Java vs other languages

Write down at least one key difference between Java and every other language we’ve seen this semester. Why do you think the designers of Java felt it was important for Java to be this way?

(Your response)
How good is it?
These are only some of the metrics for “good”:

Is it correct?
Do we have tests?

Is it clear?
Is it documented? Is it straightforward?

Is it efficient?
Is it judicious in its use of time and space?

Is it extensible?
How easy is it to add new kinds of data or new behaviors?

...
How to design and implement a data structure

Design the interface: what operations will it support?
What’s the “contract” between implementor and user?

Write tests
Yes, do this first! It helps make the specification clear.

Design an encoding that can handle the operations
What existing data structures will you use to implement your new one?

Implement the operations
Test as you go. Where possible, reuse code (interface, helper methods, etc.)
How to design and implement a data structure in Java

Design the interface: what operations will it support?
What’s the “contract” between implementor and user?

Write tests
Yes, do this first! It helps make the specification clear.

Design an encoding that can handle the operations
What existing data structures will you use to implement your new one?

Implement the operations
Test as you go. Where possible, reuse code (interface, helper methods, etc.)
An interface for a list of integers

public interface IntList {
    /**
     * The number of elements in the list
     */
    public int size();

    /**
     * Returns true if the list is empty
     */
    public boolean isEmpty();

    /**
     * Returns true if the list contains the value
     *
     * @param value
     */
    public boolean contains(int value);

    /**
     * Sorts the elements of the list
     */
    public void sort();
}
An implementation for a list of integers

public class ArrayIntList implements IntList {
    private int[] values;

    public ArrayIntList() {
        this.values = new int[] {};
    }

    public ArrayIntList(int[] values) {
        this.values = values;
    }

    @Override
    public int size() {
        return values.length;
    }

    // Implement: isEmpty, contains

    ...
How good is it?

These are only *some* of the metrics for “good”.

**Is it correct?**
Do we have tests?

**Is it clear?**
Is it documented? Is it straightforward?

**Is it efficient?**
Is it judicious in its use of time and space?

**Is it extensible?**
How easy is it to add new kinds of data or new behaviors?

...
Adding an operation to the interface

```java
public interface IntList {
    /**
     * The number of elements in the list
     */
    public int size();

    /**
     * Returns true if the list is empty
     */
    public boolean isEmpty();

    /**
     * Returns true if the list contains the value
     *
     * @param value
     */
    public boolean contains(int value);

    /**
     * Sorts the elements of the list
     */
    public void sort();

    /**
     * Insert a value at the end of the list
     *
     * @param value
     */
    public void insert(int value);
}
```
Implementing array-based insert

```java
@Override
public void insert(int value) {
    int[] newValues = new int[this.size()+1];

    for (int i=0; i<this.size(); i++)
        newValues[i] = this.values[i];

    newValues[this.size()] = value;

    this.values = newValues;
}

...or...

@Override
public void insert(int value) {
    int[] newValues = new int[this.size()+1];

    System.arraycopy(this.values, 0, newValues, 0, this.size());

    newValues[this.size()] = value;

    this.values = newValues;
}
```
public interface IntList {
    /**
     * The number of elements in the list
     */
    public int size();

    /**
     * Returns true if the list is empty
     */
    public boolean isEmpty();

    /**
     * Returns true if the list contains the value
     * @param value
     */
    public boolean contains(int value);

    /**
     * Sorts the elements of the list
     */
    public void sort();

    /**
     * Insert a value at the end of the list
     * @param value
     */
    public void insert(int value);
}
...different implementation

```java
public class LinkedIntList implements IntList {

    private ListNode first;

    public LinkedIntList() {
        this.first = null;
    }

    public LinkedIntList(int[] values) {
        this();
        for (int value : values) {
            this.insert(value);
        }
    }

    // Implement: size, isEmpty, contains, insert

    public class ListNode {
        public int value;
        public ListNode next;

        ListNode(int value) {
            this.value = value;
            this.next = null;
        }

        ListNode(int value, ListNode next) {
            this.value = value;
            this.next = next;
        }
    }
}
```
A “new” data structure: Stack

```java
public interface IntStack {
    /**
     * The number of elements in the stack
     */
    public int size();

    /**
     * Returns true if the stack is empty
     */
    public boolean isEmpty();

    /**
     * Insert a value into the stack
     * @param value
     */
    public void push(int value);

    /**
     * Remove a value from the stack
     * @return
     */
    public int pop();
}
```
public interface IntQueue {
    /**
     * The number of elements in the queue
     */
    public int size();

    /**
     * Returns true if the queue is empty
     */
    public boolean isEmpty();

    /**
     * Insert a value into the queue
     * @param value
     */
    public void enqueue(int value);

    /**
     * Remove a value from the queue
     * @return
     */
    public int dequeue();
}
How good is it?

These are only some of the metrics for “good”.

Is it correct?
Do we have tests?

Is it clear?
Is it documented? Is it straightforward?

Is it efficient?
Is it judicious in its use of time and space?

Is it extensible?
How easy is it to add new kinds of data or new behaviors?

...
An interface for a list of integers

```java
public interface IntList {
    /**
     * The number of elements in the list
     */
    public int size();

    /**
     * Returns true if the list is empty
     */
    public boolean isEmpty();

    /**
     * Returns true if the list contains the value
     *
     * @param value
     */
    public boolean contains(int value);

    /**
     * Sorts the elements of the list
     */
    public void sort();
}
```
An interface for a **generic** list

```java
public interface MyList<T> {
    /**
     * The number of elements in the list
     */
    public T size();

    /**
     * Returns true if the list is empty
     */
    public boolean isEmpty();

    /**
     * Returns true if the list contains the value
     * @param value
     */
    public boolean contains(T value);

    /**
     * Sorts the elements of the list
     */
    public void sort();

    /**
     * Insert a value at the end of the list
     * @param value
     */
    public void insert(T value);
}
```

```java
public class MyArrayList implements MyList<Integer> {
    private int[] values;
    ...
}
```

```java
MyList<Integer> values = new MyArrayList();
```