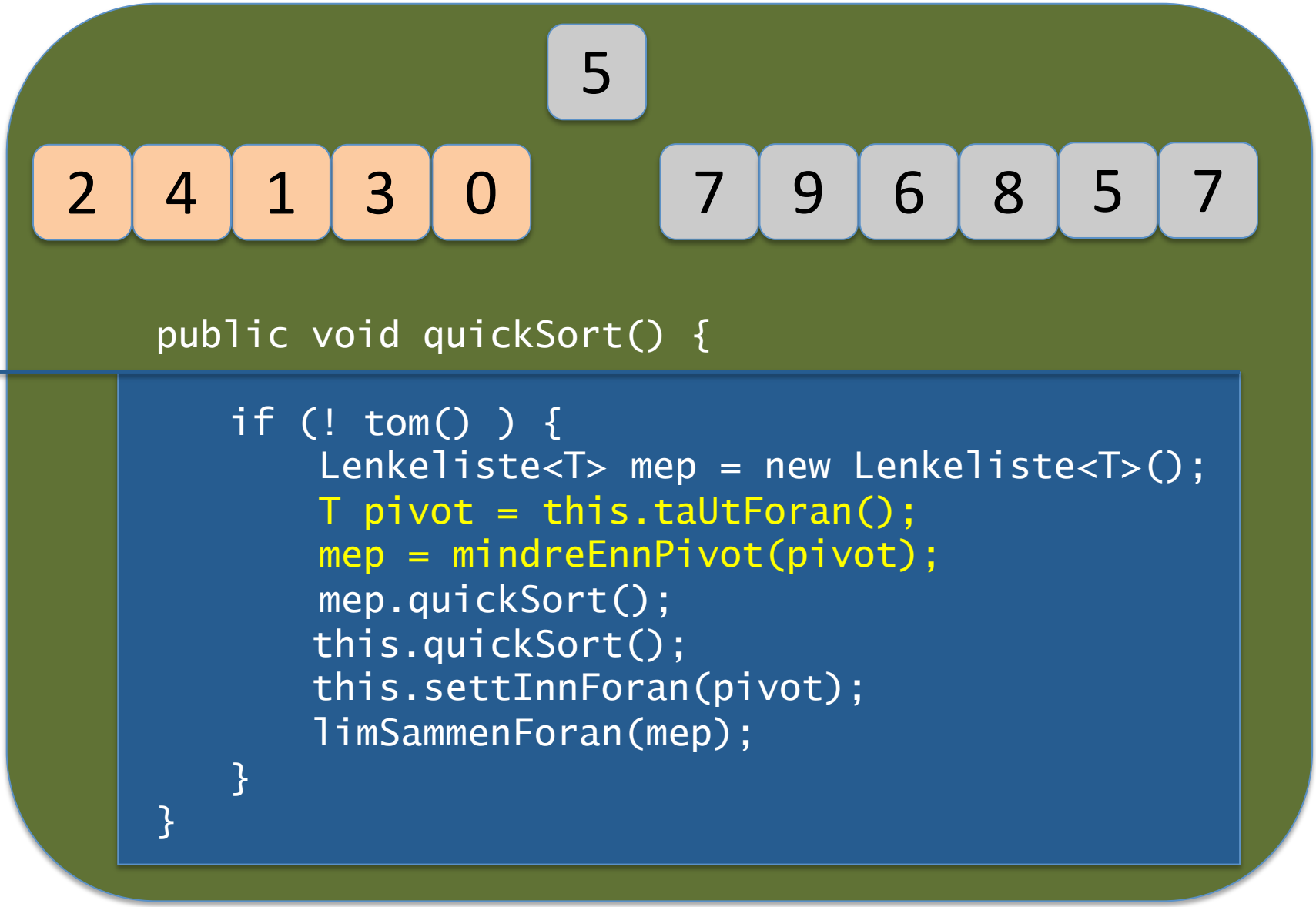
```
public void quickSort() {
    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort(); this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```

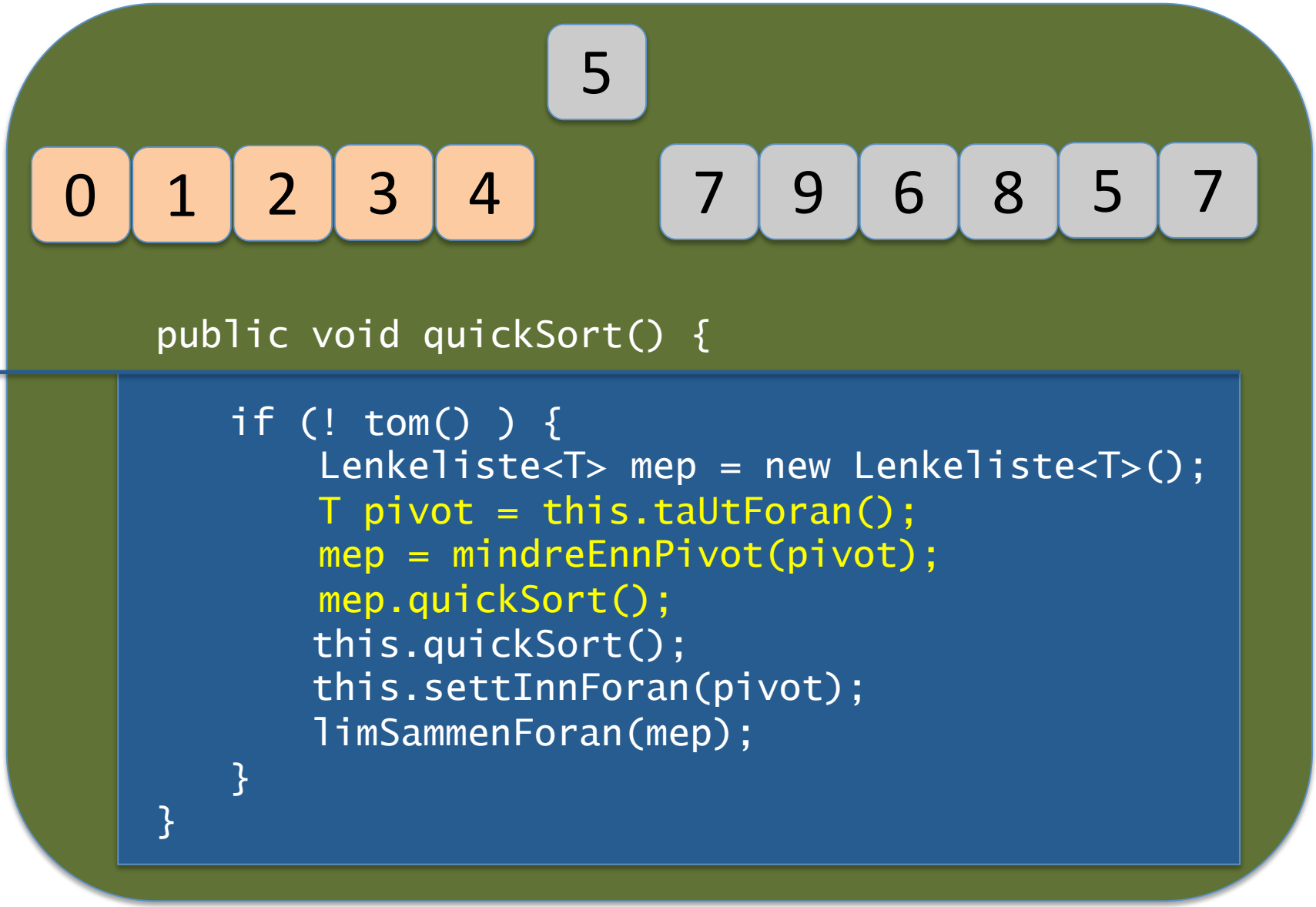


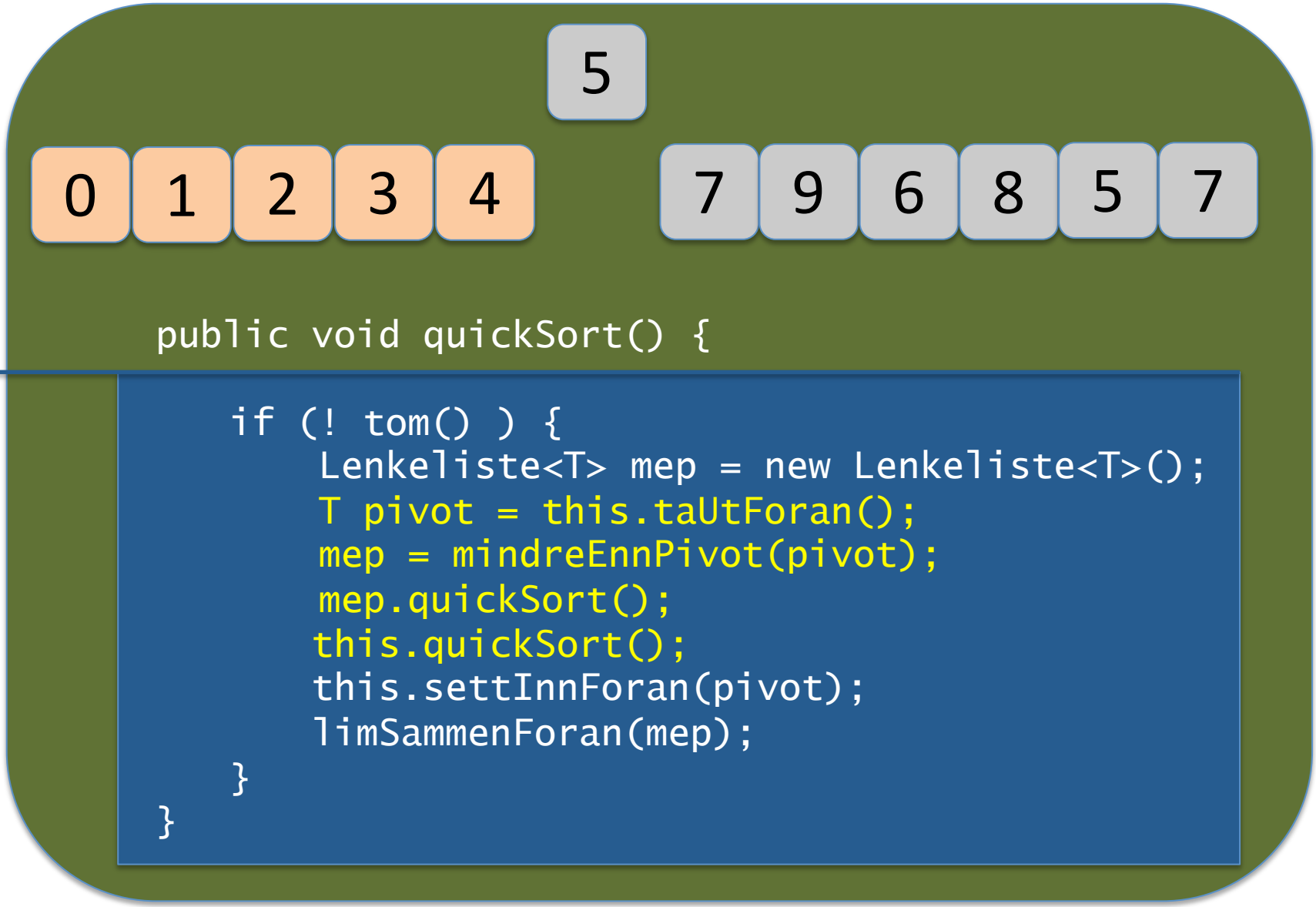

```
public void quickSort() {
    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort(); this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```

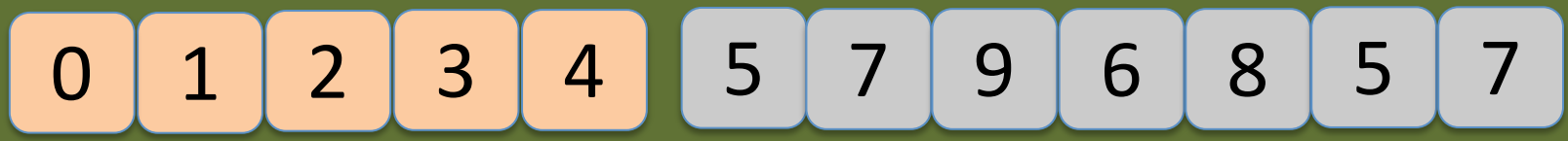



```
public void quickSort() {
    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort(); this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```

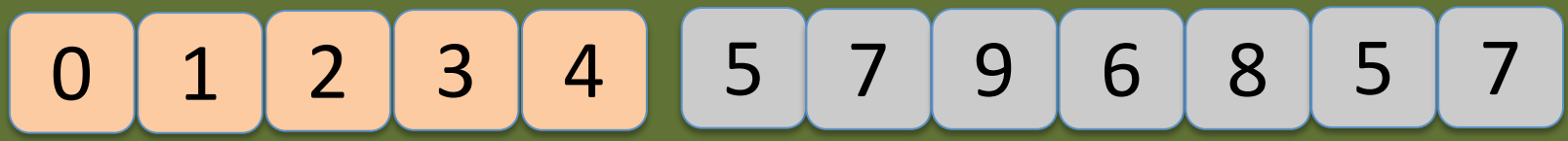

5
2 4 1 3 0
7 9 6 8 5 7
public void quickSort() {
if (! tom()) {
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
limSammenForan(mep);
}
}


5
0 1 2 3 4
7 9 6 8 5 7
public void quickSort() {
if (! tom()) {
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
limSammenForan(mep);
}
}

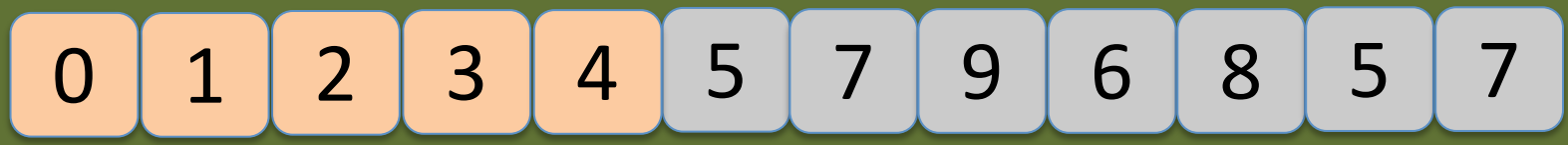

5
0 1 2 3 4
7 9 6 8 5 7
public void quickSort() {
if (! tom()) {
Lenkeliste<T> mep = new Lenkeliste<T>();
T pivot = this.taUtForan();
mep = mindreEnnPivot(pivot);
mep.quickSort();
this.quickSort();
this.settInnForan(pivot);
limSammenForan(mep);
}
}



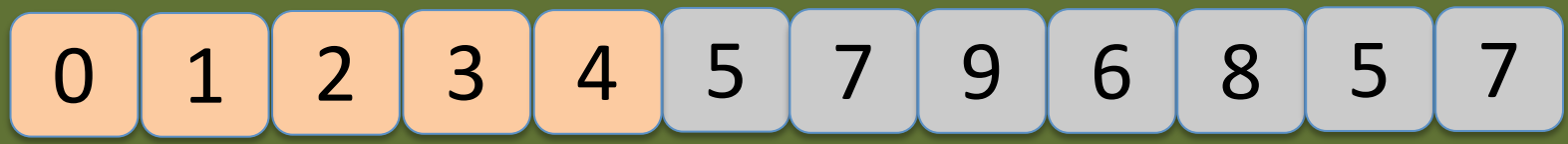

```
public void quickSort() {

    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort();
        this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```




```
public void quickSort() {

    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort();
        this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```




```
public void quickSort() {

    if (! tom() ) {
        Lenkeliste<T> mep = new Lenkeliste<T>();
        T pivot = this.taUtForan();
        mep = mindreEnnPivot(pivot);
        mep.quickSort();
        this.quickSort();
        this.settInnForan(pivot);
        limSammenForan(mep);
    }
}
```



et voilà !