What if our Frog (from lab) could say 10 different things?

```java
public class Frog{
    private boolean formerPrince;
    private String phrase1;
    private String phrase2;
    private String phrase3;
    private String phrase4;
    private String phrase5;
    private String phrase6;
    private String phrase7;
    private String phrase8;
    private String phrase9;
    private String phrase10;
    ...
}
```

What if we want to store lots of things...?

- But we don’t want to declare a separate variable for each one?
- That’s what arrays are good for
What is an Array?

- It's an easy way to declare lots of variables that all have the same type.

```java
type [] variableName = new type [#];
```

E.g. declare an array of integers

```java
int[] data = new int[5]; //total ints = 5
```

- When an array of particular primitive type is created, Java initializes the elements to the type's default value. E.g. Array of ints – default value is zero

```
0 0 0 0 0
```

Array Elements and Indices

- To initialize integer array
  - `data[0] = 6;`
  - `data[1] = 10;`
  - `data[2] = 12;`

- The number within square brackets is called an index
- The valid indices are 0 thru (array length - 1)
  - `0`: first element of the array
  - `n-1`: last element of the array

An Array is an Object

- `int[] data;` - `data` is a reference variable whose type is `int[]`, meaning “array of ints.” At this point its value is null.

```java
data = new int[5];
```

- The new operator causes a chunk of memory big enough for 5 ints to be allocated on the heap. Here, `data` is assigned a reference to the heap address.

```java
data[0] = 6;
data[1] = 10;
data[2] = 12;
```

Initially, all five ints are 0. Here, three of them are assigned other values.

```
int[] info = {6, 10, 12, 0, 0};
int[] info = new int[]{6, 10, 12, 0, -1};
```

Using Array Elements in Expressions

- An element of an array of ints can be used virtually anywhere an expression of type int is valid.
- Likewise for arrays of other types

```java
int[] data = new int[] {6, 10, 12, 0, 0};
int x = data[0];
data[3] = data[2];
System.out.println("data[0] is \" + data[0] + ");
```
Accessing an Array’s Length

- ArrayName.length gives size of the array

```java
int[] data;
data = new int[5];  // data.length is 5
data[0] = 6;
data[1] = 10;
data[2] = 12;

// How to Sum the contents of an array
int result = 0;
for (int i = 0; i < data.length; i++) {
    result = result + data[i];
}
```

// How to Sum the contents of an array
```java
int result = 0;
for (int i = 0; i < data.length; i++) {
    result = result + data[i];
}
```

Complete the sum(…) method

```java
public class ArrayTool{
    /**
     * Takes an array of ints as an argument.
     * returns the sum of all the integers in the array.
     */
    public static int sum(int[] data) {
        for (int i = 0; i < data.length; i++) {
            result = result + data[i];
        }
    }
}
```

Welcome to DrJava
```java
> int[] data = new int[] {6, 10, 12, 0, 0};
> ArrayTool.sum(data)
28
```

Array Out of Bounds Exceptions

```java
public class ArrayTool{
    public static int sum(int[] data) {
        int sum = 0;
        for (int i = 0; i <= data.length; i++) {
            sum = sum + data[i];
        }
        return sum;
    }
}
```

```java
> int[] data = new int[] {6, 10, 12, 0, 0};
> ArrayTool.sum(data) ArrayIndexOutOfBoundsException
```

Declaring & Initializing Arrays of Primitive Type

```java
int[] info1 = {2000, 100, 40, 60};
int[] info2 = new int[]{2000, 100, 40, 60};
char[] choices1 = {'p', 's', 'q'};
char[] choices2 = new char[]{ 'p', 's', 'q'};
double[] temps1 = {75.6, 99.4, 86.7};
double[] temps2 = new double[]{75.6, 99.4, 86.7};
```

Note: The advantage of using the “new type[]” syntax is that it can be used in an assignment statement that is not a variable declaration statement.
**Complete this method**

```java
public class ArrayTool{
    /* Returns true if all integers in the
data array are positive, false otherwise.
    */
    public static boolean allPositive(int[] data){
        ...
    }
}
```

**Finding Max**

- Complete max method, that finds the maximum value in the array
- Assume that input passed is valid, no need for error checking

// Dr Java Interactions Pane
> int[] data = new int[] {6, 10, 12, 0, 0};
> ArrayTool.max(data)
12

**Array of Primitives**

```java
Array of Primitives
```

```java
int[] data;
data = new int[3];
data[0] = 5;
data[1] = 10;
```

**Array of Objects**

```java
Array of Objects
```

```java
Counter[] counters;
counters = new Counter[3];
> counters[0]= new Counter();
> counters[0].addOne();
> counters[1]= new Counter();
> counters[0].getCount()
1
> counters[2].getCount()
NullPointerException
```
E.g. Person Object

```java
public class Person{
    private String name;
    private int age;

    Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public int getAge() { return age; }
    public String getName() { return name; }
}
```

Person Array

```java
Person[] people = {new Person("jo"), new Person("flo")};
Person[] people = new Person[] {new Person("jo"),
                                new Person("flo")};
```

In PersonDB.java

- Complete printDB()

```java
PersonDB people = new PersonDB();
people.printDB()
Name  Age
jo    25
flo   18
mo    19
```

- Complete isInDatabase(...) method

```java
PersonDB people = new PersonDB();
people.isInDatabase("flo")
true
people.isInDatabase("rex")
false
```