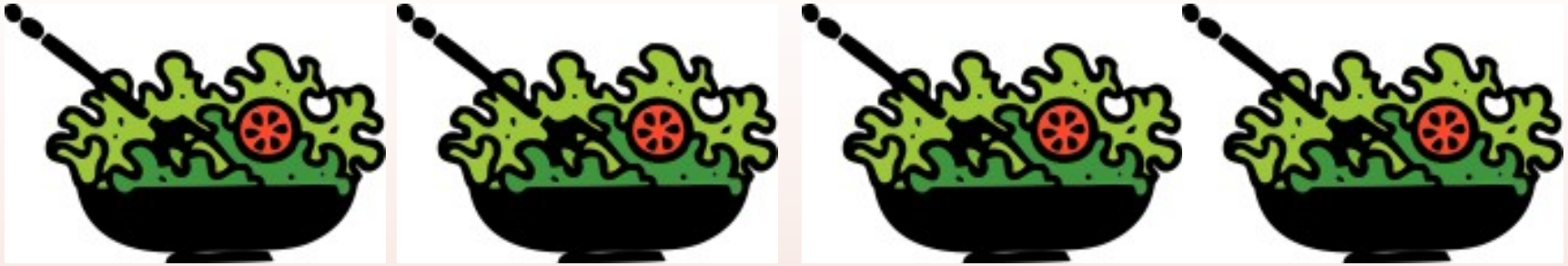

What Is An **Array**?

A "CONCEPTUAL PRESENTATION" WITH SALADS

Let's imagine we own a restaurant and serve five types of salads…

Potato Salad

Macaroni Salad

Cole Slaw

Fruit Salad

3-Bean Salad

and we want to track how many pounds of salad we have on hand…

Potato Salad

4 lbs.

Macaroni Salad

3 lbs.

Cole Slaw

5 lbs.

Fruit Salad

4 lbs.

3-Bean Salad

2 lbs.

we could write a program and declare five separate integers…

int potatoSalad = 4;

int macaroniSalad = 3;

int coleSlaw = 5;

int fruitSalad = 4;

int threeBean = 2;

We might write a small program that would display the amount of salads we have on hand…

```
import java.util.*;

  public class salads_with_ints extends Object
  {
     public static void main(String[] args)
     {
        int potatoSalad = 4;
        int macaroniSalad = 3;
        int coleSlaw = 5;
        int fruitSalad = 4;
        int threeBean = 2;

        System.out.println("Salad tub 1 has " + potatoSalad + " pounds of salad.");
        System.out.println("Salad tub 2 has " + macaroniSalad + " pounds of salad.");
        System.out.println("Salad tub 3 has " + coleSlaw + " pounds of salad.");
        System.out.println("Salad tub 4 has " + fruitSalad + " pounds of salad.");
        System.out.println("Salad tub 5 has " + threeBean + " pounds of salad.");

     }
  }
```

we might inject some logic to warn us if things got low…

int potatoSalad = 4;

int macaroniSalad = 3;

int coleSlaw = 5;

int fruitSalad = 4;

int threeBean = 2;

Now, if we wanted to track when each salad reached 1 pound, we could write a program that would warn use whenever the weight of each salad was less-than-or-equal-to 1. Since these are five separate variables, we would have to check each one individually.

```java
import java.util.*;

  public class salads_with_ifs extends Object
  {
     public static void main(String[] args)
     {
        int potatoSalad = 4;
        int macaroniSalad = 3;
        int coleSlaw = 5;
        int fruitSalad = 4;
        int threeBean = 2;

        if(potatoSalad <= 1)
        {
           System.out.println("Salad tub 1 has " + potatoSalad + " pound of salad remaining. Time to make more!");
        }
        if(macaroniSalad <= 1)
        {
           System.out.println("Salad tub 2 has " + macaroniSalad + " pound of salad remaining. Time to make more!");
        }
        if(coleSlaw <= 1)
        {
           System.out.println("Salad tub 3 has " + coleSlaw + " pound of salad remaining. Time to make more!");
        }
        if(fruitSalad <= 1)
        {
           System.out.println("Salad tub 4 has " + fruitSalad + " pound of salad remaining. Time to make more!");
        }
        if(threeBean <= 1)
        {
           System.out.println("Salad tub 5 has " + threeBean + " pound of salad remaining. Time to make more!");
        }
     }
  }
```

But there's another way to reference the five salads…

Another way, would be to treat these *NOT* as individual salads, but a **COLLECTION** of salads stored in five tubs. We might call this collection **saladBar**

We could reference the *five* salads as a single collection…

Now instead of referencing them by their individual variable names, we could reference them by their unique position in the **saladBar** collection…

Our collection of five salads is called *saladBar*

Since in programming languages the numbering scheme always starts with 0, the first salad in **saladBar** is in *position 0*, and the last salad in **saladBar** is in *position 4*.

We would declare a *collection* called saladBar this way…

So, instead of declaring five separate variables all we have to do is declare one **saladBar**. In Java, the way to declare something that is going to represent a collection is to use square braces **[ ]** as part of the declaration


```
int [] saladBar;
```

This collection called saladBar is an array

This is an **array** which is programming-speak for a collection of things (or a group of things, or a list of things, *etc*).

Here the *square brackets* tell Java that we want to set up an *integer* array called **saladBar**, but they don't tell us *how many* things the array should hold.


```
int [] saladBar;
```

We also need to declare the size (or length) of the array

To do that, we have to set up a **new** array **object**.

In between the square brackets you need the pre-defined size (called ***length***) of the array. The length is how many positions the array should hold.


```
int [] saladBar;
saladBar = new int[5];
```

We can actually do the entire declaration on one line…

If you prefer, and most developers do, you can make the array declaration and set its length all on the same line.

Here we are declaring an array of integers named saladBar with a length of 5, meaning our array can hold 5 elements.


```
int [] saladBar = new int[5];
```

Now we can enter data into each element of the saladBar array…

Now to enter the data representing the weight of each of the salads in the saladBar collection, we would initiate each salad by its position in the array, like this:


```
int [] saladBar = new int[5];

saladBar[0] = 4; // weight of potato salad
saladBar[1] = 3; // weight of macaroni salad
saladBar[2] = 5; // weight of cole slaw
saladBar[3] = 4; // weight of fruit salad
saladBar[4] = 2; // weight of 3-bean salad
```

There's another way to declare and populate data in an array…

If we had a short array, and we knew the values of each of its elements going in, we could declare the array and populate (initialize) the data in the array all in one line with this kind of shortcut, called an **array literal**:


```
int [] saladBar = {4, 3, 5, 4, 2};
```

So, why would we want to use arrays over individual variables?

So, looking at all this, why would we ever decide to use an array over using individual variable declarations like we saw in Slide 4?

There's a very simple answer.

Because arrays are great to use with loops!

```
import java.util.*;

  public class salads extends Object
  {
     public static void main(String[] args)
     {
        int [] saladBar = new int[5];

        saladBar[0] = 4;
        saladBar[1] = 3;
        saladBar[2] = 5;
        saladBar[3] = 4;
        saladBar[4] = 2;

        // Of course you could have done it this way too: int [] saladBar = {4, 3, 5, 4, 2};

        for(int i = 0; i < saladBar.length; i++)
        {
           System.out.println("Salad tub " + (i + 1) + " has " + saladBar[i] + " pounds of salad.");
        }
     }
  }
```

The Output After Running:

```
Salad tub 1 has 4 pounds of salad.
Salad tub 2 has 3 pounds of salad.
Salad tub 3 has 5 pounds of salad.
Salad tub 4 has 4 pounds of salad.
Salad tub 5 has 2 pounds of salad.
```

http://faculty.cascadia.edu/cduckett/bit115/lecture_15/salads.java

And that's basically it for now!

This was a gentle introduction to arrays, how they are declared and initialized, and how they might be called and used programmatically

We will start looking at arrays in a more structured manner starting with the next lecture.