Java Oefeningen – Reeks 4 (extra) – Interfaces

We gaan onze eigen versie van een **LinkedList** en een **ArrayList** van Java maken. We gaan gebruik maken van de volgende interface:

```
public interface MijnList<T> {
	void add(T element);
	T get(int index);
	int size();
}
```

Deze interface definieert een minimalistische lijst. In deze lijst kunnen we elementen van type T toevoegen, elementen opvragen en de grootte van de lijst controleren. De interface definieert slechts de functionaliteit die **MijnList** moet bieden, maar zegt niet hoe die precies geïmplementeerd moet worden.

MijnArrayList

Maak een nieuwe klasse MijnArrayList die de MijnLijst interface implementeert:

```
public class MijnArrayList<T> implements MijnList<T> {
	T[] data;
}
```

Deze klasse moet variabelen hebben die de elementen van de lijst bijhouden. De ArrayList slaat de elementen op in een array van type T. Telkens er een nieuw element wordt toegevoegd moet dus de array groter gemaakt worden. Dit betekent: maak een nieuwe array van de gewenste grootte, kopieer alle elementen uit de originele array en zet het nieuwe element op de laatste plaats.

Noot: je kan niet rechtstreeks een array van generics construeren. Dit doe je op de volgende wijze:
`T[] data = (T[]) new Object[size];`

Maak nu ook een klasse TestLijsten aan die een main functie bevat. In deze klasse kan je jouw implementatie uitproberen:

```
public class TestLijsten {
	public static void main(String[] args) {
		MijnList<Integer> l = new MijnArrayList<Integer>();
		l.add(1);l.add(5);l.add(-4); l.add(17); l.add(3);
		for(int i = 0; i < l.size(); i++)
			System.out.println(l.get(i));
	}
}
```

MijnLinkedList

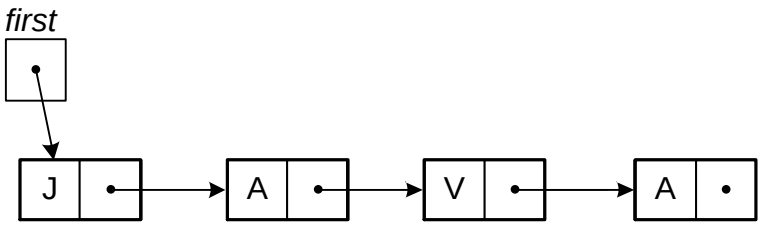
Een LinkedList houdt de data niet bij in een Java array, maar in een speciale structuur van LijstElementen.

```
public class MijnLinkedList<T> implements MijnList<T> {
	private class LijstElement{
		T data;
		LijstElement next;
		…
	}

	LijstElement first = null;

	…
}
```

Een **LinkedList** is dus een ketting van **LijstElementen** waarbij het *next* veld van een **LijstElement** telkens wijst naar het volgende element in de lijst. Om een element toe te voegen in de lijst moet je dus enkel maar een new LijstElement aanmaken met een bepaalde waarde en de *next* van het laatste element naar dit nieuwe element laten wijzen. Om een element op een bepaalde index terug te vinden, start je op het eerste element en volg je met een for- of while-loop de next waarde van het huidige element totdat je op de index zit die er gevraagd werd.



Figuur 1: Grafische voorstelling van een LinkedList die letters bevat.

Uitbreidingen

Breid nu de MijnLijst interface uit met 2 nieuwe functies:

```
public interface MijnList<T> {
	void add(T element);
	T get(int index);
	int size();

	void remove(int index); // verwijdert het element op positie index
	void swap(int i1, int i2); // wisselt elementen op i1 en i2 van plaats
}
```

Implementeer nu ook deze functies in de **MijnArrayList** en **MijnLinkedList** implementaties.