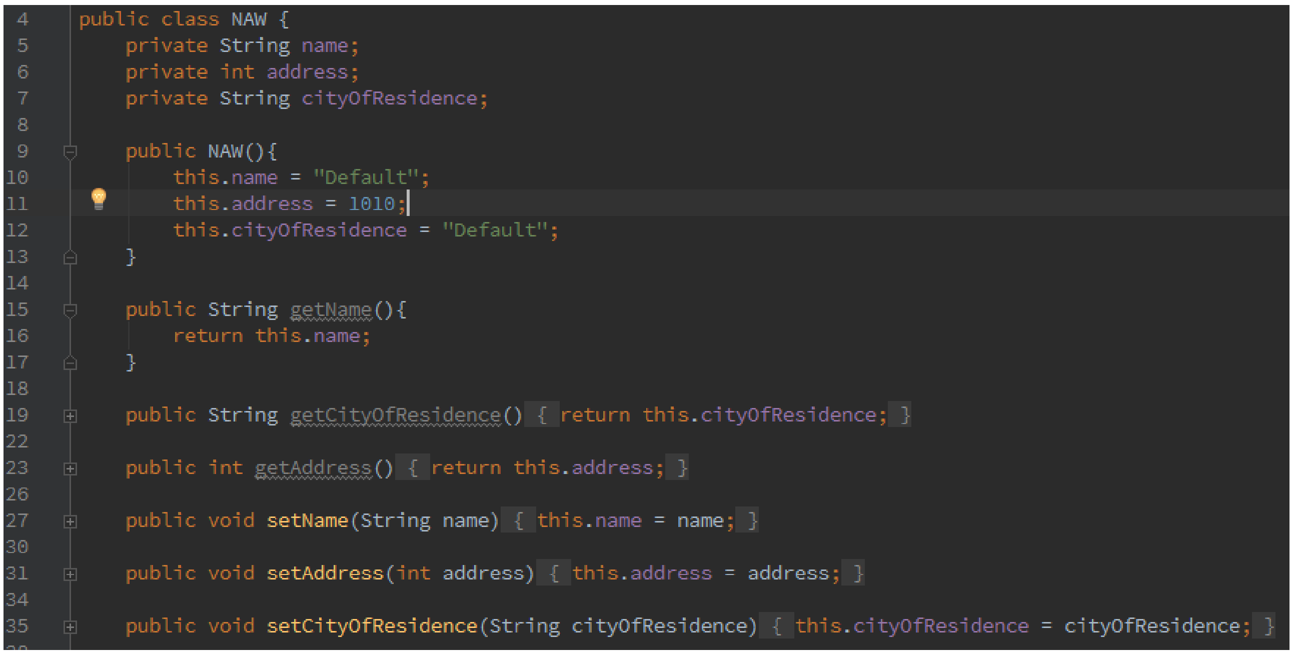
Deel 1

Thomas van Lingen

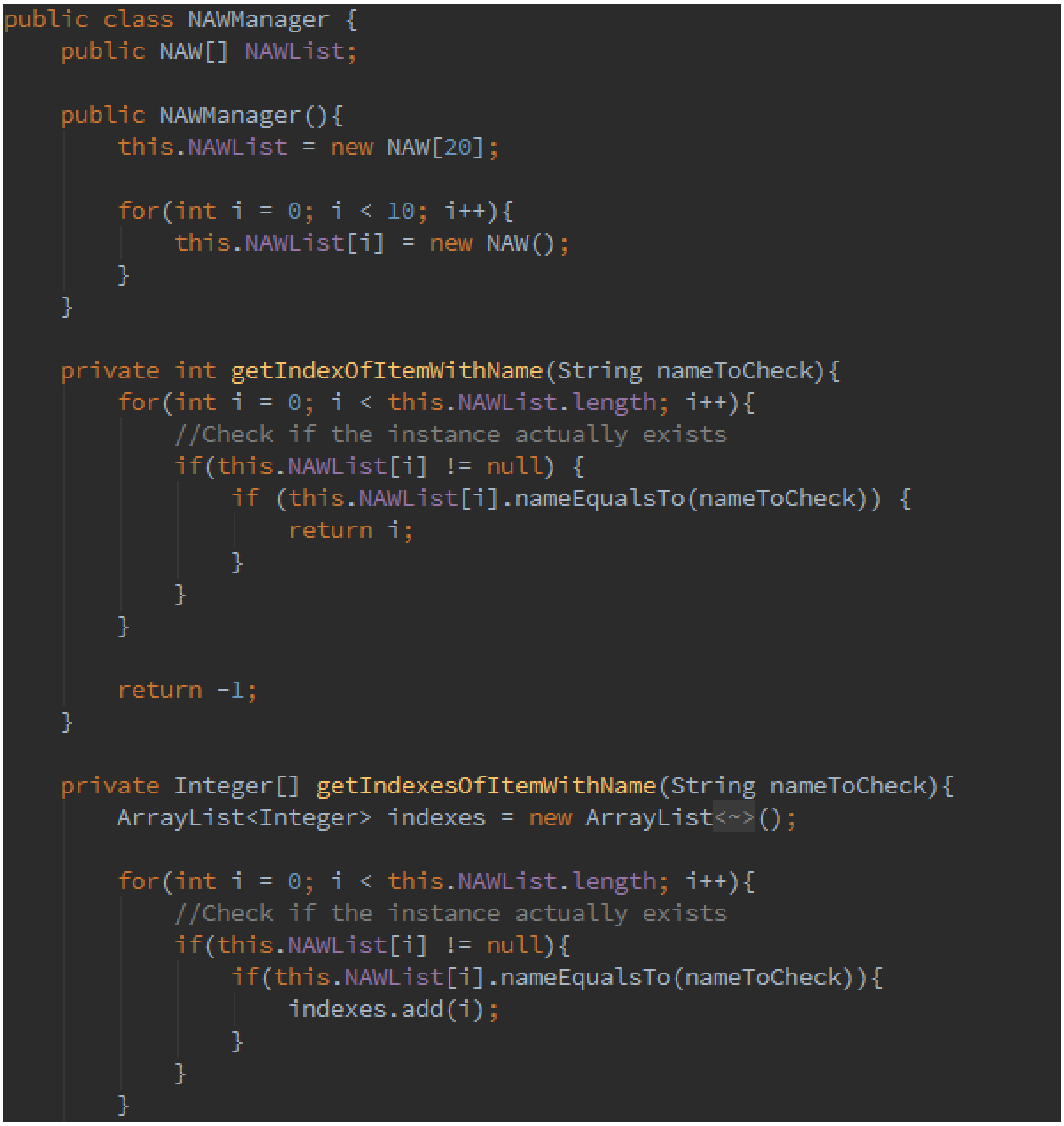
2974690

Opgave 1

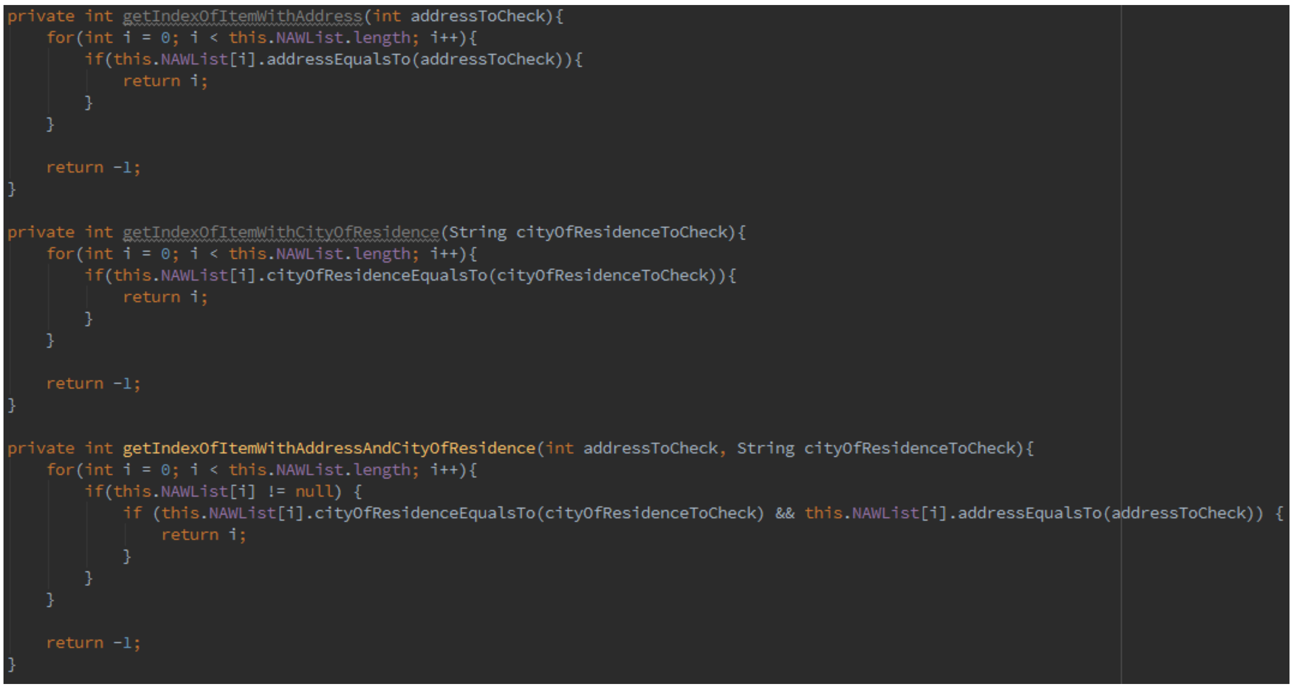


```java
public class NAW {
    private String name;
    private int address;
    private String cityOfResidence;

    public NAW(){
        this.name = "Default";
        this.address = 1010;
        this.cityOfResidence = "Default";
    }

    public String getName(){
        return this.name;
    }

    public String getCityOfResidence() { return this.cityOfResidence; }

    public int getAddress() { return this.address; }

    public void setName(String name) { this.name = name; }

    public void setAddress(int address) { this.address = address; }

    public void setCityOfResidence(String cityOfResidence) { this.cityOfResidence = cityOfResidence; }
```

Opgave 2

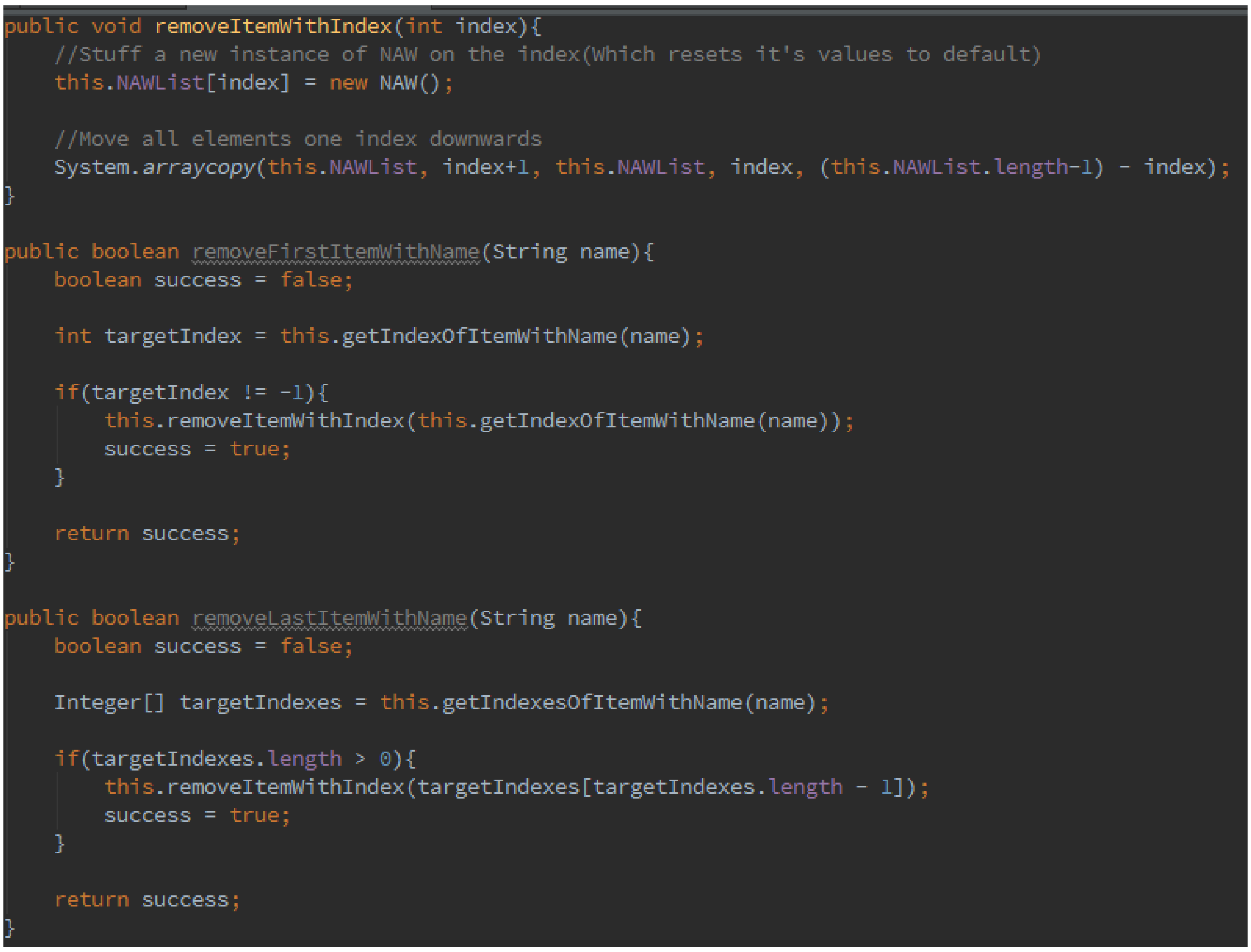


```java
public class NAWManager {
    public NAW[] NAWList;

    public NAWManager(){
        this.NAWList = new NAW[20];

        for(int i = 0; i < 10; i++){
            this.NAWList[i] = new NAW();
        }
    }

    private int getIndexOfItemWithName(String nameToCheck){
        for(int i = 0; i < this.NAWList.length; i++){
            //Check if the instance actually exists
            if(this.NAWList[i] != null) {
                if (this.NAWList[i].nameEqualsTo(nameToCheck)) {
                    return i;
                }
            }
        }

        return -1;
    }

    private Integer[] getIndexesOfItemWithName(String nameToCheck){
        ArrayList<Integer> indexes = new ArrayList<~>();

        for(int i = 0; i < this.NAWList.length; i++){
            //Check if the instance actually exists
            if(this.NAWList[i] != null){
                if(this.NAWList[i].nameEqualsTo(nameToCheck)){
                    indexes.add(i);
                }
            }
        }
```

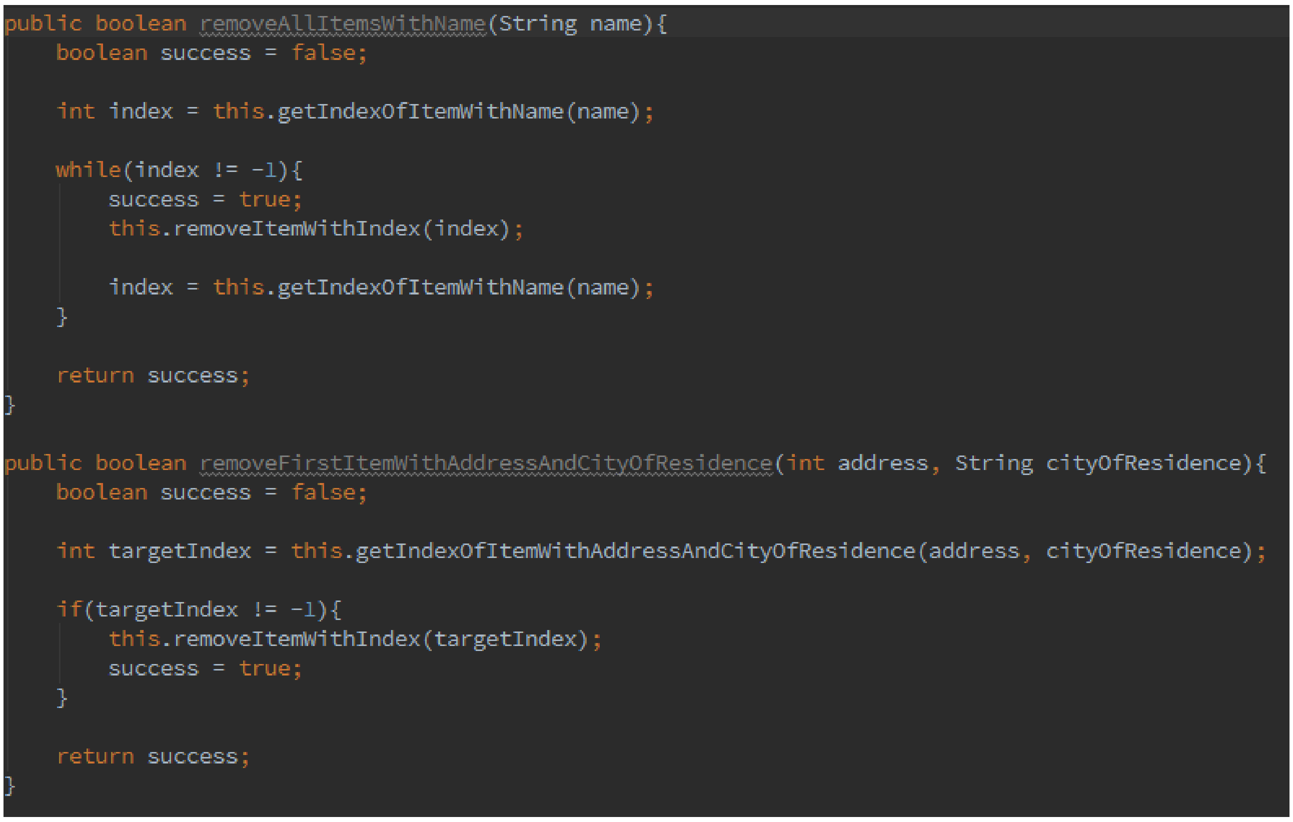


```java
private int getIndexOfItemWithAddress(int addressToCheck){
    for(int i = 0; i < this.NAWList.length; i++){
        if(this.NAWList[i].addressEqualsTo(addressToCheck)){
            return i;
        }
    }

    return -1;
}

private int getIndexOfItemWithCityOfResidence(String cityOfResidenceToCheck){
    for(int i = 0; i < this.NAWList.length; i++){
        if(this.NAWList[i].cityOfResidenceEqualsTo(cityOfResidenceToCheck)){
            return i;
        }
    }

    return -1;
}

private int getIndexOfItemWithAddressAndCityOfResidence(int addressToCheck, String cityOfResidenceToCheck){
    for(int i = 0; i < this.NAWList.length; i++){
        if(this.NAWList[i] != null) {
            if (this.NAWList[i].cityOfResidenceEqualsTo(cityOfResidenceToCheck) && this.NAWList[i].addressEqualsTo(addressToCheck)) {
                return i;
            }
        }
    }

    return -1;
}
```

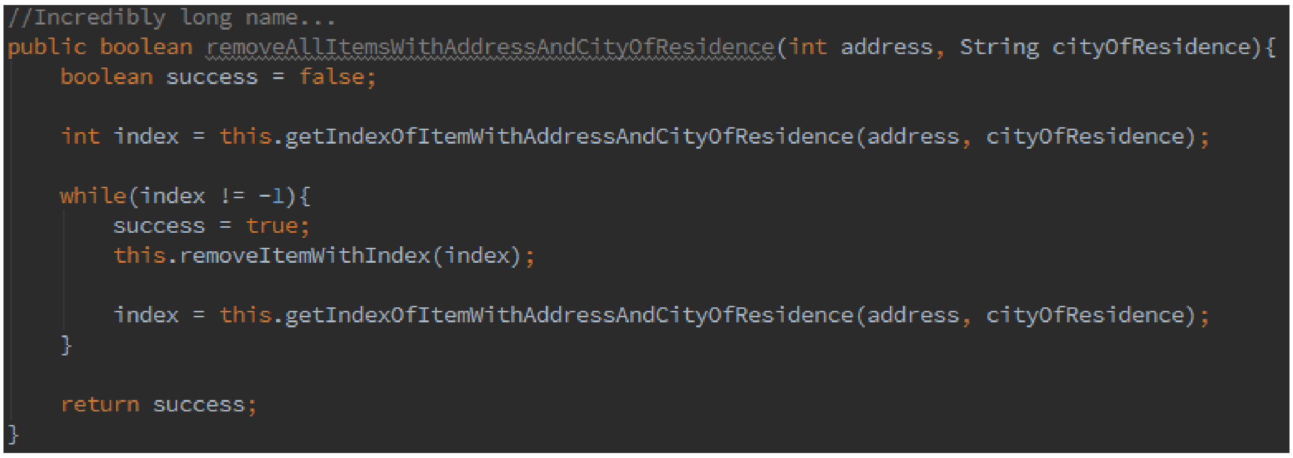
Opgave 4



```java
public void removeItemWithIndex(int index){
    //Stuff a new instance of NAW on the index(Which resets it's values to default)
    this.NAWList[index] = new NAW();

    //Move all elements one index downwards
    System.arraycopy(this.NAWList, index+1, this.NAWList, index, (this.NAWList.length-1) - index);
}

public boolean removeFirstItemWithName(String name){
    boolean success = false;

    int targetIndex = this.getIndexOfItemWithName(name);

    if(targetIndex != -1){
        this.removeItemWithIndex(this.getIndexOfItemWithName(name));
        success = true;
    }

    return success;
}

public boolean removeLastItemWithName(String name){
    boolean success = false;

    Integer[] targetIndexes = this.getIndexesOfItemWithName(name);

    if(targetIndexes.length > 0){
        this.removeItemWithIndex(targetIndexes[targetIndexes.length - 1]);
        success = true;
    }

    return success;
}
```

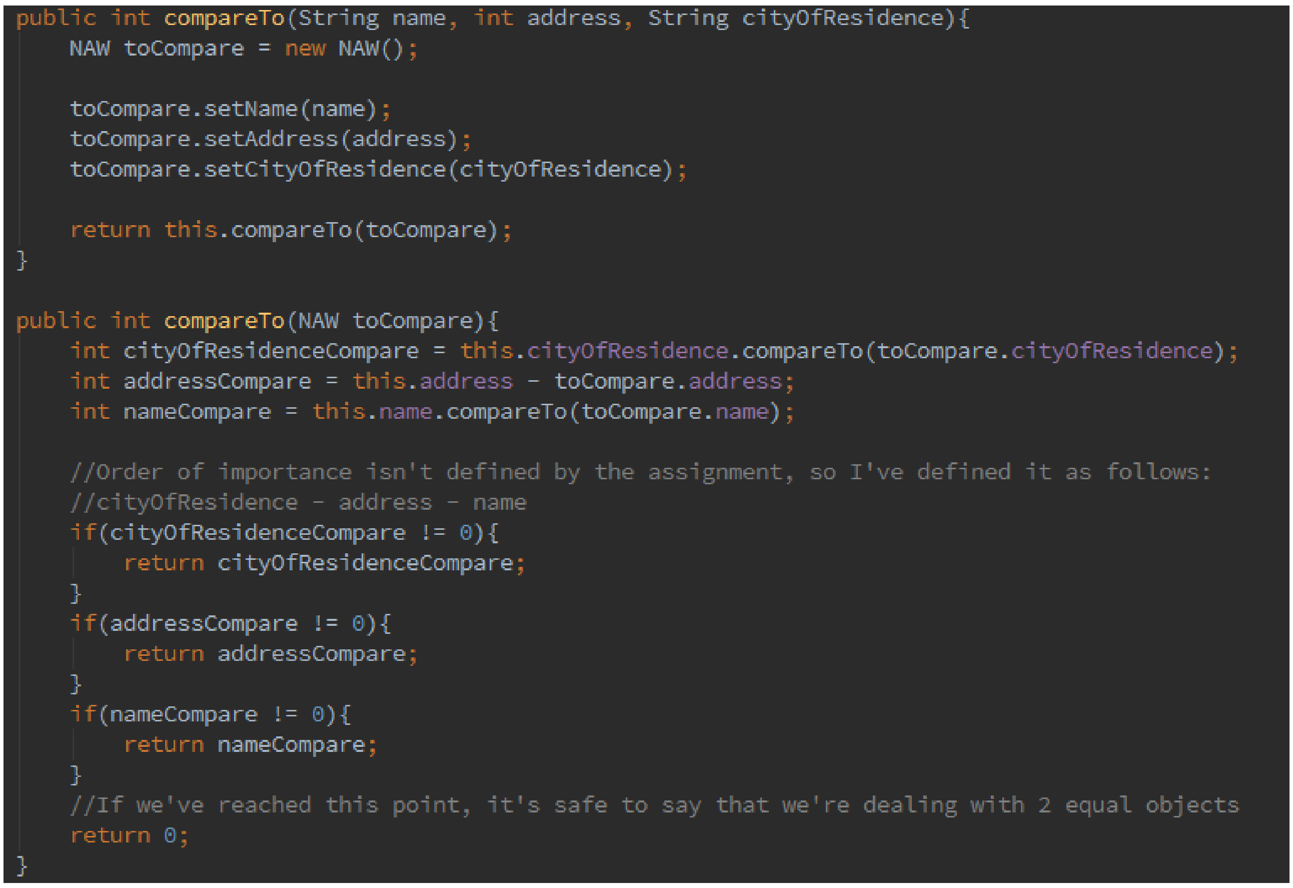


```java
public boolean removeAllItemsWithName(String name){
    boolean success = false;

    int index = this.getIndexOfItemWithName(name);

    while(index != -1){
        success = true;
        this.removeItemWithIndex(index);

        index = this.getIndexOfItemWithName(name);
    }

    return success;
}

public boolean removeFirstItemWithAddressAndCityOfResidence(int address, String cityOfResidence){
    boolean success = false;

    int targetIndex = this.getIndexOfItemWithAddressAndCityOfResidence(address, cityOfResidence);

    if(targetIndex != -1){
        this.removeItemWithIndex(targetIndex);
        success = true;
    }

    return success;
}
```



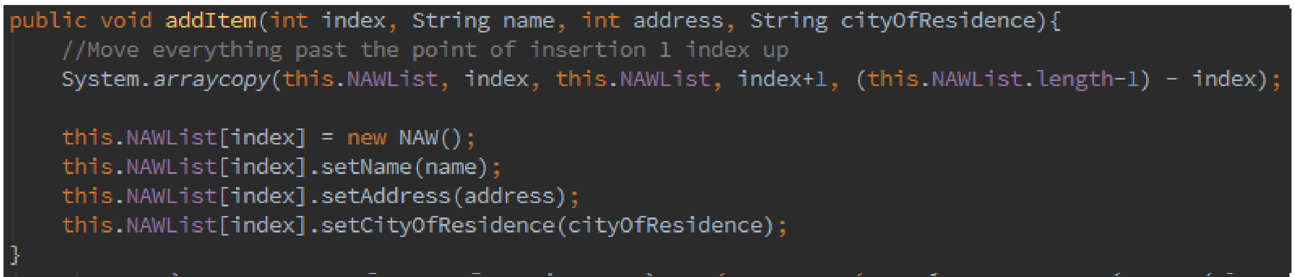
```java
//Incredibly long name...
public boolean removeAllItemsWithAddressAndCityOfResidence(int address, String cityOfResidence){
    boolean success = false;

    int index = this.getIndexOfItemWithAddressAndCityOfResidence(address, cityOfResidence);

    while(index != -1){
        success = true;
        this.removeItemWithIndex(index);

        index = this.getIndexOfItemWithAddressAndCityOfResidence(address, cityOfResidence);
    }

    return success;
}
```

Deel 2

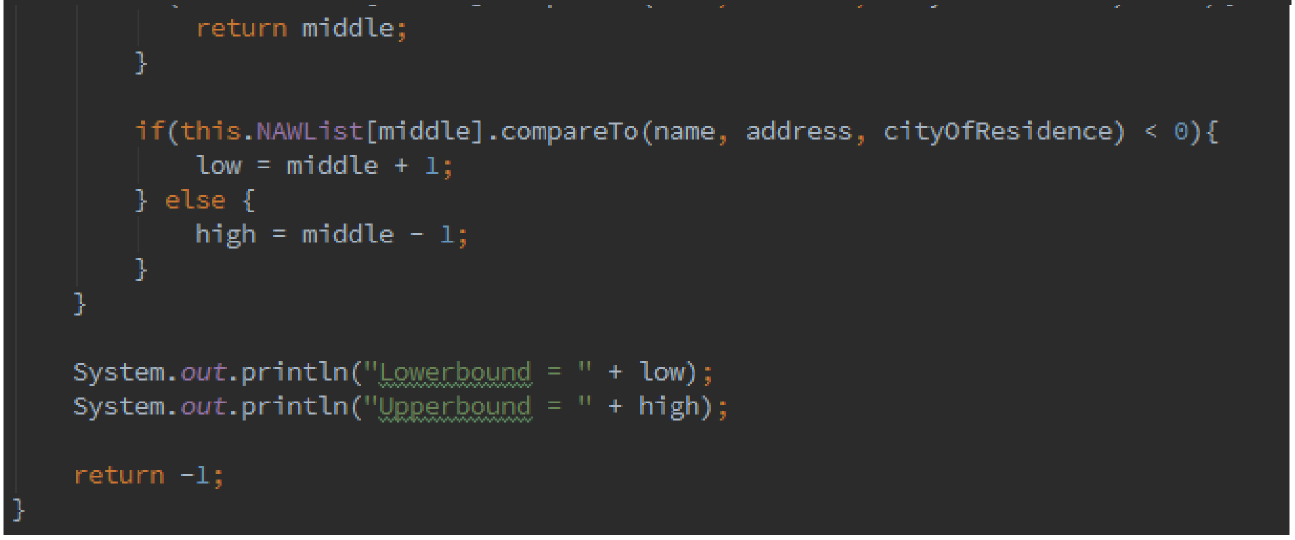


```java
public int compareTo(String name, int address, String cityOfResidence){
    NAW toCompare = new NAW();

    toCompare.setName(name);
    toCompare.setAddress(address);
    toCompare.setCityOfResidence(cityOfResidence);

    return this.compareTo(toCompare);
}

public int compareTo(NAW toCompare){
    int cityOfResidenceCompare = this.cityOfResidence.compareTo(toCompare.cityOfResidence);
    int addressCompare = this.address - toCompare.address;
    int nameCompare = this.name.compareTo(toCompare.name);

    //Order of importance isn't defined by the assignment, so I've defined it as follows:
    //cityOfResidence - address - name
    if(cityOfResidenceCompare != 0){
        return cityOfResidenceCompare;
    }
    if(addressCompare != 0){
        return addressCompare;
    }
    if(nameCompare != 0){
        return nameCompare;
    }
    //If we've reached this point, it's safe to say that we're dealing with 2 equal objects
    return 0;
}
```

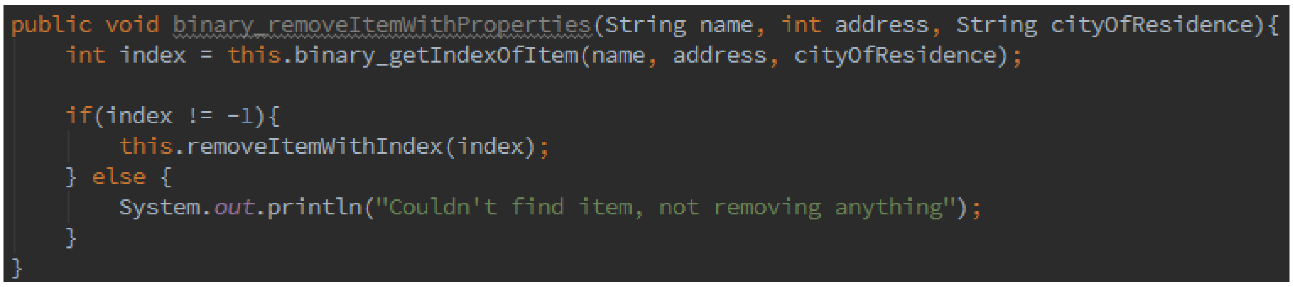
Opgave 1



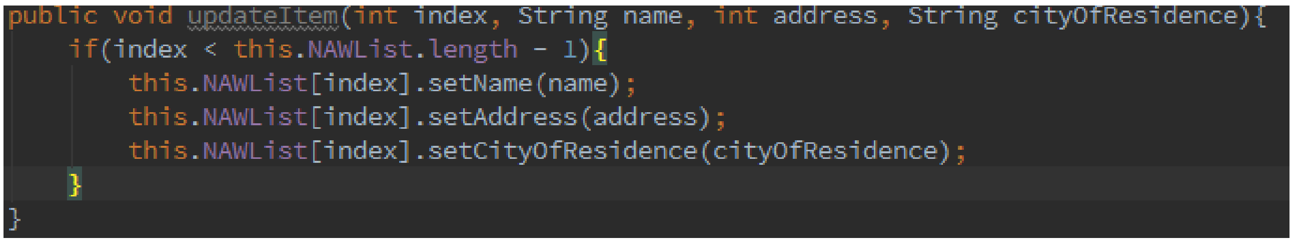
```
public void addItem(int index, String name, int address, String cityOfResidence){
    //Move everything past the point of insertion 1 index up
    System.arraycopy(this.NAWList, index, this.NAWList, index+1, (this.NAWList.length-1) - index);

    this.NAWList[index] = new NAW();
    this.NAWList[index].setName(name);
    this.NAWList[index].setAddress(address);
    this.NAWList[index].setCityOfResidence(cityOfResidence);
}
```



```
            return middle;
        }

        if(this.NAWList[middle].compareTo(name, address, cityOfResidence) < 0){
            low = middle + 1;
        } else {
            high = middle - 1;
        }
    }

    System.out.println("Lowerbound = " + low);
    System.out.println("Upperbound = " + high);

    return -1;
}
```



```
public void binary_removeItemWithProperties(String name, int address, String cityOfResidence){
    int index = this.binary_getIndexOfItem(name, address, cityOfResidence);

    if(index != -1){
        this.removeItemWithIndex(index);
    } else {
        System.out.println("Couldn't find item, not removing anything");
    }
}
```

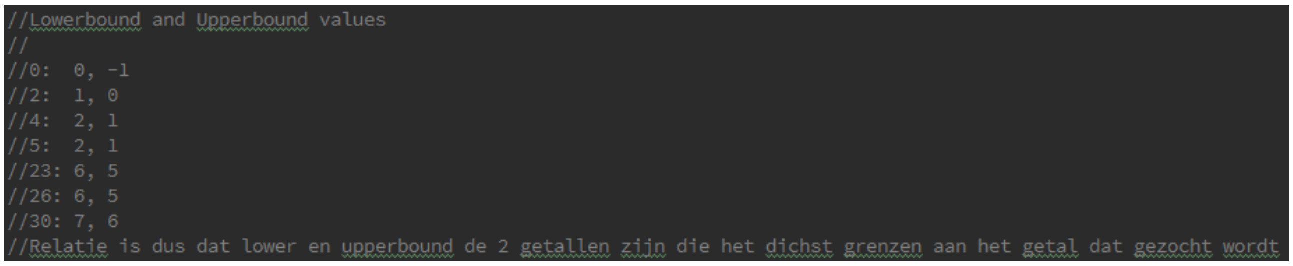


```
public void updateItem(int index, String name, int address, String cityOfResidence){
    if(index < this.NAWList.length - 1){
        this.NAWList[index].setName(name);
        this.NAWList[index].setAddress(address);
        this.NAWList[index].setCityOfResidence(cityOfResidence);
    }
}
```

Opgave 3

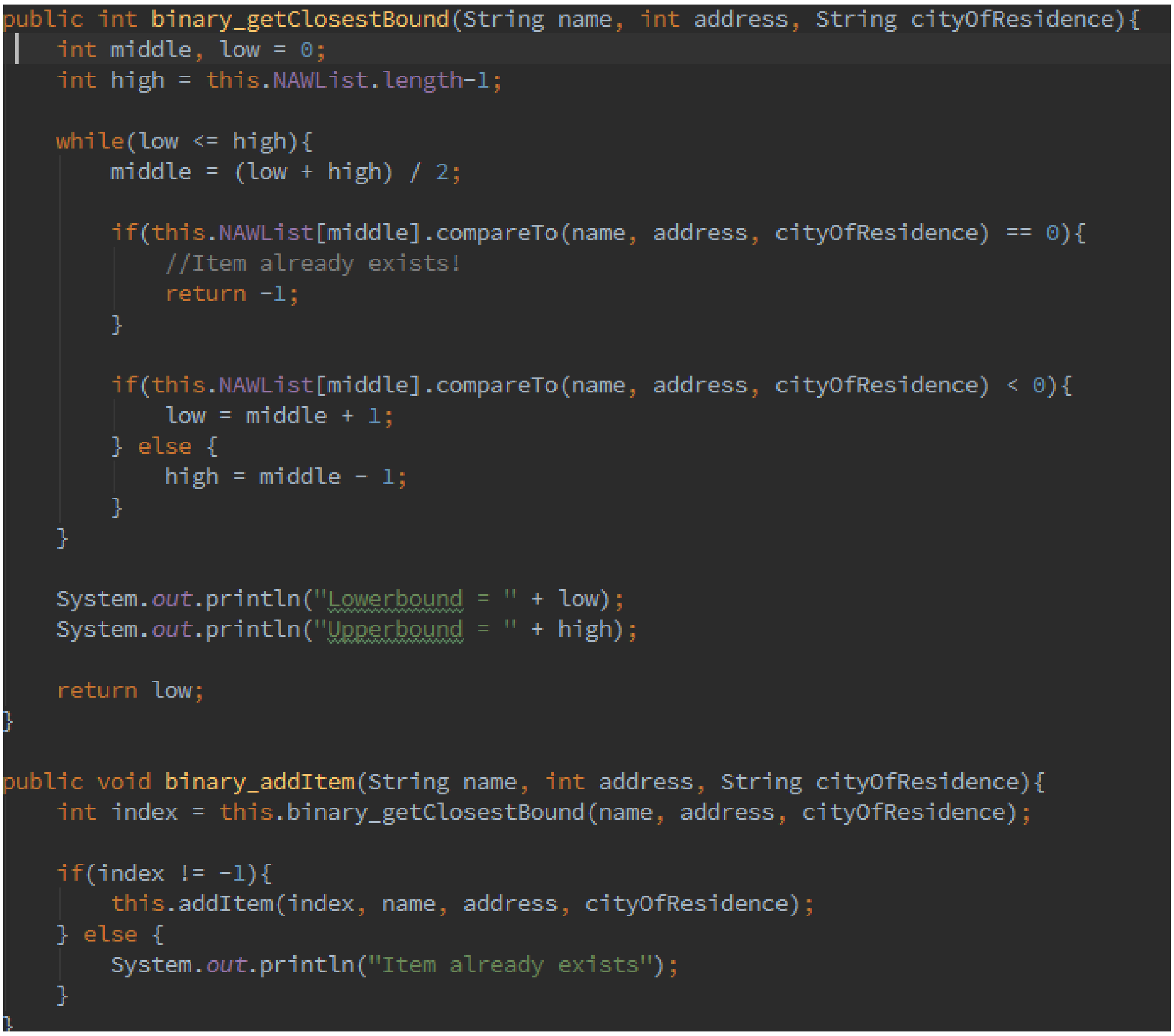
Staat in vorige implementatie

b/c:



```
//Lowerbound and Upperbound values
//
//0:  0, -1
//2:  1, 0
//4:  2, 1
//5:  2, 1
//23: 6, 5
//26: 6, 5
//30: 7, 6
//Relatie is dus dat lower en upperbound de 2 getallen zijn die het dichst grenzen aan het getal dat gezocht wordt
```

Opgave 4



```java
public int binary_getClosestBound(String name, int address, String cityOfResidence){
    int middle, low = 0;
    int high = this.NAWList.length-1;

    while(low <= high){
        middle = (low + high) / 2;

        if(this.NAWList[middle].compareTo(name, address, cityOfResidence) == 0){
            //Item already exists!
            return -1;
        }

        if(this.NAWList[middle].compareTo(name, address, cityOfResidence) < 0){
            low = middle + 1;
        } else {
            high = middle - 1;
        }
    }

    System.out.println("Lowerbound = " + low);
    System.out.println("Upperbound = " + high);

    return low;
}

public void binary_addItem(String name, int address, String cityOfResidence){
    int index = this.binary_getClosestBound(name, address, cityOfResidence);

    if(index != -1){
        this.addItem(index, name, address, cityOfResidence);
    } else {
        System.out.println("Item already exists");
    }
}
```