

## Arrays of Primitive Values

15-110 Summer 2010  
Margaret Reid-Miller

## Arrays

- Arrays are objects that hold multiple values of the **same type**.
- Each data value stored in an array is called an *element*.
- Each element is accessed using an integer *index* or *subscript*.
  - As with strings, the first subscript is 0.
- Organizing data in an array allows programs to access huge amount of data multiple times and in any order!

Summer 2010

15-110 (Reid-Miller)

2

## Array Objects

- Arrays are like objects and need to be created with the new operator:

```
int[] counts = new int[6];
```

Annotations:   
- `int[]`: type  
- `new int[6]`: creates the array  
- `6`: number of elements

- You can use any type: E.g.,

```
double[] lowTemp = new double[365];  
boolean[] isOn = new boolean[20];
```

must match

Summer 2010

15-110 (Reid-Miller)

3

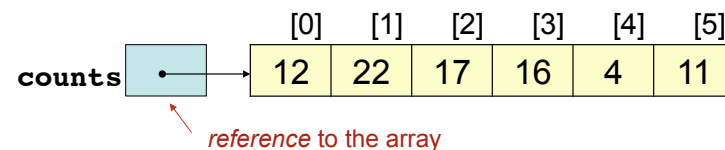
## Creating an Array

no length

```
int[] counts = new int[2*3];  
counts[0] = 12;  
counts[1] = 22;  
counts[2] = 17;  
counts[3] = 16;  
counts[4] = 4;  
counts[5] = 11;
```

length can be an int expression

bacteria colony count on 6 Petri dishes



Summer 2010

15-110 (Reid-Miller)

4

## Array Traversal

```
int sum = 0;
for (int i = 0; i < counts.length; i++) {
    sum = sum + counts[i];
}

if (counts.length > 0){
    System.out.print("The average number of " +
        "bacteria colonies is ");

    System.out.println(_____);
}
```

Constant: number of elements in the array

Not a method: There is no () !!

Summer 2010

15-110 (Reid-Miller)

5

## Array Basics

- Anywhere you can use a variable you can use an array element of the same type.

E.g., Suppose **y** is of type **int** and **x** is of type **int []**:

```
y = x[0] + x[1];
x[2] = x[0] / x[1];
x[1] += 2;
```

- Java treats **x[1]** as type **int** and it can be manipulated the same way as **y**.

Summer 2010

15-110 (Reid-Miller)

6

## Array Index

- The index of an array must be a literal, variable, or expression of type **int**.

E.g.,

<b>x[0] = 3;</b>	assign the first element 3;
<b>x[num] = num;</b>	use an <b>int</b> variable as index
<b>x[j-3] = x[j];</b>	use an <b>int</b> expression as index
<b>x[x[0]] = max;</b>	use an <b>int</b> array element as index

Summer 2010

15-110 (Reid-Miller)

7

## Bounds Checking

- Out-of-bounds* errors occur when you attempt to access an array element that does not exist.

index < 0 is out of bounds	[0] [1] [2] [3] [4] [5]	index > 5 is out of bounds
	12 22 17 16 4 11	

- If you write **x[index]** the compiler cannot determine if **index** is out of bounds.
- When you run the program, however, Java checks whether the index is out of bounds.

Summer 2010

15-110 (Reid-Miller)

8

## Bounds Checking

```
int sum = 0
for (int i = 0; i <= counts.length; i++) {
    sum += counts[i];
}
```

*When  $i$  equals `counts.length` and the program attempt to access `counts[i]`, Java raises an **ArrayIndexOutOfBoundsException***

- You need either to make sure the index stays within the bounds or to check that it is so before using it.

## Initializer List

- You can declare, create, and initialize an array with a **list of literals**.

```
int[] counts = {12, 22, 17, 16, 4, 11};
```

- You can use an initializer list only when the array is first declared.
- Each value must match the type of the array.
- The values go into the array in the order given and determine the length of the array.

## Example

Find the minimum value stored in an array:

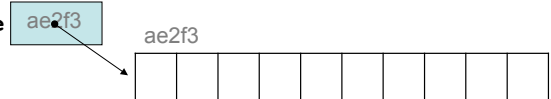
```
int min = counts[0] ;
for (int i = 1; i < counts.length; i++) {
    if (counts[i] < min) {
        min = counts[i];
    }
}
```

- What happens if there are two or more values in the array that are the minimum?
- How do we modify the code to return the index of the minimum bacteria count?

## Arrays as Parameters

```
→ int[] sequence = new int[10];
   fillAllSums(sequence);

... sequence
```



```
public static void fillAllSums(int[] seq) {
    seq[0] = 0;
    for (int i = 1; i < seq.length; i++) {
        seq[i] = seq[i-1] + i;
    }
}
```

## Arrays as Parameters

```
int[] sequence = new int[10];
→ fillAllSums(sequence);
... sequence ae2f3
    
```

0	1	3	6	10	15	21	28	36	45
---	---	---	---	----	----	----	----	----	----

```

public static void fillAllSums(int[] seq) {
    seq[0] = 0;
    for (int i = 1; i < seq.length; i++) {
        seq[i] = seq[i-1] + i;
    }
}
    
```

Summer 2010 15-110 (Reid-Miller) 13

## Arrays as Parameters

```
int[] sequence = new int[10];
fillAllSums(sequence);
→ ... sequence ae2f3
    
```

0	1	3	6	10	15	21	28	36	45
---	---	---	---	----	----	----	----	----	----

```

public static void fillAllSums(int[] seq) {
    seq[0] = 0;
    for (int i = 1; i < seq.length; i++) {
        seq[i] = seq[i-1] + i;
    }
}
    
```

Changes to an array in a method are visible outside the method!

Summer 2010 15-110 (Reid-Miller) 14

## Returning an Array

```
int[] sequence = getAllSums( 8 );
...
}
public static int[] getAllSums(int n) {
    int[] seq = new int[n];
    seq[0] = 0;
    for (int i = 1; i < n; i++) {
        seq[i] = seq[i-1] + i;
    }
    return seq;
}
    
```

Summer 2010 15-110 (Reid-Miller) 15

## The null reference

- An array variable holds the special value **null** if no array is created (distinct from an array with 0 elements.)

```
int array[] counts;    counts null
    
```

```

public static int[] copy(int[] data){
    if (data == null) return null;
    int[] data2 = new int[data.length];
    for (int i = 0; i < data.length; i++){
        data2[i] = data[i];
    }
    return data2;
}
    
```

Causes `NullPointerException` if data is null

Summer 2010 15-110 (Reid-Miller) 16