Introduction to Arrays

Part 1: Creating, Initializing, Accessing

Course: CPSC 1150

Instructor: Dr. Bita Shadgar

Lecture 15

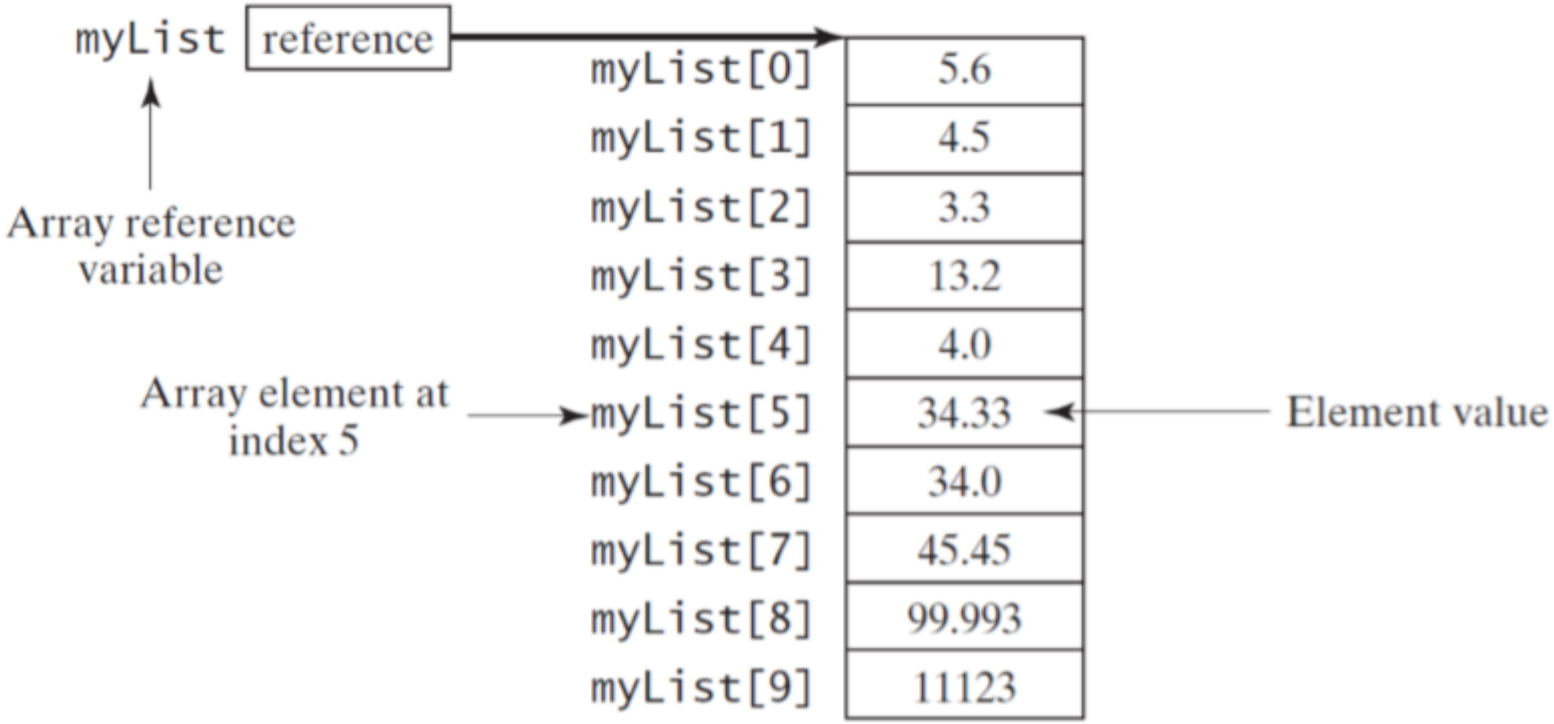
Learning Outcomes

- Justify the usage of arrays
- Define and create an array
- Declare and initialize an array
- Apply arrays in programming
- Access the array's element
- Use arrays to solve the problems

What is an array?

- List of variables of the same type
- Stored contiguously (all in one block) in memory
- Referenced with an identifier and an offset

```
double[] myList = new double[10];
```



Why arrays?

- Useful whenever you'd need a list to store information
  - For a list with 100 entries, instead of declaring 100 variables, only need to declare one
- Can pass an array reference into a method, whereas it's very difficult to pass 100 separate variables into a method
- Strings provide some of the same functionality, however, consider storing a list of `ints` in a String...
  - To go to the next int, can't just increment position by 1
  - To change a particular int, need to re-save the whole String
  - Need to perform String processing to go back and forth between `ints` and Strings
  - How could you access the 35th int in the String?

Declaring array variables

- An array variable declaration consists of an element **type**, the array symbol – [], and a **reference variable** identifier

Declaration example

```
int[] ages;
Boolean[] isAvailable;
```

- Note: ages is not an array
  - It is a variable that can reference an array
- An array reference variable can be assigned and reassigned to different arrays of the same type
- Different from declaring a primitive type
  - Declaring an array reference variable does not allocate any memory

Creating arrays

- Components of creating an array:
  - new keyword (like with Scanners)
  - Element type
  - Array length, in square brackets – length is fixed upon creation and can never be modified
- Memory is allocated now – how many bytes?
- Must assign the new array to a reference variable in order to access it again after creation

New array examples

```
ages = new int[15];
isAvailable = new boolean[4];
```

- It's common to combine declaring and creating an array:

```
Double[] prices = new double[30];
```

Accessing array elements

- An array element is accessed by the array reference variable and an index

Examples

```
prices[1] = 23.31;
prices[0] = prices[1] * 0.15;
```

- For an array of length N, indices range from 0 to N-1
- To get the length of an array, use the array's reference variable, followed by `.length`
  - For example, `prices.length` evaluates to 30
  - Unlike for `Strings`, this is not a method – no ()

Initial values

- When a new array is created, its elements are assigned default values
  - Numerical arrays get initialized to 0
  - char arrays get initialized to \u0000 (is known as null)
  - boolean arrays get initialized to false
- To set array values, can individually assign a value to each element as follows:

```
prices[0] = 13.99;
prices[1] = 23.31;
...
prices[29] = 14.90;
```

Quicker ways to initialize arrays

- If array values will follow some pattern, can usually initialize in a loop

```
int[] squares = new int[10];
for(int i = 0; i < squares.length; i++){
 squares[i] = i * i; //squares[i] holds i squared
}
```

- Otherwise, if the values are arbitrary, can use an array initializer to declare an array ref. variable, create a new array and initialize all its values at once
  - This is shorthand – must include declaration
  - Don't need new keyword, or array length

```
int[] temps = {12, 3, 1, 17, -3, -8, -18, 4, 6, 21};
```

Trace Program with Arrays

Declare array variable values, create an array, and assign its reference to values

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

i becomes 1

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

i (=1) is less than 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

After this line is executed, value[1] is 1

```
public class Test {
  public static void main(Str        args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

After i++, i becomes 2

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

```
public class Test {
  public static void main(String[]
      args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] +
      values[4];
  }
}
```

i (= 2) is less than 5

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

After this line is executed,
values[2] is 3 (2 + 1)

```
public class Test {
  public static void main(Str        args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this, i becomes 3.

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

i (=3) is still less than 5.

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this line, values[3] becomes 6 (3 + 3)

```
public class Test {
  public static void main(String[] ar
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

After this, i becomes 4

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

i (=4) is still less than 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

After this, values[4] becomes 10 (4 + 6)

```
public class Test {
  public static void main(Str[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

After i++, i becomes 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

i (=5) < 5 is false. Exit the loop

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

After this line, values[0] is 11 (1 + 10)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

0	11
1	1
2	3
3	6
4	10

For-each loops

- Special loop made for array traversal
- Goes through all array elements in order
- No need for an index

Example

```
for(double p: prices){
    System.out.print(p + " ");
}
```

- Element type must match array type
- **Note:** Cannot use a for-each loop to modify an array – elements are passed by value to the element parameter in the for-each loop

Examples

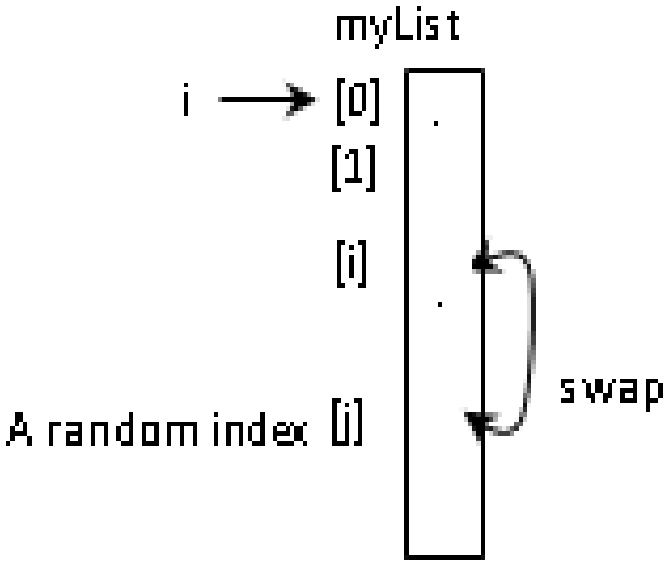
See the examples in the textbook (section 7.2.6).

1. (Initializing arrays with input values)
2. (Initializing arrays with random values)
3. (Printing arrays)
4. (Summing all elements)
5. (Finding the largest element)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)
8. (*Shifting elements*)

Example : Random shuffling

```
for (int i = 0; i < myList.length - 1; i++) {
  // Generate an index j randomly
  int j = (int)(Math.random() * myList.length);

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[j];
  myList[j] = temp;
}
```

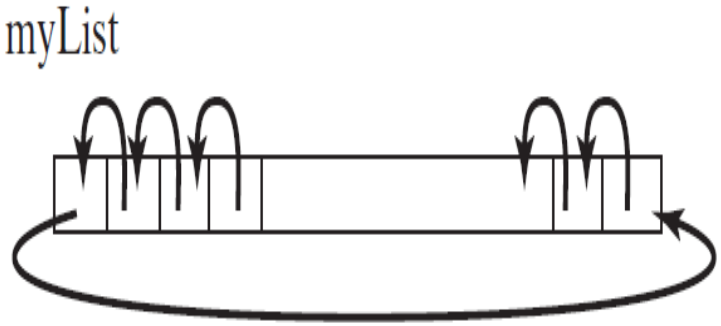


Example : Shifting Elements

```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```



More Exercise - Analyze Numbers

- Read one hundred numbers, compute their average, and find out how many numbers are above the average.