Thomas van Lingen

2074690

Maak een gekoppelde lijst waarin je NAW –gegevens opslaat.
Maak de Link–klasse en LinkedList–klasse.

```
public class NAWLink {
    public NAW linkItem;
    public NAWLink nextNAWLink;
    public NAWLink(NAW linkItem) {
        this.linkItem = linkItem;
    }
    public void setLink(NAWLink link){
        this.nextNAWLink = link;
    }
    public boolean hasLink(){
        return(this.nextNAWLink != null);
    }
}

public class NAWLinkedList {
    public NAWLink firstLink;
    public NAWLinkedList(NAWLink firstLink){
        this.firstLink = firstLink;
    }
}
```

Maak een methode om NAW–gegevens toe te voegen aan het begin van de gekoppelde lijst.

```
public void addLink(NAWLink linkToAdd){
    linkToAdd.setLink(this.firstLink);
    this.firstLink = linkToAdd;
}
```

Maak een methode om een NAW–instantie te zoeken.

```
public NAWLink searchForNAW(String name, int address, String cityOfResidence){
    for(NAWLink iterationLink = this.firstLink; iterationLink != null;
iterationLink = iterationLink.nextNAWLink){
        if(iterationLink.linkItem.compareTo(name, address, cityOfResidence) == 0){
            return iterationLink;
        }
    }
    return null;
}
```

Maak een methode om alle NAW–gegevens te tonen.

Sorry voor de vreemde formatting, lange lijnen code plakken in libreoffice is geen success...

```
public void printContents(){
       int linkCount = 0;
       for(NAWLink currentLink = this.firstLink; currentLink != null; currentLink =
currentLink.nextNAWLink, linkCount++){
             System.out.format("Link %d: %s - %s:%d\n", linkCount,
             currentLink.linkItem.getName(), currentLink.linkItem.getCityOfResidence(),
             currentLink.linkItem.getAddress());
       }
}
```

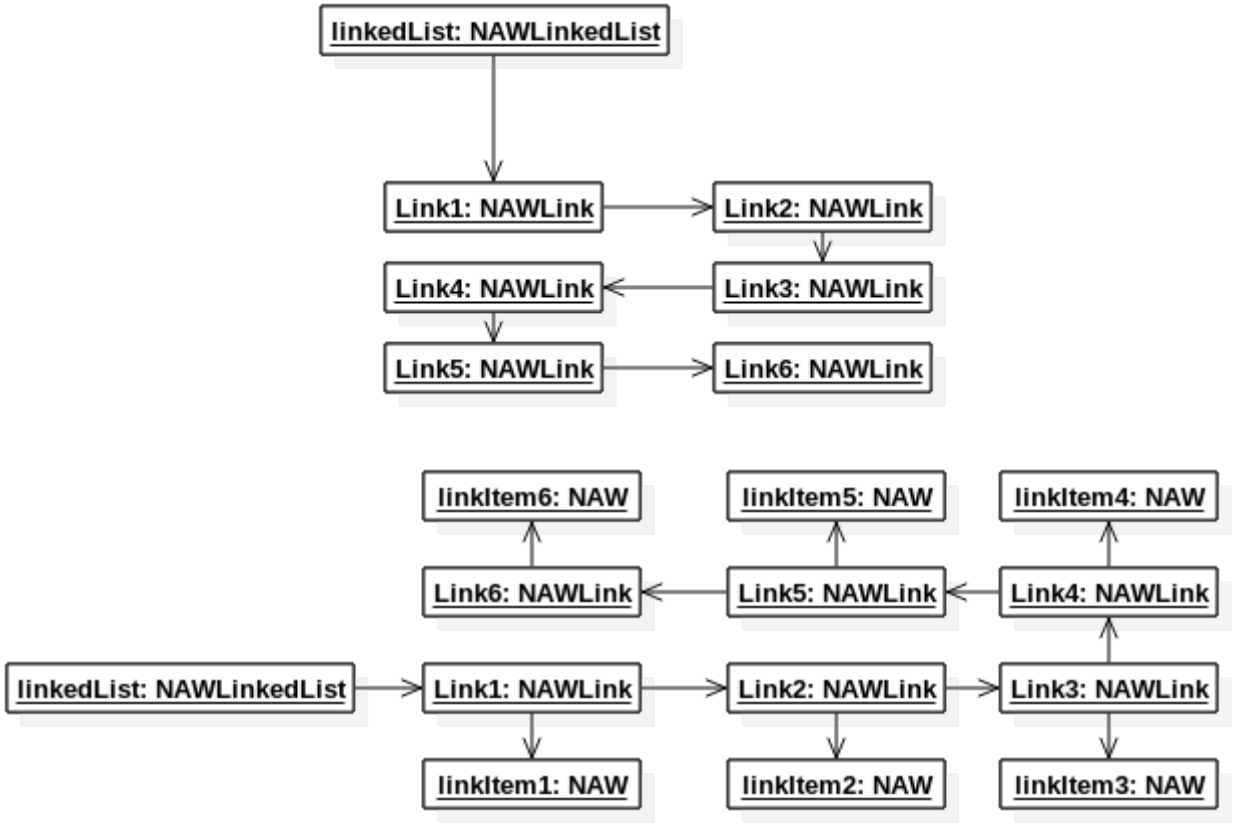
Maak een methode om bepaalde NAW – gegevens te verwijderen.

```
public void removeNAWLinks(String name, int address, String cityOfResidence){

    int currentLinkIndex = 0, removedItems = 0;
       for(NAWLink currentLink = this.firstLink; currentLink != null; currentLink =
       currentLink.nextNAWLink, currentLinkIndex++){
         if(currentLink.linkItem.compareTo(name, address, cityOfResidence) == 0){
             this.removeNAWFromList(currentLinkIndex);
             currentLinkIndex -= 1;
              //Since we're removing a link, naturally our currenty link must be
              decremented as well
             removedItems++;
         }
    }
    System.out.println(removedItems + " NAWLinks removed");
}

public void removeNAWFromList(int nestedIndex){

    if(nestedIndex == 0){
       //This is a special case, since we only need to change the linkedlist's reference!
         this.firstLink = this.firstLink.nextNAWLink;
    } else{
         //We're dealing with a regular case, link the previous link to the next one
         NAWLink previousLink = this.getLink(nestedIndex - 1);
         NAWLink linkToRemove = this.getLink(nestedIndex);
         if(previousLink != null && linkToRemove != null) {
             previousLink.setLink(linkToRemove.nextNAWLink);
         }
    }
}

public NAWLink getLink(int indexToGet){

    int linkCount = 0;
    for(NAWLink currentLink = this.firstLink; currentLink != null; currentLink =
currentLink.nextNAWLink, linkCount++){
         if(linkCount == indexToGet){
             return currentLink;
         }
    }
    return null;
}
```

Nu we toch een getLink method hebben kan dit ook anders(Wellicht leesbaarder)

```
public void removeNAWLinks2(String name, int address, String cityOfResidence){

    int removedItems = 0;
    for(int currentLinkIndex = 0; this.getLink(currentLinkIndex) != null;
currentLinkIndex++){
         if(this.getLink(currentLinkIndex).linkItem.compareTo(name, address,
cityOfResidence) == 0){
             this.removeNAWFromList(currentLinkIndex);
             currentLinkIndex -= 1;
              //Since we're removing a link, naturally our currently link must be
              decreased as well
             removedItems++;
         }
    }
    System.out.println(removedItems + " NAWLinks removed");
}
```

Objectdiagrammen



Maak en method waarmee je een instantie uit een gekoppelde lijst kunt opvragen aan de hand van de index

```
public IntLink getIntLink(int index){
    int linkCount = 0;

    for(IntLink currentLink = this.firstLink; currentLink != null; currentLink =
currentLink.nextLink, linkCount++){
        if(linkCount == index){
            return currentLink;
        }
    }

    return null;
}
```

Maak een methode waarmee je een instantie kunt veranderen aan de hand van de index

```
public void setLink(int index, int value){
    IntLink toSet = this.getIntLink(index);

    if(toSet != null){
        toSet.value = value;
    }
}
```

Herschrijf het bubble-sort-algoritme zodat je met behulp van de getAt-methode een gekoppelde lijst kunt sorteren.

```
public void bubbleSort(){
    //Swap whilst incrementing the endIndex until it approaches the length
    for(int endIndex = 0; endIndex < this.length -1; endIndex++){
        //One swap routine(From end to endIndex)
        for(int currentIndex = this.length -1; currentIndex > endIndex; currentIndex--){
            if(this.getIntLink(currentIndex).value < this.getIntLink(currentIndex-
1).value){
                this.swapLinks(currentIndex, currentIndex-1);
            }
        }
    }
}

private void swapLinks(int swapIndex1, int swapIndex2){
    int copy = this.getIntLink(swapIndex1).value;
    this.getIntLink(swapIndex1).value = this.getIntLink(swapIndex2).value;
    this.getIntLink(swapIndex2).value = copy;
}

public IntLink getIntLink(int index){
    int linkCount = 0;

    for(IntLink currentLink = this.firstLink; currentLink != null; currentLink = cur-
rentLink.nextLink, linkCount++){
        if(linkCount == index){
            return currentLink;
        }
    }

    return null;
}
```

Geef in de big-O Notatie aan van welke orde dit algoritme is. Verklaar je antwoord.
O(N^3)
We hebben te maken met 3 loops:
- **2 loops die van het bubblesort algorithme zijn**
  - **1 voor elke swaproutine zelf**
  - **1 voor alle swaproutines**
- **3e loop komt van de getIntLink functie**

Vul een gekoppelde lijst met respectievelijk 100, 1000, 10.000 en 100.000 random – waarden. Onderzoekde sorteersnelheid. Komt deze overeen met de orde van het algoritme? Verklaar je antwoord.

Done with sorting 100 ints!
Time elapsed in ms: 1,901429
Done with sorting 1000 ints!
Time elapsed in ms: 1434,329212
Done with sorting 10000 ints
Time elapsed in ms: 1409489,272178

Voor O(N^3) zou betekenen dat als we de lengte vertienvoudigen, de tijdsduur 10^3, zou moeten verduizendvoudigen
Dit is ook ongeveer het geval voor zowel 100, 1000 als 1000, 10000

Implementeer een efficiënt bubble sort algoritme in een tweerichtingslijst

```
public void efficientBubbleSort(){
    for(int endIndex = this.length-1; endIndex > 0; endIndex--){
        //One swap routine(From endIndex to 0)
        IntLink currentLink = this.firstLink;
        for(int currentIndex = 0; currentIndex < endIndex; currentIndex++){
            if(currentLink.value > currentLink.nextLink.value){
                this.efficientSwapNext(currentLink);
            }
            currentLink = currentLink.nextLink;
        }
    }
}

public void efficientSwapNext(IntLink toSwap){
    int copy = toSwap.value;
    toSwap.value = toSwap.nextLink.value;
    toSwap.nextLink.value = copy;
}
```

Geef in de big–O Notatie aan van welke orde dit algoritme is. Verklaar je antwoord
O(N^2)
Want hier komen maar 2 loops in voor:
1 vooralle routines
1 voor routine

Vul een gekoppelde lijst met respectievelijk 100, 1000, 10.000 en 100.000 random –waarden. Onderzoek de sorteersnelheid. Komt deze overeen met de orde van het algoritme? Verklaar je antwoord.

```
Done with sorting 100 ints!
Time elapsed in ms: 0,257348
Done with sorting 1000 ints!
Time elapsed in ms: 2,770165
Done with sorting 10000 ints!
Time elapsed in ms: 208,177708
```

Maak een gekoppelde lijst die handelingen als integers opslaat. De gekoppelde lijst dient de methodes `undo`, `redo` en `insert` te bezitten.
Code voor de IntLinkedList met historie:

```
public class IntLinkedListHistory extends IntLinkedList {
    public IntLink currentLink;
    public IntLinkedListHistory(IntLink firstLink){
        super(firstLink);
        this.currentLink = this.firstLink;
    }
    @Override
    public void addLink(IntLink linkToAdd){
        linkToAdd.setNextLink(this.currentLink);
        this.firstLink.setPreviousLink(linkToAdd);
        this.firstLink = linkToAdd;
        this.currentLink = this.firstLink;
        this.length++;
    }
    public void undo(){
        if(this.currentLink.nextLink != null){
            this.currentLink = this.currentLink.nextLink;
        } else {
            System.out.println("Can't undo stuff");
        }
    }
    public void redo(){
        if(this.currentLink != this.firstLink){
            this.currentLink = this.currentLink.previousLink;
        } else {
            System.out.println("Can't redo stuff");
        }
    }
    @Override
    public void printContents() {
        for(IntLink current = this.currentLink; current != null; current = current.nextLink){
            System.out.print(current.value + " - ");
        }
        System.out.println("");
    }
}
```