Linked data structures
append and friends

Johan Montelius

KTH

HT22

LinkedList

```
public class LinkedList {
  int head;
  LinkedList tail;

  public LinkedList(int item, LinkedList list) {
    head = item;
    tail = list;
  }

  :
  :
}
```

This data structure is also refereddd to as a *cons cell*.

LinkedList

```
public class LinkedList {
  int head;
  LinkedList tail;

  public LinkedList(int item, LinkedList list) {
    head = item;
    tail = list;
  }

  :
  :
}
```

This data structure is also refereded to as a *cons cell*.

... or

```
public class LinkedList {
   Node head;
   int size;
    :
 }
```

```
public class Node {
      int value;
      Node tail;
}
```

We have a data structure that represents the empty list.
The data structure will always refer to the head of the linked list.

... or

```
public class LinkedList {
   Node head;
   int size;
    :
 }
```

```
public class Node {
      int value;
      Node tail;
}
```

We have a data structure that represents the empty list.

The data structure will always refer to the head of the linked list.

... or

```
public class LinkedList {
   Node head;
   int size;
    :
  }
```

```
public class Node {
      int value;
      Node tail;
}
```

We have a data structure that represents the empty list.
The data structure will always refer to the head of the linked list.

LinkedList - search

```
public boolean search(int key) {
  LinkedList nxt = this;
  while (nxt != null) {
    if (nxt.head == key)
      return true;
    nxt = nxt.tail;
  }
  return false;
}
```

LinkedList - add

```
public LinkedList add(int key) {
  return new LinkedList(key, this);
}
```

LinkeList - delete

```
public LinkedList delete(int key) {
  LinkedList nxt = this;
  LinkedList prev = null;
  while (nxt != null) {
    if (nxt.head == key) {
      prev.tail = nxt.tail;
      break;
    }
    prev = nxt;
    nxt = nxt.tail;
  }
  if( this == nxt)
    return null;
  else
    return this;
}
```

LinkedList - append

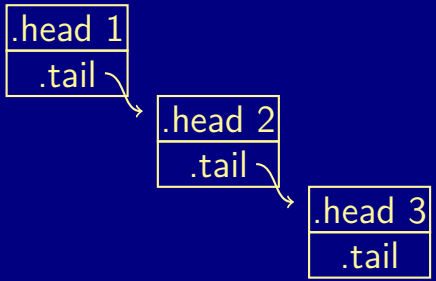
```
public void append(LinkedList b) {
  LinkedList nxt = this;
  while (nxt.tail != null) {
    nxt = nxt.tail;
  }
  nxt.tail = b;
}
```

LinkedList - reverse

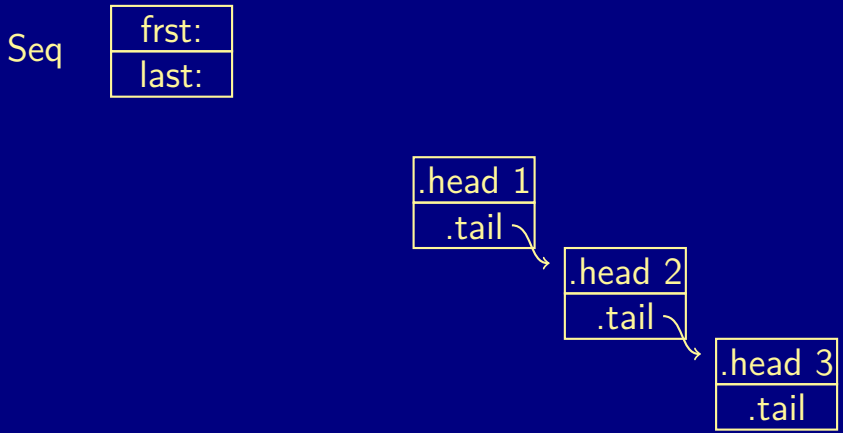
```
public LinkedList reverse() {

    LinkedList nxt = this;
    LinkedList prev = null;
    while ( nxt != null) {
        LinkedList tmp = nxt.tail;
        nxt.tail = prev;
        prev = nxt;
        nxt = tmp;
    }
    return prev;
}
```

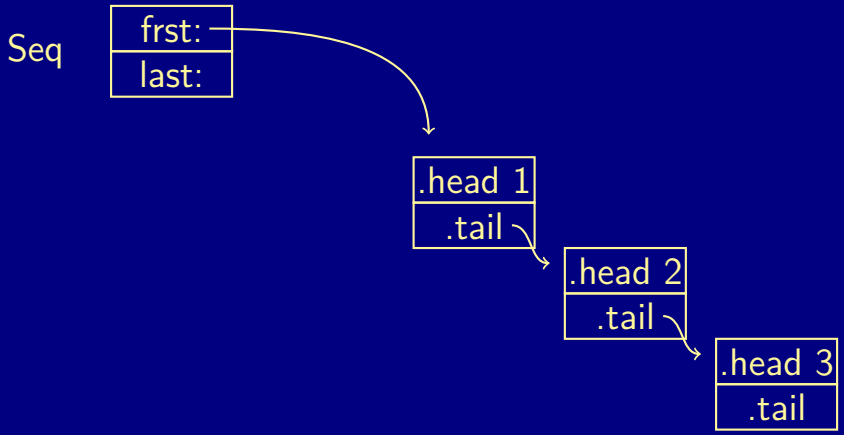
an alternative



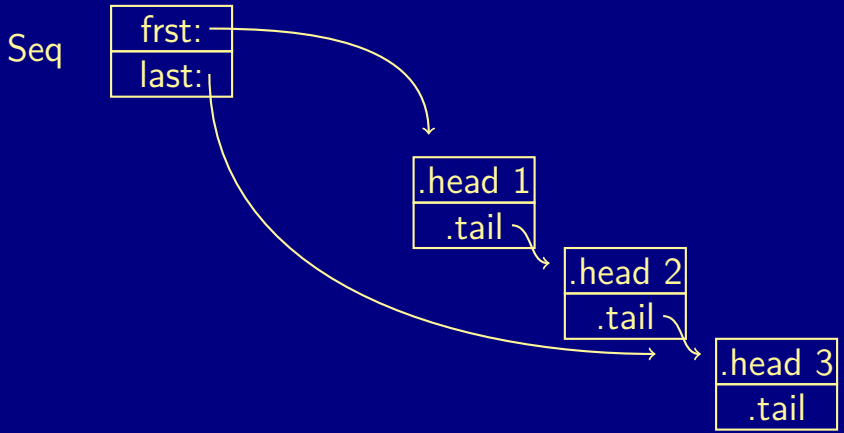
an alternative



an alternative



an alternative



Sequence

```
public class Seq {
   LinkedList first;
   LinkedList last;
    :
    :
  }
```

Sequence

```
public class Seq {
   LinkedList first;
   LinkedList last;
    :
    :
  }
```

```
  :
public void append(Seq b) {
  last.tail = b.frst;
  last = b.last;
}
  :
```