CSC220 (CSI)
Computational Problem Solving

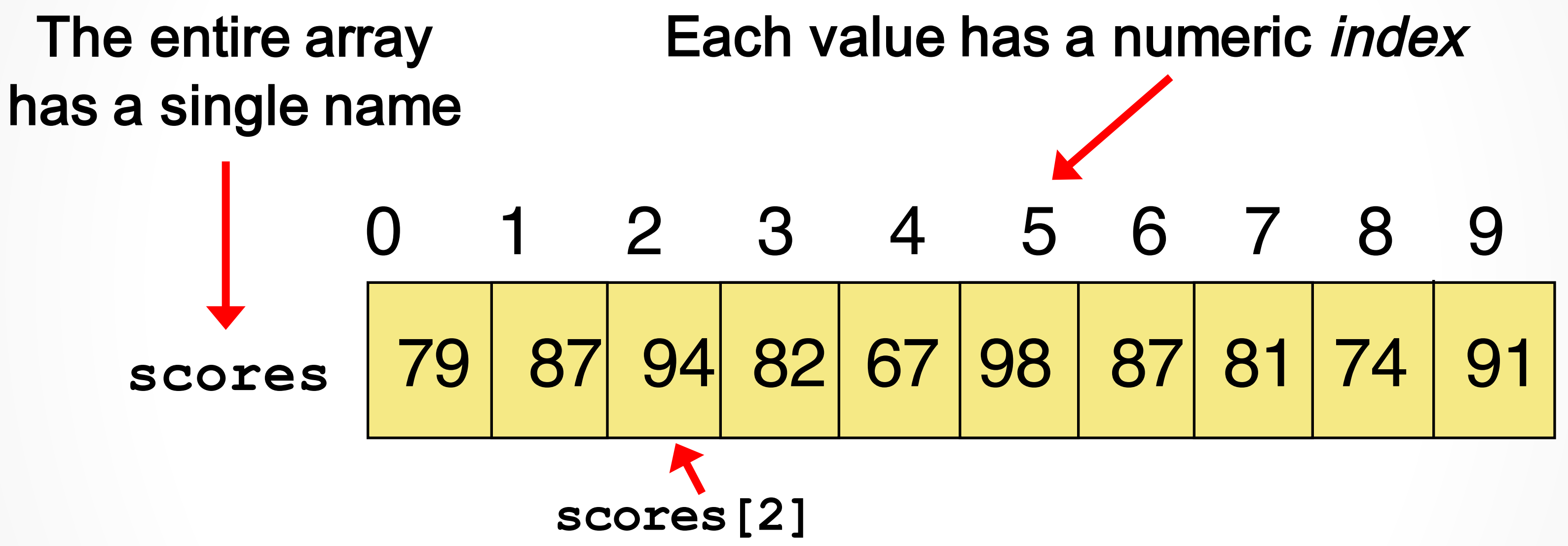
Arrays

The College of New Jersey

Please turn off your cell phone!

Arrays

- An array is an ordered list of values:


The entire array has a single name
Each value has a numeric *index*
0 1 2 3 4 5 6 7 8 9
scores
79 87 94 82 67 98 87 81 74 91
scores[2]

An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

- A particular value in an array is referenced using the array name followed by the index in brackets.

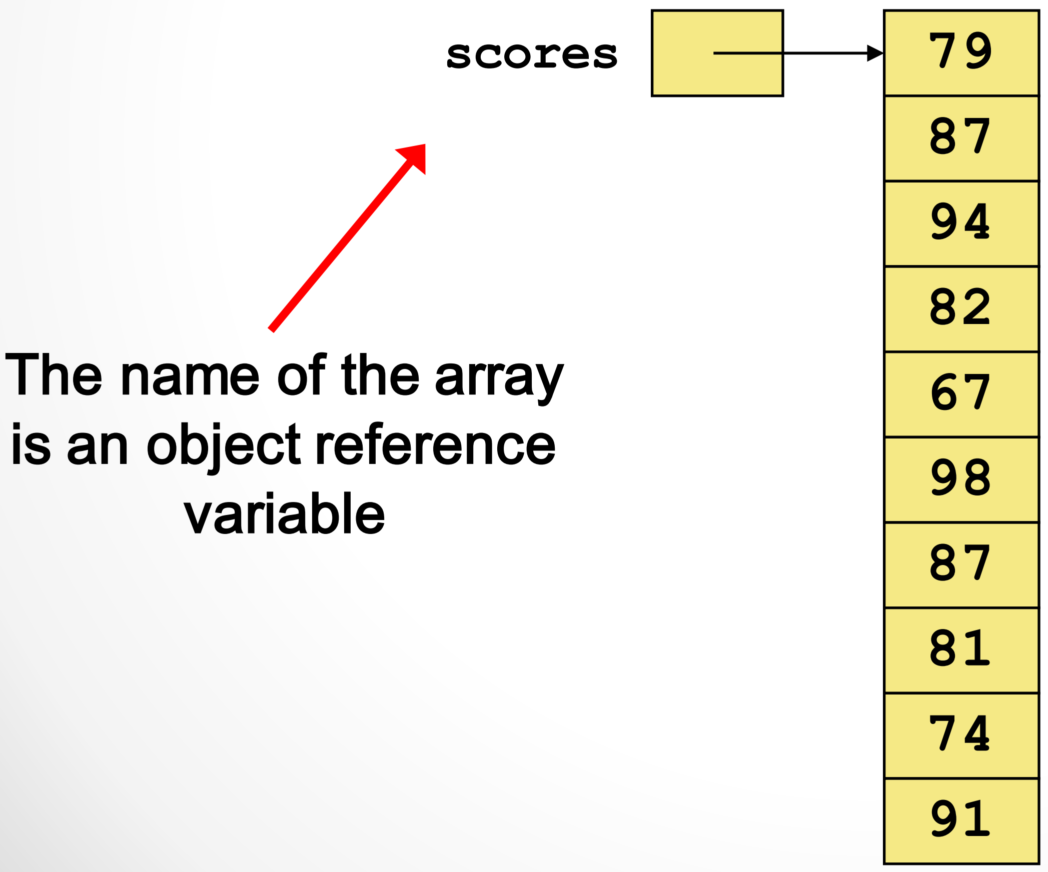
Arrays

- For example, an array element can be assigned a value, printed, or used in a calculation:

```
scores[2] = 89;

scores[first] = scores[first] + 2;

mean = (scores[0] + scores[1])/2;

System.out.println("Top = " + scores[5]);

pick = scores[rand.nextInt(11)];
```

Arrays

- In Java, the array itself is an object that must be instantiated
- Another way to depict the `scores` array:



Declaring Arrays

- The `scores` array could be declared as follows:

```
int[] scores = new int[10];
```

- The type of the variable `scores` is `int[]` (an array of integers)

- Some other examples of array declarations:

```
int[] weights = new int[2000];

double[] prices = new double[500];

boolean[] flags;
flags = new boolean[20];

char[] codes = new char[1750];
```

Using Arrays

- The for-each version of the `for` loop can be used when processing array elements:

```
for (int score : scores)
   System.out.println(score);
```

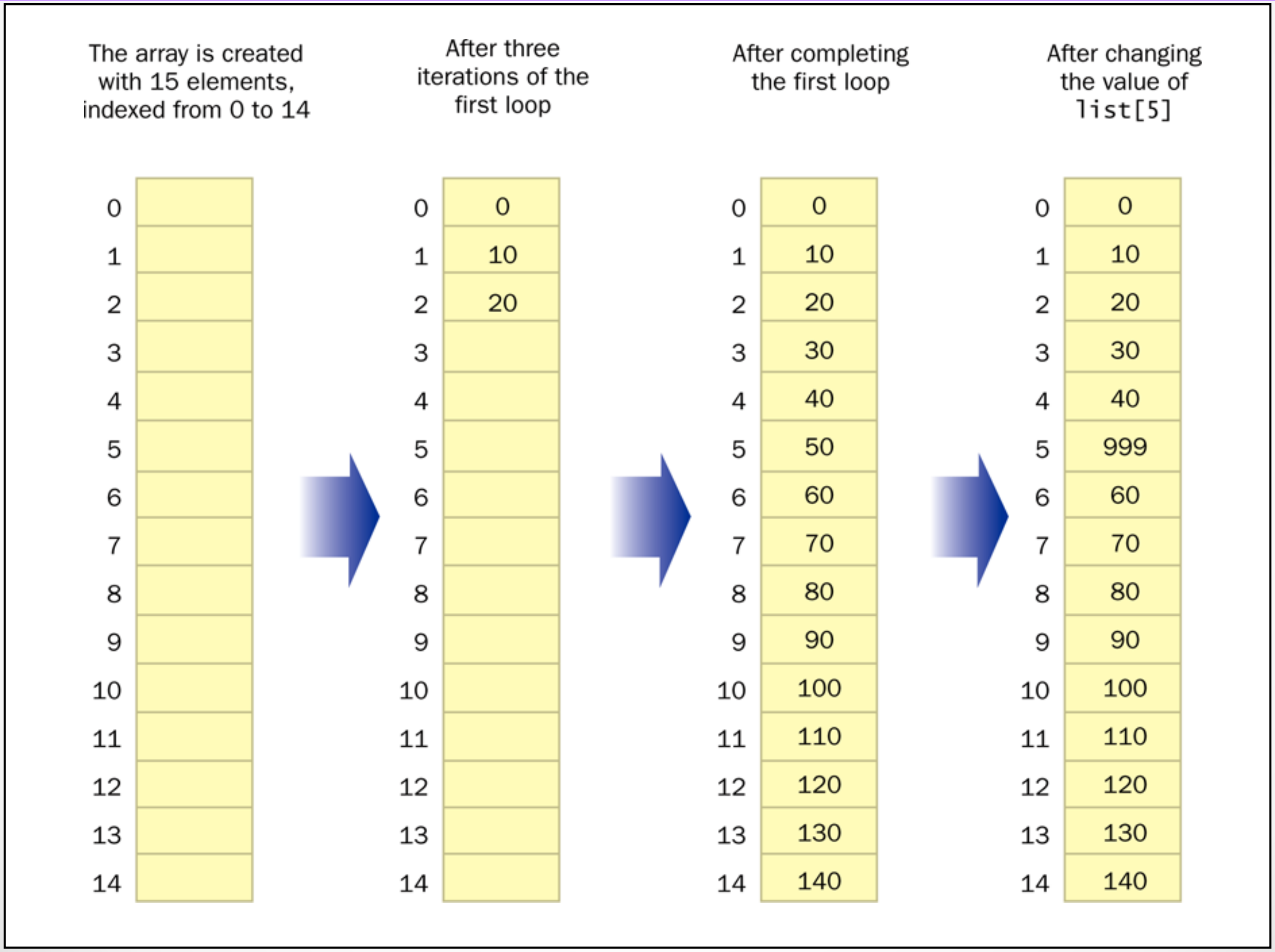
- This is only appropriate when processing all array elements starting at index 0
- It can't be used to set the array values
- See `BasicArray.java`

Output

```
0   10   20   30   40   999   60   70   80   90   100   110   120   130   140
```

```java
//********************************************************************

public class BasicArray
{
   //-----------------------------------------------------------------
   //  Creates an array, fills it with various integer values,
   //  modifies one value, then prints them out.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final int LIMIT = 15, MULTIPLE = 10;

      int[] list = new int[LIMIT];

      //  Initialize the array values
      for (int index = 0; index < LIMIT; index++)
         list[index] = index * MULTIPLE;

      list[5] = 999;  // change one array value

      //  Print the array values
      for (int value : list)
         System.out.print(value + "  ");
   }
}
```

Basic Array Example



	The array is created with 15 elements, indexed from 0 to 14	After three iterations of the first loop	After completing the first loop	After changing the value of `list[5]`
0		0	0	0
1		10	10	10
2		20	20	20
3			30	30
4			40	40
5			50	999
6			60	60
7			70	70
8			80	80
9			90	90
10			100	100
11			110	110
12			120	120
13			130	130
14			140	140

Quick Check

Write an array declaration to represent the ages of 100 children.

```
int[] ages = new int[100];
```

Write code that prints each value in an array of integers named `values`.

```
for (int value : values)
   System.out.println(value);
```

Bounds Checking

- Once an array is created, it has a fixed size
- The index value must be in range 0 to N-1
- The Java interpreter throws an `ArrayIndexOutOfBoundsException` if an array index is out of bounds
- This is called automatic *bounds checking*

problem

index < codes.length

```
for (int index=0; index <= 100; index++)
   codes[index] = index*50 + epsilon;
```

Sample Run

```
The size of the array: 10
Enter number 1: 18.36
Enter number 2: 48.9
Enter number 3: 53.5
Enter number 4: 29.06
Enter number 5: 72.404
Enter number 6: 34.8
Enter number 7: 63.41
Enter number 8: 45.55
Enter number 9: 69.0
Enter number 10: 99.18
The numbers in reverse order:
99.18  69.0  45.55  63.41  34.8  72.404  29.06  53.5  48.9  18.36
```

```
            System.out.print("Enter number " + (index+1) + ": ");
            numbers[index] = scan.nextDouble();
         }
         System.out.println("The numbers in reverse order:");
         for (int index = numbers.length-1; index >= 0; index--)
            System.out.print(numbers[index] + "  ");
      }
   }
```

```
//********************************************************************
//  LetterCount.java       Author: Lewis/Loftus
//
//  Demonstrates the relationship between arrays and strings.
//********************************************************************
import java.util.Scanner;
public class LetterCount{
   //-----------------------------------------------------------------
   //  Reads a sentence from the user and counts the number of
   //  uppercase and lowercase letters contained in it.
   //-----------------------------------------------------------------
   public static void main(String[] args){
      final int NUMCHARS = 26;
      Scanner scan = new Scanner(System.in);
      int[] upper = new int[NUMCHARS];
      int[] lower = new int[NUMCHARS];
      char current;   // the current character being processed
      int other = 0;  // counter for non-alphabetics

      System.out.println("Enter a sentence:");
      String line = scan.nextLine();
```

continue

```
      //  Count the number of each letter occurence
      for (int ch = 0; ch < line.length(); ch++){
         current = line.charAt(ch);
         if (current >= 'A' && current <= 'Z')
            upper[current-'A']++;
         else if (current >= 'a' && current <= 'z')
            lower[current-'a']++;
         else
            other++;
      }
      //  Print the results
      System.out.println();
      for (int letter=0; letter < upper.length; letter++){
         System.out.print( (char) (letter + 'A') );
         System.out.print(": " + upper[letter]);
         System.out.print("\t\t" + (char) (letter + 'a') );
         System.out.println(": " + lower[letter]);
      }

      System.out.println();
      System.out.println("Non-alphabetic characters: " + other);
   }
}
```

Sample Run

```
Enter a sentence:
In Casablanca, Humphrey Bogart never says "Play it again, Sam."

A:  0         a:  10
B:  1         b:  1
C:  1         c:  1
D:  0         d:  0
E:  0         e:  3
F:  0         f:  0
G:  0         g:  2
H:  1         h:  1
I:  1         i:  2
J:  0         j:  0
K:  0         k:  0
L:  0         l:  2
M:  0         m:  2
N:  0         n:  4
O:  0         o:  1
P:  1         p:  1
Q:  0         q:  0
```

continue

Sample Run (continued)

```
R:  0         r:  3
S:  1         s:  3
T:  0         t:  2
U:  0         u:  1
V:  0         v:  1
W:  0         w:  0
X:  0         x:  0
Y:  0         y:  3
Z:  0         z:  0

Non-alphabetic characters:  14
```

Initializer Lists

- An initializer list can be used to instantiate and fill an array in one step
- An initializer list can be used only in the array declaration
- The values are delimited by braces and separated by commas
- The size of the array is determined by the number of items in the list
- Examples:

```
int[] units = {147, 323, 89, 933, 540,
               269, 97, 114, 298, 476};

char[] grades = {'A', 'B', 'C', 'D', 'F'};
```

```java
//********************************************************************
//  Primes.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an initializer list for an array.
//********************************************************************

public class Primes
{
   //-----------------------------------------------------------------
   //  Stores some prime numbers in an array and prints them.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

      System.out.println("Array length: " + primeNums.length);

      System.out.println("The first few prime numbers are:");

      for (int prime : primeNums)
         System.out.print(prime + "  ");
   }
}
```

```java
//************
//  Primes.java
//
//  Demonstrate                                                  array.
//************                                                  ****************

public class Primes
{
   //-----------------------------------------------------------------
   //  Stores some prime numbers in an array and prints them.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int[] primeNums = {2, 3, 5, 7, 11, 13, 17, 19};

      System.out.println("Array length: " + primeNums.length);

      System.out.println("The first few prime numbers are:");

      for (int prime : primeNums)
         System.out.print(prime + "  ");
   }
}
```

Output

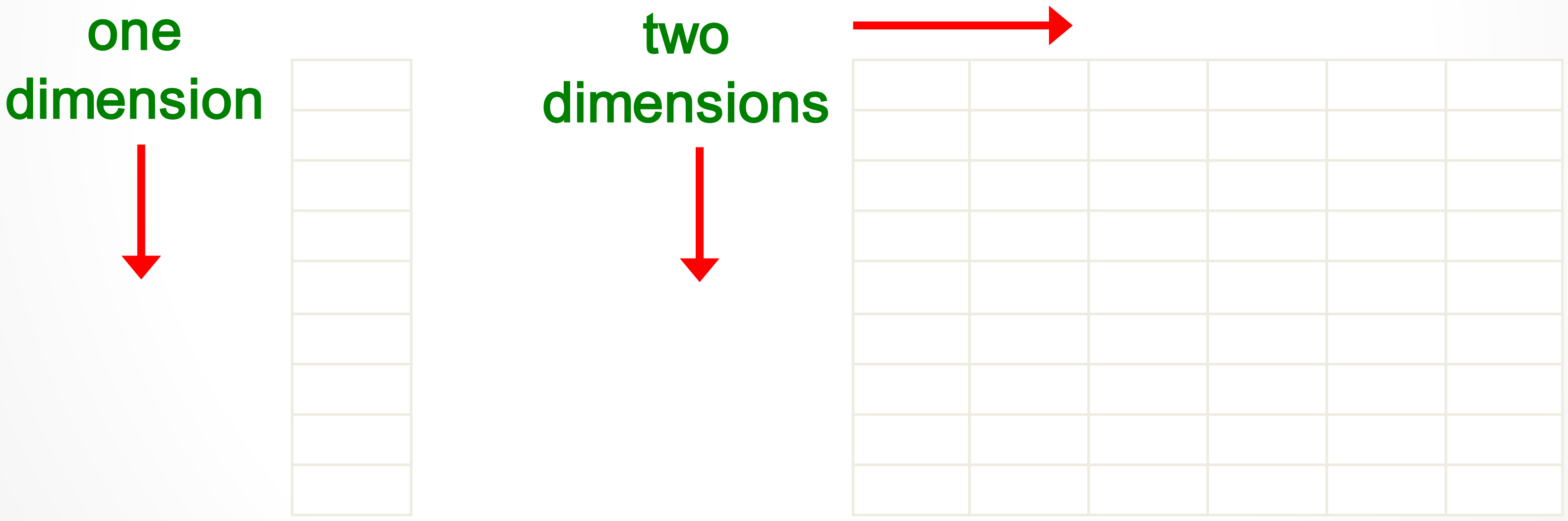
```
Array length: 8
The first few prime numbers are:
2  3  5  7  11  13  17  19
```

Arrays as Parameters

- An entire array can be passed as a parameter to a method
- Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other
- Therefore, changing an array element within the method changes the original
- An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type

Two-Dimensional Arrays

- A one-dimensional array stores a list of elements
- A two-dimensional array can be thought of as a table of elements, with rows and columns



- To be precise, in Java a two-dimensional array is an array of arrays

Two-Dimensional Arrays

- A two-dimensional array is declared by specifying the size of each dimension separately:
  - `int[][] table = new int[12][50];`
- A array element is referenced using two index values:
  - `value = table[3][6]`
- The array stored in one row can be specified using one index

Expression	Type	Description
`table`	`int[][]`	2D array of integers, or array of integer arrays
`table[5]`	`int[]`	array of integers
`table[5][12]`	`int`	integer

```
//********************************************************************
//  TwoDArray.java       Author: Lewis/Loftus
```

Output

```
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
```

```java
   public static void main(String[] args)
   {
      int[][] table = new int[5][10];

      // Load the table with values
      for (int row=0; row < table.length; row++)
         for (int col=0; col < table[row].length; col++)
            table[row][col] = row * 10 + col;

      // Print the table
      for (int row=0; row < table.length; row++)
      {
         for (int col=0; col < table[row].length; col++)
            System.out.print(table[row][col] + "\t");
         System.out.println();
      }
   }
}
```

```java
//********************************************************************
//  SodaSurvey.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a two-dimensional array.
//********************************************************************

import java.text.DecimalFormat;

public class SodaSurvey
{
   //-----------------------------------------------------------------
   //  Determines and prints the average of each row (soda) and each
   //  column (respondent) of the survey scores.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int[][] scores = { {3, 4, 5, 2, 1, 4, 3, 2, 4, 4},
                         {2, 4, 3, 4, 3, 3, 2, 1, 2, 2},
                         {3, 5, 4, 5, 5, 3, 2, 5, 5, 5},
                         {1, 1, 1, 3, 1, 2, 1, 3, 2, 4} };

      final int SODAS = scores.length;
      final int PEOPLE = scores[0].length;

      int[] sodaSum = new int[SODAS];
      int[] personSum = new int[PEOPLE];
```

continue

continue

```
      for (int soda=0; soda < SODAS; soda++)
         for (int person=0; person < PEOPLE; person++)
         {
            sodaSum[soda] += scores[soda][person];
            personSum[person] += scores[soda][person];
         }

      DecimalFormat fmt = new DecimalFormat("0.#");
      System.out.println("Averages:\n");

      for (int soda=0; soda < SODAS; soda++)
         System.out.println("Soda #" + (soda+1) + ": " +
                    fmt.format((float)sodaSum[soda]/PEOPLE));

      System.out.println ();
      for (int person=0; person < PEOPLE; person++)
         System.out.println("Person #" + (person+1) + ": " +
                    fmt.format((float)personSum[person]/SODAS));
   }
}
```

continue

```
      for (int soda=0;
         for (int perso                                person++)
         {
            sodaSum[soc                               son];
            personSum[p                               [person];
         }

      DecimalFormat fmt                               "0.#");
      System.out.printl

      for (int soda=0;
         System.out.pri                               +1) + ": " +
                     fmt                              m[soda]/PEOPLE));

      System.out.printl
      for (int person=0                               son++)
         System.out.pri                               rson+1) + ": " +
                     fmt                              Sum[person]/SODAS));
   }
}
```

Output

```
Averages:

Soda #1: 3.2
Soda #2: 2.6
Soda #3: 4.2
Soda #4: 1.9

Person #1: 2.2
Person #2: 3.5
Person #3: 3.2
Person #4: 3.5
Person #5: 2.5
Person #6: 3
Person #7: 2
Person #8: 2.8
Person #9: 3.2
Person #10: 3.8
```

Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a multidimensional array

- Each dimension subdivides the previous one into the specified number of elements

- Each dimension has its own `length` constant

- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths
  - these are sometimes called ragged arrays

Arrays of Objects

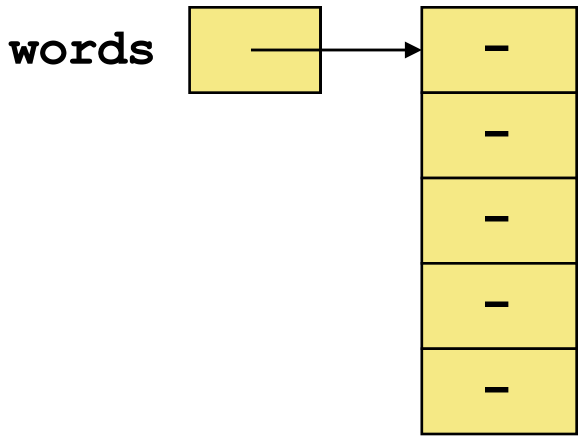
- The elements of an array can be object references
- The following declaration reserves space to store 5 references to String objects

```
String[] words = new String[5];
```

- It does not create the String objects themselves
- Initially an array of objects holds null references
- Each object stored in an array must be instantiated separately

Arrays of Objects

- The `words` array when initially declared

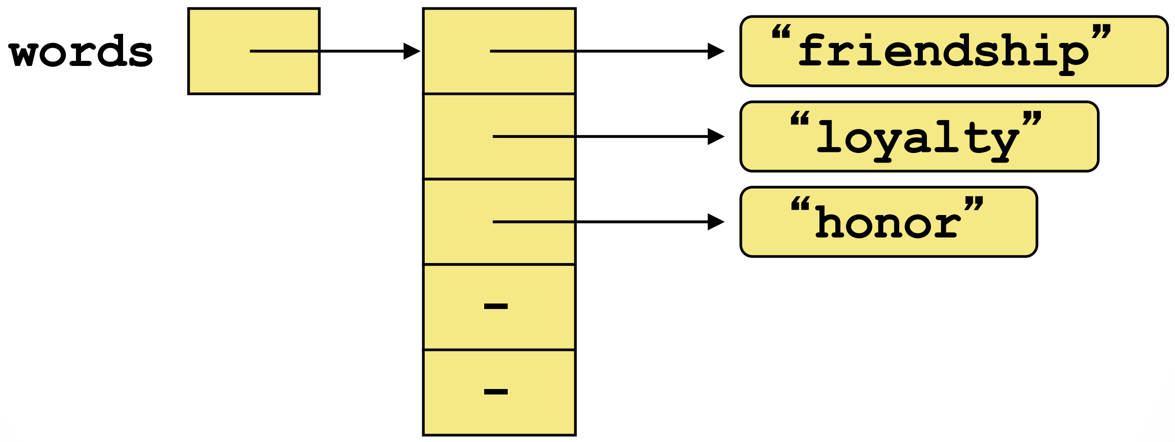


At this point, the following reference would throw a `NullPointerException`

```
System.out.println (words[0]);
```

Arrays of Objects

- After some `String` objects are created and stored in the array



Arrays of Objects

- Keep in mind that `String` objects can be created using literals

- The following declaration creates an array object called `verbs` and fills it with four `String` objects created using string literals

```
String[] verbs = {"play", "work", "eat", "sleep"};
```