

OO for Data Structures

**CPSC 219: Introduction to Computer Science for Multidisciplinary
Studies II
Fall 2023**

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

Friday, 10 November 2023

Copyright © 2023



Reproducing ArrayList

- Let's recreate Java's ArrayList as a test of learned skills and design
- We'll do this in two ways
 1. Simple inheritance
 2. Full re-creation

Reproducing ArrayList

- Let's recreate Java's ArrayList as a test of learned skills and design
- We'll do this in two ways
 1. Simple inheritance
 2. Full re-creation
- To differentiate ours we'll add a name to our list

```
private String name;
```

Simple Inheritance

Simple Named ArrayList

```
public class NamedArrayListSimple{
    private String name;
    public NamedArrayListSimple(String name){
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Reproducing ArrayList

- Now that we have a named String storage we'll expand the existing ArrayList

```
import java.util.ArrayList;
```

```
public class NamedArrayListSimple extends ArrayList {
```

Reproducing ArrayList

- Let's try to use it

```
public static void main(String[] args) {  
    NamedArrayListSimple nals = new NamedArrayListSimple("MyList");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
}
```

Reproducing ArrayList

- Let's try to use it

```
public static void main(String[] args) {  
    NamedArrayListSimple nals = new NamedArrayListSimple("MyList");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
    System.out.println(nals);    [1, 2, 3]  
}
```


Reproducing ArrayList

- Fix that issue with not printing name

@Override

```
public String toString(){  
    return this.name+super.toString();  
}
```

MyList[1, 2, 3]

Reproducing ArrayList

- Now let us look at equals?

```
public static void main(String[] args) {  
    NamedArrayListSimple nals = new NamedArrayListSimple("MyList1");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
    NamedArrayListSimple nals2 = new NamedArrayListSimple("MyList2");  
    nals2.add(1);  
    nals2.add(2);  
    nals2.add(3);  
    System.out.println(nals.equals(nals2));    True (names are different)  
}
```

Reproducing ArrayList

@Override

```
public boolean equals(Object other) {  
    if (this == other) {  
        return true;  
    }  
    if (other == null || !(other instanceof NamedArrayListSimple)) {  
        return false;  
    }  
    if (!super.equals(other)) {  
        return false;  
    }  
    NamedArrayListSimple that = (NamedArrayListSimple) other;  
    return this.name.equals(that.name);  
}
```

Reproducing ArrayList

- Look at hashCode?

```
public static void main(String[] args) {  
    NamedArrayListSimple nals = new NamedArrayListSimple("MyList1");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
    NamedArrayListSimple nals2 = new NamedArrayListSimple("MyList2");  
    nals2.add(1);  
    nals2.add(2);  
    nals2.add(3);  
    System.out.println(nals.hashCode());  
    System.out.println(nals2.hashCode());  
}
```

The same number 30817

Reproducing ArrayList

- Look at hashCode?

@Override

```
public int hashCode() {  
    return Objects.hash(super.hashCode(), name);  
}
```

-1138147225

-1138147224

Reproducing ArrayList

- What about type limiting

```
public static void main(String[] args) {  
    NamedArrayListSimple nals = new NamedArrayListSimple("MyList1");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
    nals.add("Jonathan");  
    System.out.println(nals);  
}
```

MyList1[1, 2, 3, Jonathan]

Reproducing ArrayList

- Will not compile

```
public static void main(String[] args) {  
    NamedArrayListSimple<Integer> nals = new NamedArrayListSimple<>("MyList1");  
    nals.add(1);  
    nals.add(2);  
    nals.add(3);  
    nals.add("Jonathan");  
    System.out.println(nals);  
}
```

Reproducing ArrayList

- Fix so now the fail at compile is that Generics are not being followed

```
public class NamedArrayListSimple<E> extends ArrayList<E>{
```

```
NamedArrayListSimple<Integer> nals = new NamedArrayListSimple<>("MyList1");  
nals.add(1);  
nals.add(2);  
nals.add(3);  
nals.add("Jonathan");  
System.out.println(nals);
```


Re-Create

Named ArrayList (Re-Created)

```
public class NamedArrayList {  
    private String name;  
    private Object[] data;  
    public NamedArrayList(String name) {  
        this.name = name;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

Named ArrayList (Generics)

```
public class NamedArrayList<E> {  
    private String name;  
    private E[] data;  
    public NamedArrayList(String name) {  
        this.name = name;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

Named ArrayList (Data)

```
public class NamedArrayList<E> {  
    private String name;  
    private E[] data;  
    private int next_index;  
    private static final int DEFAULT = 10;  
    public NamedArrayList(String name) {  
        this.name = name;  
        this.data = (E[]) (new Object[DEFAULT]);  
        this.next_index = 0;  
    }  
}
```

Named ArrayList (Data)

```
public class NamedArrayList<E> {  
    private String name;  
    private E[] data;  
    private int next_index;  
    private static final int DEFAULT = 10;  
    public NamedArrayList(String name, int size) {  
        this.name = name;  
        this.data = (E[]) (new Object[size]);  
        this.next_index = 0;  
    }  
}
```

Named ArrayList (Make it a List)

```
import java.util.List;
```

```
public class NamedArrayList<E> implements List<E> {
```

- Will not compile as we need to complete List interface contract

Named ArrayList (Make it a List)

```
public int size()
public boolean isEmpty()
public boolean contains(Object o)
public Iterator<E> iterator()
public Object[] toArray()
public <T> T[] toArray(T[] a)
public boolean add(E e)
public boolean remove(Object o)
public boolean containsAll(Collection<?> c)
public boolean addAll(Collection<? extends E> c)
public boolean addAll(int index,
Collection<? extends E> c)
public boolean removeAll(Collection<?> c)
public boolean retainAll(Collection<?> c)
public void clear()
public E get(int index)
public E set(int index, E element)
public void add(int index, E element)
public E remove(int index)
public int indexOf(Object o)
public int lastIndexOf(Object o)
public ListIterator<E> listIterator()
public ListIterator<E> listIterator(int index)
public List<E> subList(int fromIndex, int
toIndex)
```

Named ArrayList (Make it a List)

- We'll start with simpler ones

```
public int size()
public boolean isEmpty()
public boolean contains(Object o)
public boolean add(E e)
public boolean remove(Object o)
public void clear()
public E get(int index)
public E set(int index, E element)
public void add(int index, E element)
public E remove(int index)
```


Named ArrayList (Make it a List)

- We'll start with simpler ones

```
@Override
```

```
public int size() {  
    return next_index;  
}
```

```
@Override
```

```
public boolean isEmpty() {  
    return next_index==0;  
}
```

Named ArrayList (Make it a List)

- We'll start with simpler ones

@Override

```
public boolean contains(Object o) {  
    for (int i = 0; i < next_index; i++) {  
        if(data[i].equals(o)){  
            return true;  
        }  
    }  
    return false;  
}
```

Named ArrayList (Make it a List)

@Override

```
public boolean add(E e) {  
    if(next_index == data.length){  
        E[] new_data = (E[])(new Object[data.length*2]);  
        for (int i = 0; i < data.length; i++) {  
            new_data[i] = data[i];  
        }  
        this.data = new_data;  
    }  
    data[next_index] = e;  
    next_index++;  
    return true;  
}
```

Named ArrayList (Make it a List)

@Override

```
public boolean remove(Object o) {  
    for (int i = 0; i < data.length; i++) {  
        if(data[i].equals(o)){  
            for (int j = i + 1; j < data.length; j++) {  
                data[j-1] = data[j];  
            }  
            next_index--;  
            data[next_index] = null;  
            return true;  
        }  
    }  
    return false;  
}
```

Named ArrayList (Make it a List)

```
public E get(int index) {  
    if(index < 0 || index >= size()){  
        throw new IndexOutOfBoundsException(...);  
    }  
    return data[index];  
}
```

Named ArrayList (Make it a List)

@Override

```
public E set(int index, E element) {  
    if(index < 0 || index >= size()){  
        throw new IndexOutOfBoundsException(...);  
    }  
    E temp = this.data[index];  
    this.data[index] = element;  
    return temp;  
}
```

Named ArrayList (Make it a List)

@Override

```
public void add(int index, E element) {  
    if(index < 0 || index >= size()){  
        throw new IndexOutOfBoundsException(...);  
    }  
    if(next_index == data.length){  
        //expand  
    }  
    for (int i = next_index-1; i >= index; i--) {  
        this.data[i+1] = this.data[i];  
    }  
    this.data[index] = element;  
    next_index++;  
}
```

Named ArrayList (Make it a List)

- @Override

```
public E remove(int index) {  
    if(index < 0 || index >= size()){  
        throw new IndexOutOfBoundsException(...);  
    }  
    E temp = this.data[index];  
    for (int i = index; i < next_index; i++) {  
        this.data[i] = this.data[i+1];  
    }  
    this.next_index--;  
    this.data[next_index] = null;  
    return temp;  
}
```


Named ArrayList (Make it a List)

@Override

```
public String toString(){
    StringBuilder sb = new StringBuilder();
    sb.append(this.name);
    sb.append("[");
    for (int i = 0; i < next_index-1; i++) {
        sb.append(this.data[i].toString());
        sb.append(", ");
    }
    if(next_index != 0){
        sb.append(this.data[next_index-1].toString());
    }
    sb.append("]");
    return sb.toString();
}
```

MyList[1, 2, 3]

true

false

MyList[2, 3]

false

MyList[]

Named ArrayList (Make it a List)

```
public static void main(String[] args) {  
    NamedArrayList<Integer> nal = new NamedArrayList<>("MyList");  
    nal.add(1);  
    nal.add(2);  
    nal.add(3);  
    System.out.println(nal);  
}
```

MyList[1, 2, 3]

Named ArrayList (Make it a List)

```
public static void main(String[] args) {  
    NamedArrayList<Integer> nal = new NamedArrayList<>("MyList");  
    System.out.println(nal.size() + " " + nal.isEmpty());  
    nal.add(1);  
    System.out.println(nal.size() + " " + nal.isEmpty());    0 true  
    nal.add(2);                                              1 false  
    System.out.println(nal.size() + " " + nal.isEmpty());    2 false  
    nal.add(3);                                              3 false  
    System.out.println(nal.size() + " " + nal.isEmpty());  
}
```

Named ArrayList (Make it a List)

```
public static void main(String[] args) {  
    NamedArrayList<Integer> nal = new NamedArrayList<>("MyList");  
    nal.add(1);    nal.add(2);    nal.add(3);  
    System.out.println(nal);  
    System.out.println(nal.contains(1));  
    System.out.println(nal.contains(4));  
    nal.remove(new Integer(1));  
    System.out.println(nal);  
    System.out.println(nal.contains(1));  
    nal.clear();  
    System.out.println(nal);  
}
```

```
MyList[1, 2, 3]  
true  
false  
MyList[2, 3]  
false  
MyList[]
```

Named ArrayList (Make it a List)

```
public static void main(String[] args) {  
    NamedArrayList<Integer> nal = new NamedArrayList<>("MyList");  
    nal.add(1);    nal.add(2);    nal.add(3);  
    System.out.println(nal);  
    System.out.println(nal.get(0));  
    System.out.println(nal.get(1));  
    System.out.println(nal.get(2));  
    nal.set(1, 99);  
    System.out.println(nal);  
    nal.add(1,4);  
    System.out.println(nal);  
    nal.remove(2);  
    System.out.println(nal);  
}
```

MyList[1, 2, 3]

1

2

3

MyList[1, 99, 3]

MyList[1, 4, 99, 3]

MyList[1, 4, 3]

Onward to ... Recursive Data Structure Design.

Jonathan Hudson
jwhudson@ucalgary.ca
<https://pages.cpsc.ucalgary.ca/~jwhudson/>



UNIVERSITY OF
CALGARY