



# Programming in Java – Solution of assignments 2



Paolo Vercesi

Technical Program Manager

# Assignment

```
public interface Collection {  
    boolean isEmpty();  
    int getSize();  
    boolean contains(String string);  
    String[] getValues();  
}  
public interface Stack extends Collection {  
    void push(String string);  
    String pop();  
    String top();  
}
```

```
public interface List extends Collection {  
    void add(String string);  
    String get(int index);  
    void insertAt(int index, String string);  
    void remove(int index);  
    int indexOf(String string);  
}
```

Implement the Stack and List interfaces. Minimize code duplication.

Hint: consider the usage of both inheritance and composition.



# Collection

```
public interface Collection {  
    String[] getValues();  
  
    default boolean isEmpty() {  
        return getSize() == 0;  
    }  
  
    default int getSize() {  
        return getValues().length;  
    }  
  
    default boolean contains(String value) {  
        for (String datum : getValues()) {  
            if (Objects.equals(datum, value)) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

The `getValues()` method exposes a lot of information. We can implement all the other “optional” methods from `getValues()`.



# List and Stack

```
public interface List extends Collection {  
  
    void add(String value);  
  
    default String get(int index) {  
        return getValues()[index];  
    }  
  
    void insert(int index, String value);  
  
    void remove(int index);  
  
    default int indexOf(String value) {  
        for (int i = 0; i < getSize(); i++) {  
            if (Objects.equals(get(i), value)) {  
                return i;  
            }  
        }  
        return -1;  
    }  
}
```

```
public interface Stack extends Collection {  
  
    void push(String value);  
  
    String pop();  
  
    default String top() {  
        return getValues()[getSize() - 1];  
    }  
}
```

We are able to implement many methods as queries on other methods. The `getValues()` method provide access to the full state of the collection.



# MyList

```
public class MyList implements List {  
  
    private String[] data = new String[0];  
  
    public String[] getValues() {  
        return Arrays.copyOf(data, data.length);  
    }  
  
    public void add(String value) {  
        String[] newData = new String[data.length + 1];  
        System.arraycopy(data, 0, newData, 0, data.length);  
        newData[newData.length - 1] = value;  
        this.data = newData;  
    }  
  
    public void insert(int index, String value) {  
        String[] newData = new String[data.length + 1];  
        System.arraycopy(data, 0, newData, 0, index);  
        newData[index] = value;  
        System.arraycopy(data, index, newData, index + 1, data.length - index);  
        this.data = newData;  
    }  
  
    public void remove(int index) {  
        String[] newData = new String[data.length - 1];  
        System.arraycopy(data, 0, newData, 0, index);  
        System.arraycopy(data, index, newData, index, data.length - index - 1);  
        this.data = newData;  
    }  
}
```



# MyStack

```
public class MyStack implements Stack {  
    private String[] data = new String[0];  
  
    public String[] getValues() {  
        return Arrays.copyOf(data, data.length);  
    }  
  
    public void push(String value) {  
        String[] newData = new String[data.length + 1];  
        System.arraycopy(data, 0, newData, 0, data.length);  
        newData[newData.length - 1] = value;  
        this.data = newData;  
    }  
  
    public String pop() {  
        String value = top();  
        String[] newData = new String[data.length - 1];  
        System.arraycopy(data, 0, newData, 0, newData.length);  
        this.data = newData;  
        return value;  
    }  
}
```

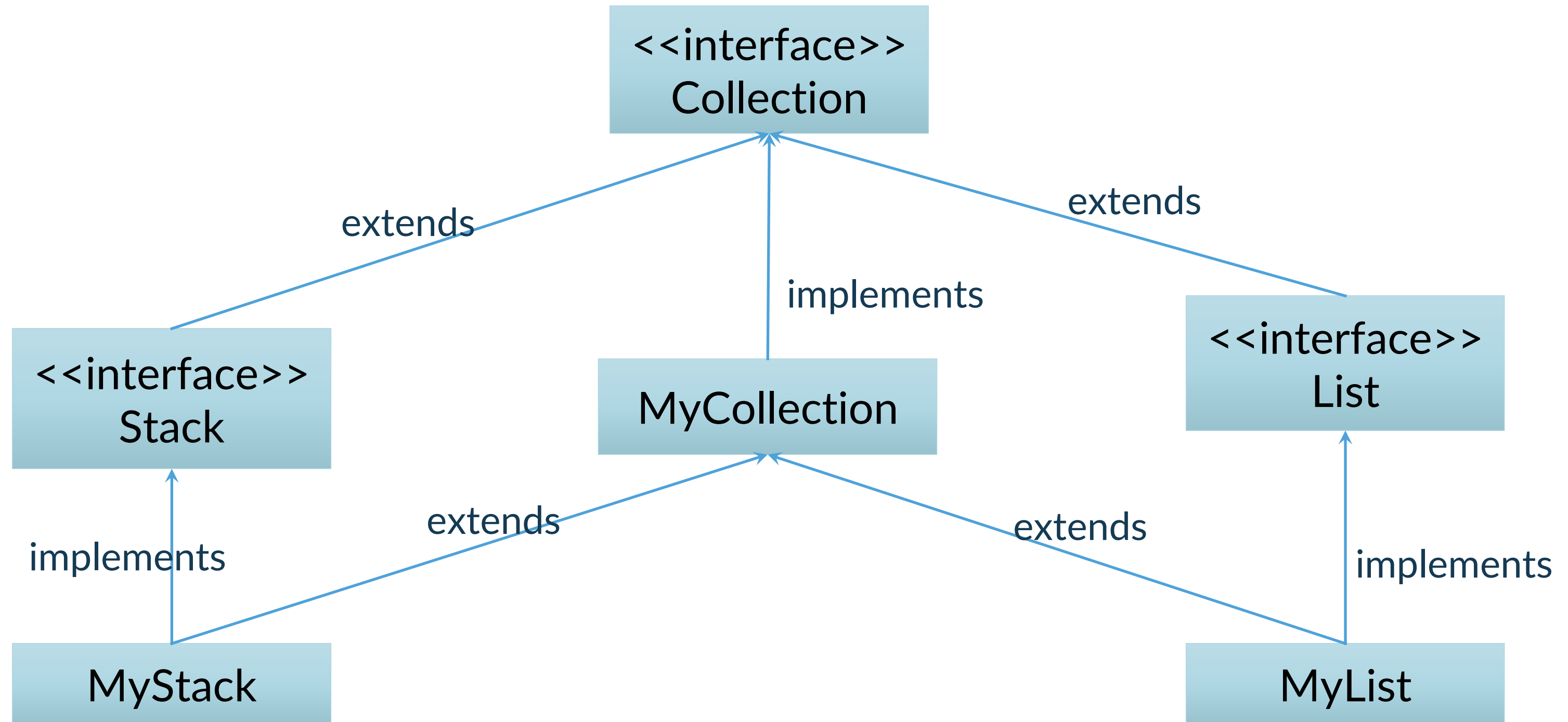


# Comparison of MyList and MyStack

- Object state is represented in the same way  
`private String[] data = new String[0];`
- Same implementation of  
`public String[] getValues()`
- Same implementation of  
`public void push(String value)`  
`public void add(String value)`



# Inheritance 1





```

class MyCollection implements Collection {

    private String[] data = new String[0];

    public String[] getValues() {
        return Arrays.copyOf(data, data.length);
    }

    public int getSize() {
        return data.length;
    }

    void add(String value) {
        String[] newData = new String[data.length + 1];
        System.arraycopy(data, 0, newData, 0, data.length);
        newData[newData.length - 1] = value;
        this.data = newData;
    }

    void remove(int index) {
        String[] newData = new String[data.length - 1];
        System.arraycopy(data, 0, newData, 0, index);
        System.arraycopy(data, index, newData, index, data.length - index - 1);
        this.data = newData;
    }

    void insert(int index, String value) {
        String[] newData = new String[data.length + 1];
        System.arraycopy(data, 0, newData, 0, index);
        newData[index] = value;
        System.arraycopy(data, index, newData, index + 1, data.length - index);
        this.data = newData;
    }
}

```

# Inheritance 1 - MyCollection

Methods that are not part of the Collection interface are **package protected**



# Inheritance 1 - MyStack

```
public class MyStack extends MyCollection implements Stack {  
  
    @Override  
    public void push(String value) {  
        super.add(value);  
    }  
  
    @Override  
    public String pop() {  
        String value = top();  
        remove(getSize()-1);  
        return value;  
    }  
  
}
```

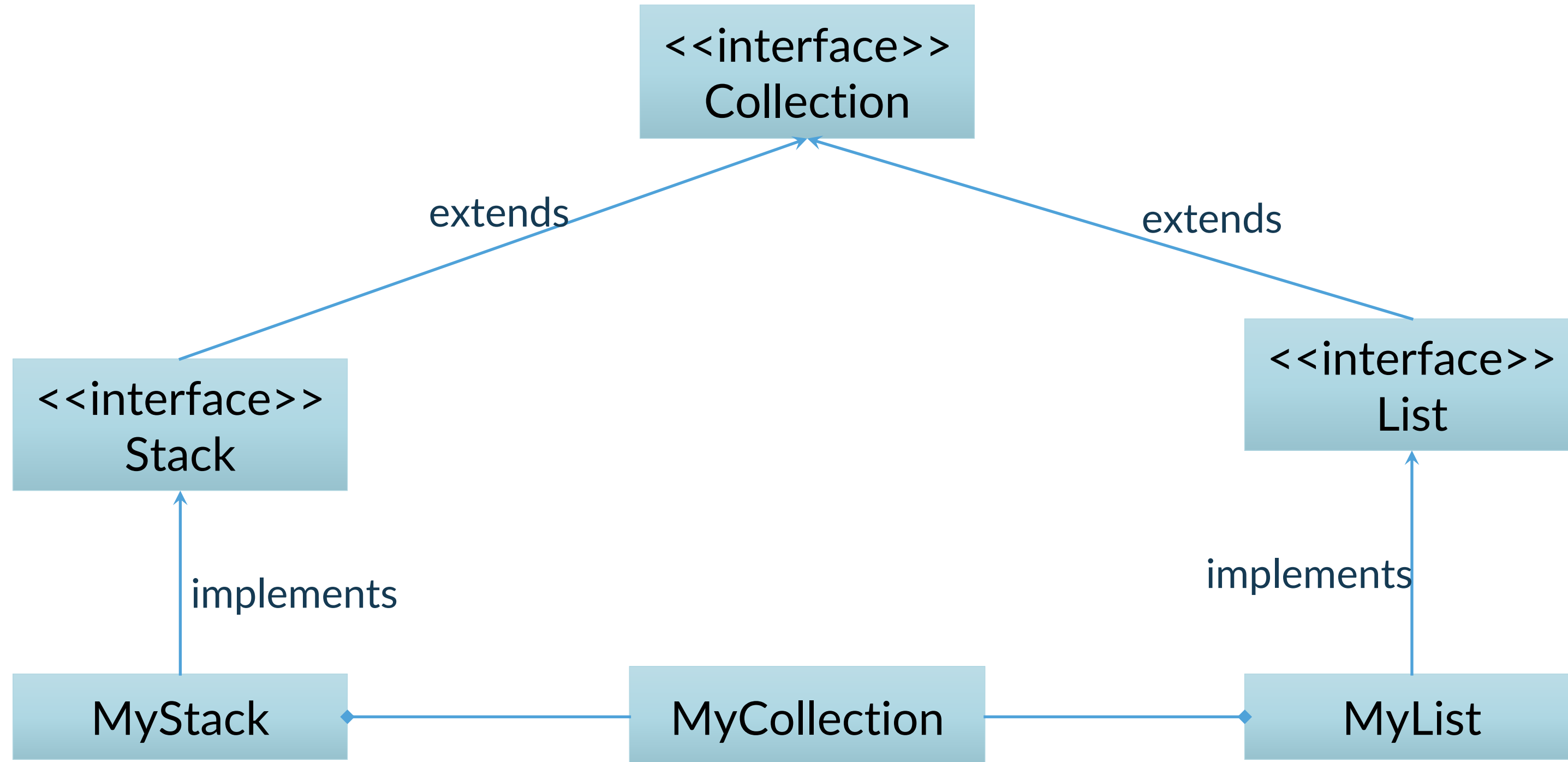


# Inheritance 1 - MyList

```
public class MyList extends MyCollection implements List {  
  
    @Override  
    public void add(String value) {  
        super.add(value);  
    }  
  
    @Override  
    public void insert(int index, String value) {  
        super.insert(index, value);  
    }  
  
    @Override  
    public void remove(int index) {  
        super.remove(index);  
    }  
}
```



# Composition



# Composition - MyCollection

```
class MyCollection {  
  
    private String[] data = new String[0];  
  
    String[] getValues() {  
        return Arrays.copyOf(data, data.length);  
    }  
  
    int getSize() {  
        return data.length;  
    }  
  
    void add(String value) {  
        String[] newData = new String[data.length + 1];  
        System.arraycopy(data, 0, newData, 0, data.length);  
        newData[newData.length - 1] = value;  
        this.data = newData;  
    }  
  
    void remove(int index) {  
        String[] newData = new String[data.length - 1];  
        System.arraycopy(data, 0, newData, 0, index);  
        System.arraycopy(data, index, newData, index, data.length - index - 1);  
        this.data = newData;  
    }  
  
    public void insert(int index, String value) {  
        String[] newData = new String[data.length + 1];  
        System.arraycopy(data, 0, newData, 0, index);  
        newData[index] = value;  
        System.arraycopy(data, index, newData, index + 1, data.length - index);  
        this.data = newData;  
    }  
}
```



# Composition - MyList

```
public class MyList implements List {  
  
    private final MyCollection collection = new MyCollection();  
  
    @Override  
    public String[] getValues() {  
        return collection.getValues();  
    }  
  
    @Override  
    public void add(String value) {  
        collection.add(value);  
    }  
  
    @Override  
    public void insert(int index, String value) {  
        collection.insert(index, value);  
    }  
  
    @Override  
    public void remove(int index) {  
        collection.remove(index);  
    }  
  
}
```

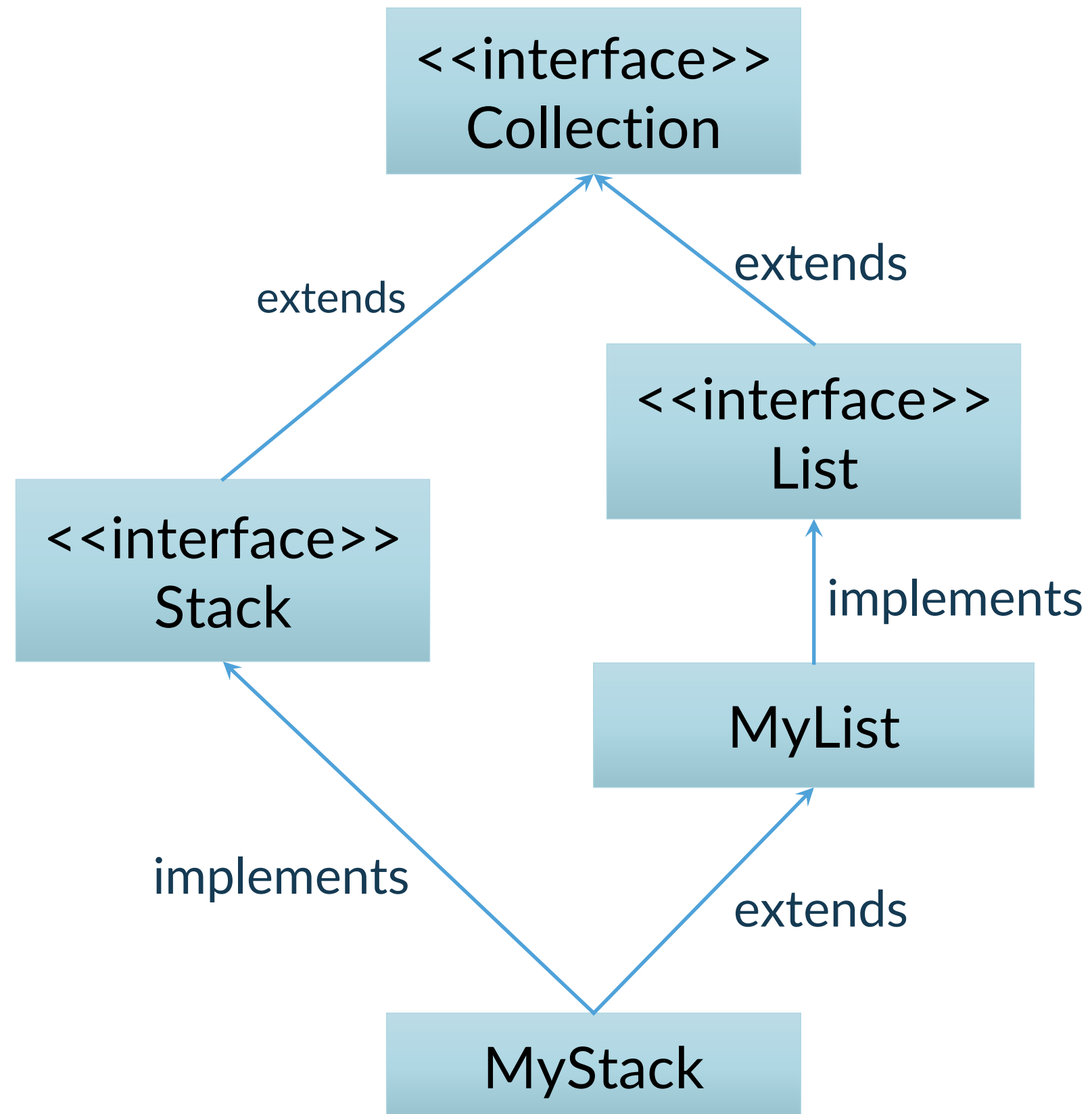


# Composition - MyStack

```
public class MyStack implements Stack {  
    private final MyCollection collection = new MyCollection();  
  
    @Override  
    public String[] getValues() {  
        return collection.getValues();  
    }  
  
    @Override  
    public void push(String value) {  
        collection.add(value);  
    }  
  
    @Override  
    public String pop() {  
        String value = top();  
        collection.remove(collection.getSize()-1);  
        return value;  
    }  
}
```



# Inheritance 2







Thank you!

[esteco.com](http://esteco.com)

