Introduction to Java for C++ Programmers

Segment - 2

JAC 444

Professor: Mahboob Ali

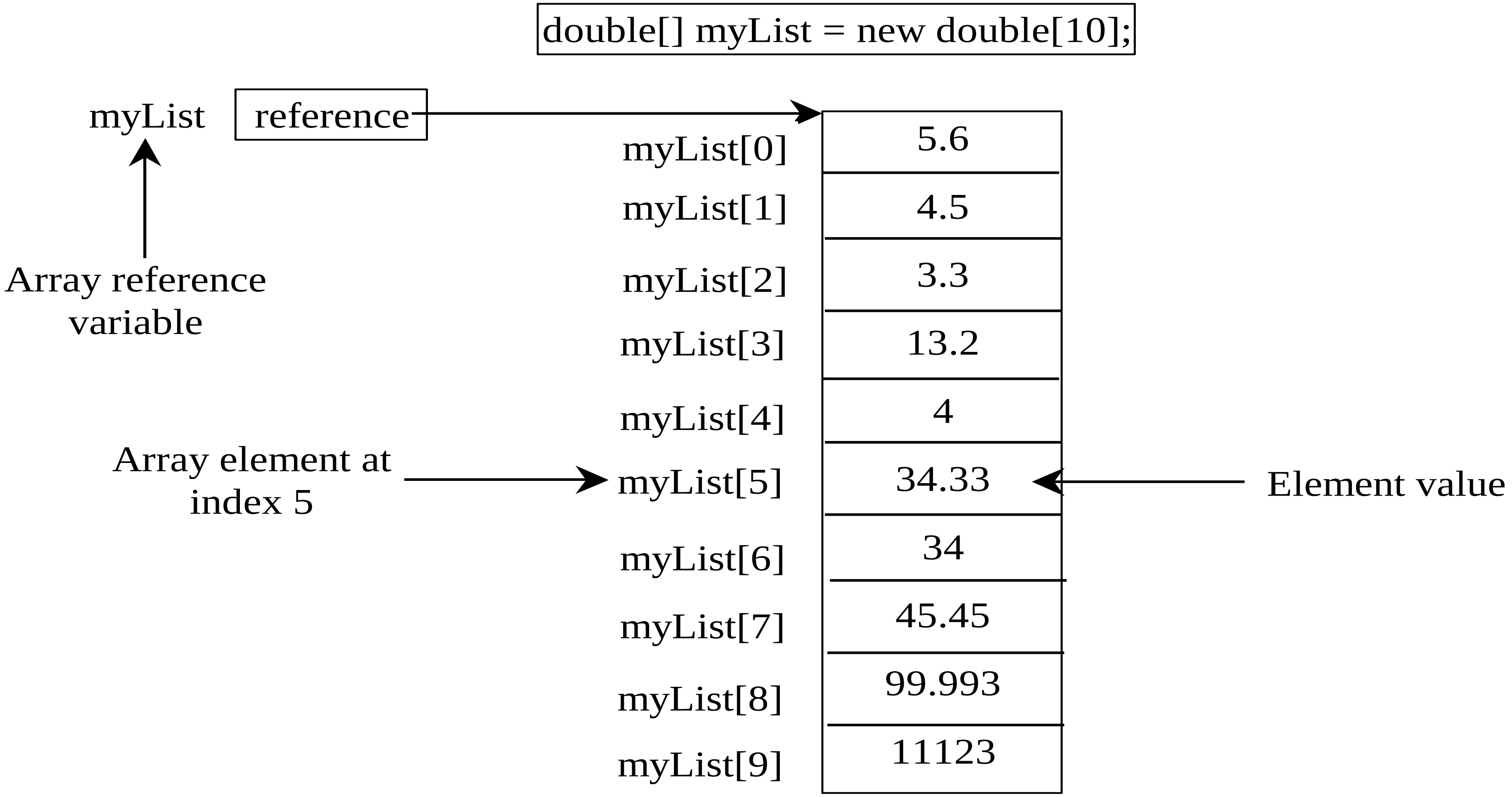
Objective

By the end of this segment students should be able to have understanding about:

- Single Dimension Arrays
- Double Dimension Arrays

Single Dimension Arrays

- Array is a data structure that represents a collection of the same types of data.



Syntax

- `datatype[] arrayRefVar = new datatype[arraySize];`

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

- `datatype arrayRefVar[] = new datatype[arraySize];`

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

- How to access the first element of the array?

`myList[0]` references the first element in the array.

- How to access the last element of the array?

`myList[9]` references the last element in the array.

- Once an array is created, its size is fixed. It cannot be changed. You can find its size using

```
arrayRefVar.length
```

- Default value of an array once declared

  0 for the numeric primitive data types,

  '\u0000' for char types, and

  false for boolean types.

- Each element in the array is represented using the following syntax, known as an *indexed variable*:

```
arrayRefVar[index];
```

Array Initializer

- 3 ways to initialize array elements.
  1. `double[] myList = {1.9, 2.9, 3.4, 3.5};`

     Declaring, creating and initializing arrays in one step

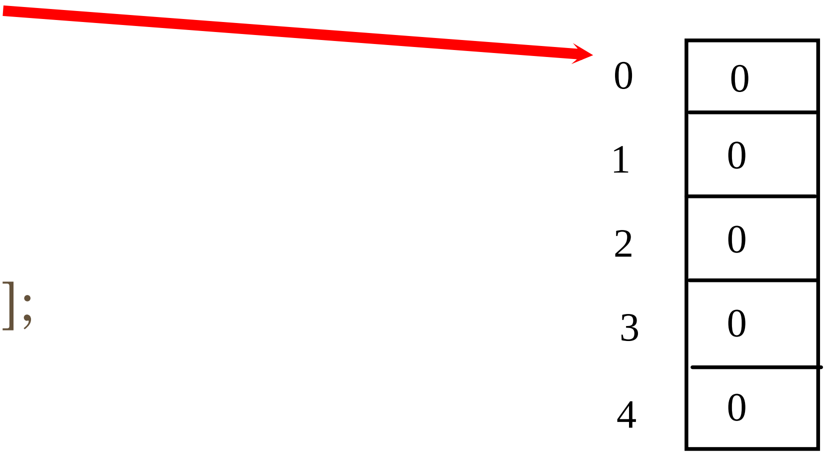
  2.
     ```
     double[] myList = new double[2];

     myList[0] = 1.9;

     myList[1] = 2.9;
     ```
  3. With the help of a for loop.

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

```
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

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

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];
  }
}
```

After i++, i becomes 2

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

i (= 2) 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];
  }
}
```

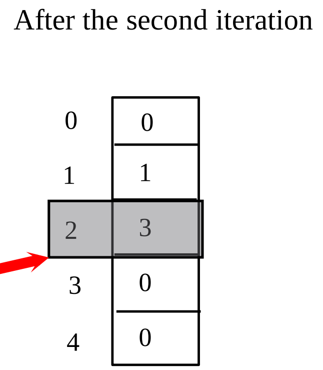
After the first iteration

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(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

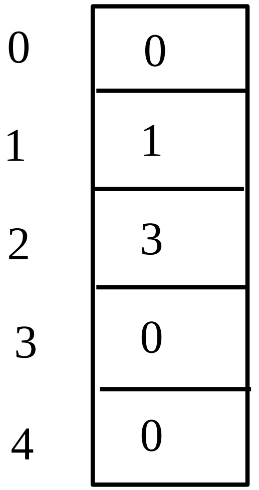
0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this, 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];
  }
}
```

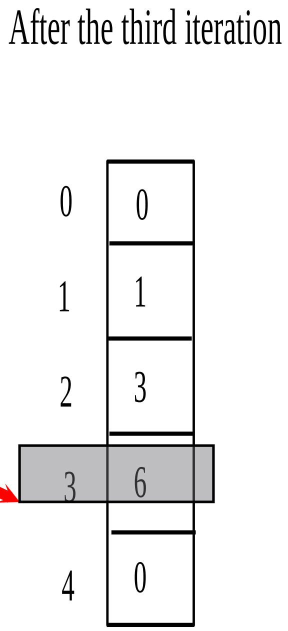
After the second iteration

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[]
       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, 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

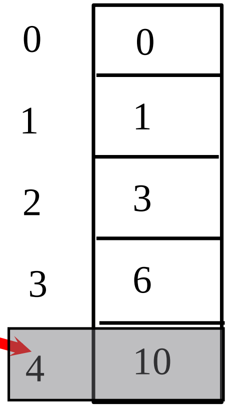
0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

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

```
public class Test {
  public static void main(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 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

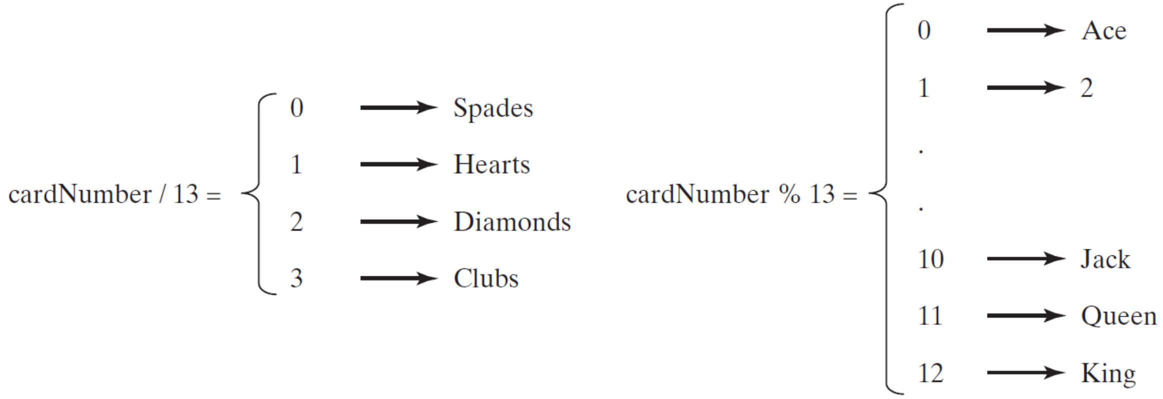
Problem: Deck of Cards

- The problem is to write a program that picks four cards randomly from a deck of 52 cards.

- All the cards can be represented using an array named deck, filled with initial values 0 to 52, as follows:

```
int[] deck = new int[52];
// Initialize cards
for (int i = 0; i < deck.length; i++)
  deck[i] = i;
```

```
public class DeckOfCards {
  public static void main(String[] args) {
    int[] deck = new int[52];
    String[] suits = {"Spades", "Hearts", "Clubs",
"Diamonds"};
    String[] ranks = {"Ace", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "Jack", "Queen", "King"};

    // Initialize cards
    for (int i = 0; i < deck.length; i++)
      deck[i] = i;

    // Shuffle the cards
    for (int i = 0; i < deck.length; i++) {
      // Generate an index randomly
      int index = (int)(Math.random() * deck.length);
      int temp = deck[i];
      deck[i] = deck[index];
      deck[index] = temp;
    }

    // Display the first four cards
    for (int i = 0; i < 4; i++) {
      String suit = suits[deck[i] / 13];
      String rank = ranks[deck[i] % 13];
      System.out.println("Card number " + deck[i] + ": "
        + rank + " of " + suit);
    }
  }
}
```



cardNumber / 13 =

0 → Spades

1 → Hearts

2 → Diamonds

3 → Clubs

cardNumber % 13 =

0 → Ace

1 → 2

.

.

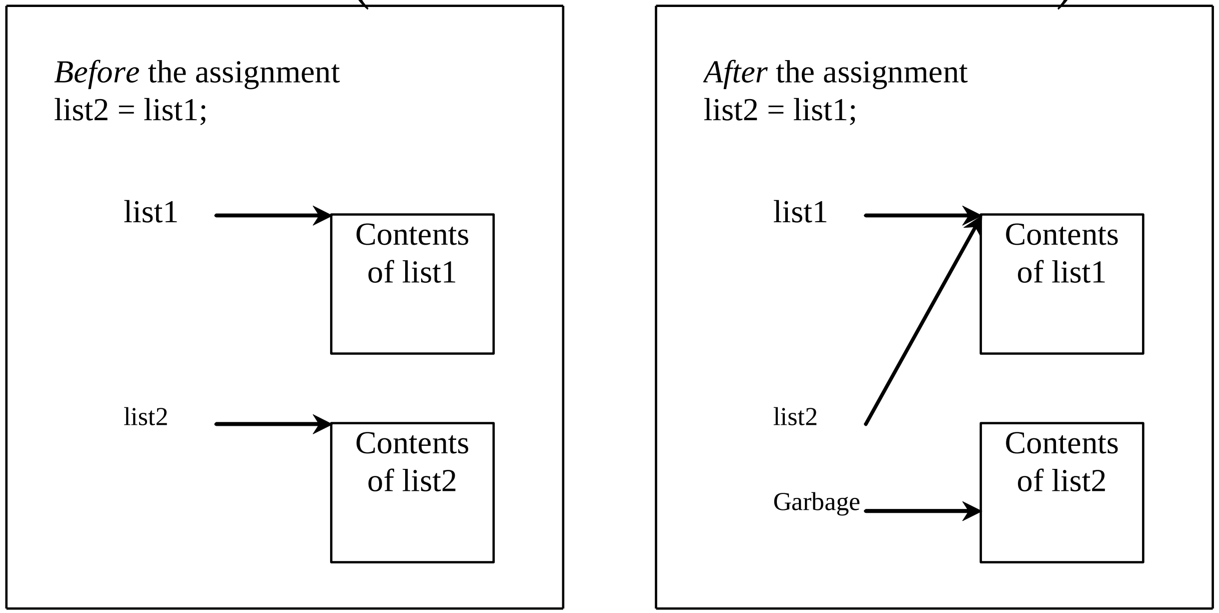
10 → Jack

11 → Queen

12 → King

Copying an Array

- You can use assignment statements to copy primitive data type variables not arrays.
- Three ways to copy an array in java:
  1. Use a loop to individual elements one by one.
  2. Use a static **arraycopy** method in the **System** class.
  3. Use the clone method (will be introduced later).



Using a loop:

```
int[] sourceArray = {2, 3, 1, 5, 10};

int[] targetArray = new int[sourceArray.length];


for (int i = 0; i < sourceArrays.length; i++)

   targetArray[i] = sourceArray[i];
```

Use arraycopy Method:

```
arraycopy(sourceArray, src_pos, targetArray, tar_pos, length);
```

Example:

```
System.arraycopy(sourceArray, 0, targetArray, 0,
 sourceArray.length);
```

Passing Arrays to the Methods

- Java uses ***pass by value*** to pass arguments to a method. There are important differences between passing a value of variables of primitive data types and passing arrays.

➢ For a parameter of a primitive type value, the actual value is passed. Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.

➢ For a parameter of an array type, the value of the parameter contains a reference to an array; this reference is passed to the method. Any changes to the array that occur inside the method body will affect the original array that was passed as the argument.

```
public class Test {
  public static void main(String[] args) {
    int x = 1; // x represents an int value
    int[] y = new int[10]; // y represents an array of int values

    m(x, y); // Invoke m with arguments x and y

    System.out.println("x is " + x);
    System.out.println("y[0] is " + y[0]);
  }

  public static void m(int number, int[] numbers) {
    number = 1001; // Assign a new value to number
    numbers[0] = 5555; // Assign a new value to numbers[0]
  }
}
```

```
x is 1
y[0] is 5555
```

Passing Arrays to Methods

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

Invoke the method

```
int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

Invoke the method

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

No Explicit reference

Returning an Array

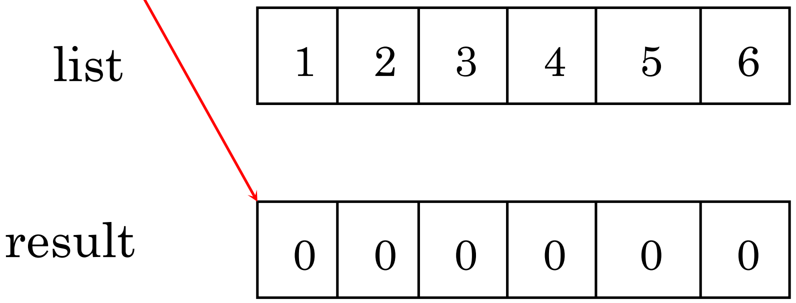
- When a method return an array, the reference of the array is returned.

Trace the reverse Method

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

Declare result and create array

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



list

1	2	3	4	5	6

result

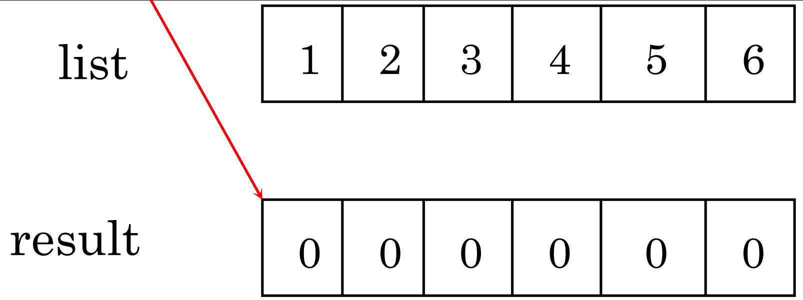
0	0	0	0	0	0

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

i = 0 and j = 5

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

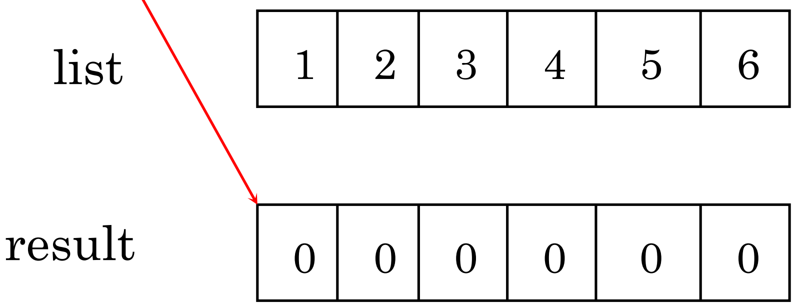


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (= 0) is less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



list 1 2 3 4 5 6

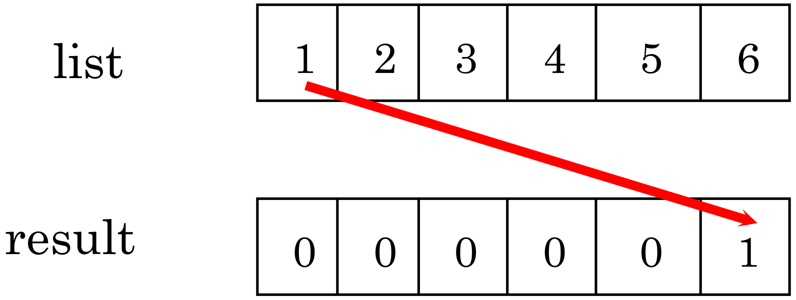
result 0 0 0 0 0 0

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 0 and j = 5
Assign list[0] to
result[5]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

After this, i becomes 1 and j becomes 4

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6
result	0	0	0	0	0	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=1) is less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

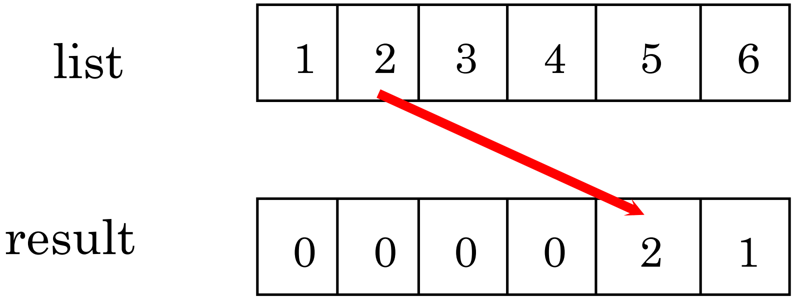
list	1	2	3	4	5	6
result	0	0	0	0	0	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i = 1 and j = 4
Assign list[1] to
result[4]

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



list	1	2	3	4	5	6
result	0	0	0	0	2	1

Trace the reverse Method, cont.

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

After this, i becomes 2 and j becomes 3

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length   1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list						
	1	2	3	4	5	6

result						
	0	0	0	0	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=2) is still less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

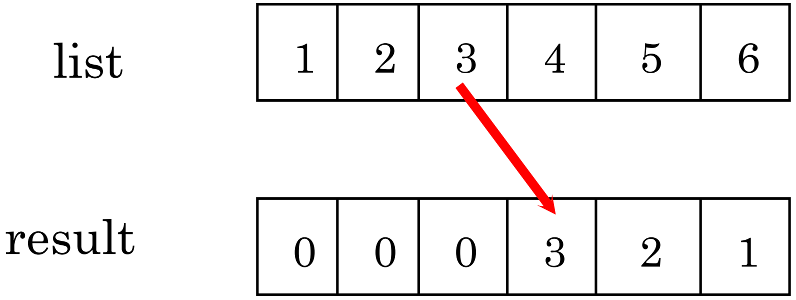
list	1	2	3	4	5	6
result	0	0	0	0	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i = 2 and j = 3
Assign list[i] to result[j]

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

After this, i becomes 3 and j becomes 2

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list

1	2	3	4	5	6

result

0	0	0	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=3) is still less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

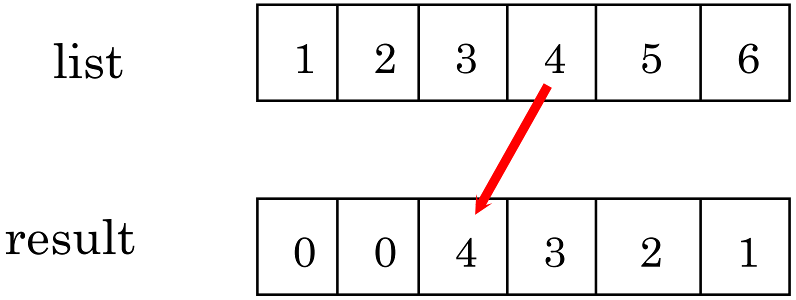
list	1	2	3	4	5	6
result	0	0	0	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 3 and j = 2
Assign list[i] to result[j]



list	1	2	3	4	5	6
result	0	0	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

After this, i becomes 4 and j becomes 1

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6
result	0	0	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=4) is still less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6

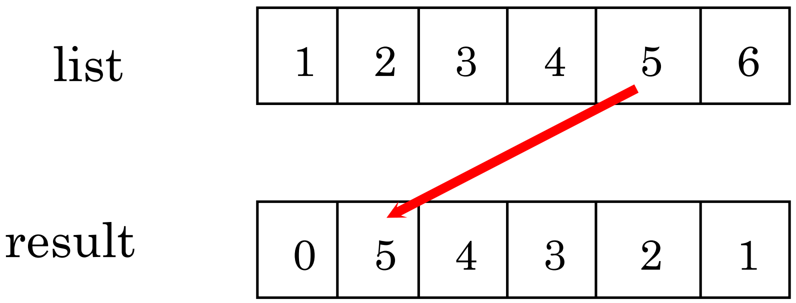
result	0	0	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 4 and j = 1
Assign list[i] to result[j]



list	1	2	3	4	5	6
result	0	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

After this, i becomes 5 and j becomes 0

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6
result	0	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=5) is still less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length]

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6

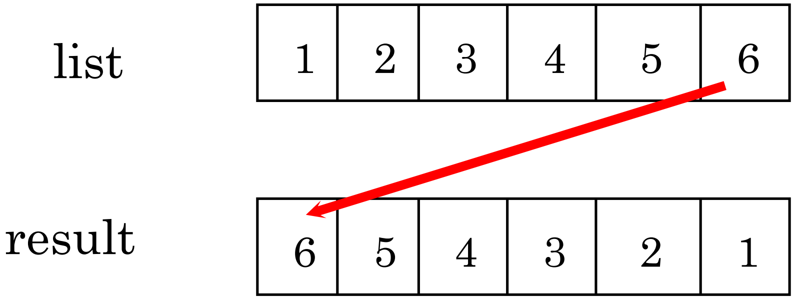
result	0	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 5 and j = 0
Assign list[i] to result[j]



list 1 2 3 4 5 6

result 6 5 4 3 2 1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

After this, i becomes 6 and j becomes -1

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6

result	6	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};

int[] list2 = reverse(list1);
```

i (=6) < 6 is false. So exit the loop.

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length]

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

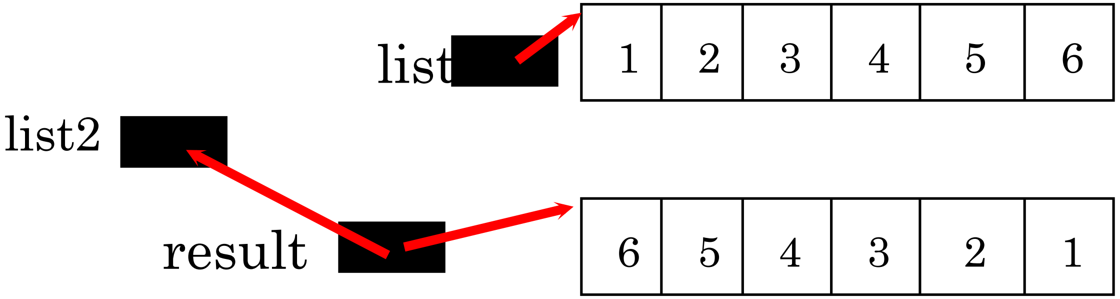
list	1	2	3	4	5	6
result	6	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Return result

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



Disadvantages

- **Size Limit:** We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

Enhanced for Statement

- Iterates through the elements of an array *without* using a counter, thus avoiding the possibility of “stepping outside” the array.
- Syntax:
    - for (*parameter* : *arrayName*)
      statement
  - where *parameter* has a type and an *identifier*, and *arrayName* is the array through which to iterate.
  - Parameter type must be consistent with the type of the elements in the array.

```
public class EnhancedFor{
     public static void main(String[] args){
           int[] array = {87, 68, 94, 100, 83, 78, 85, 91};
           int total = 0;

           //add each element’s value to the total
           for(int number : array){
                total += number;
           }
           System.out.println(“Total of the array elements:
“ + total);
     }
}
```

Multidimensional Arrays

Syntax

```
// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];


// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

Illustration

	0	1	2	3	4
0					
1					
2					
3					
4					

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

	0	1	2	3	4
0					
1					
2		7			
3					
4					

```
matrix[2][1] = 7;
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

matrix.length? 5

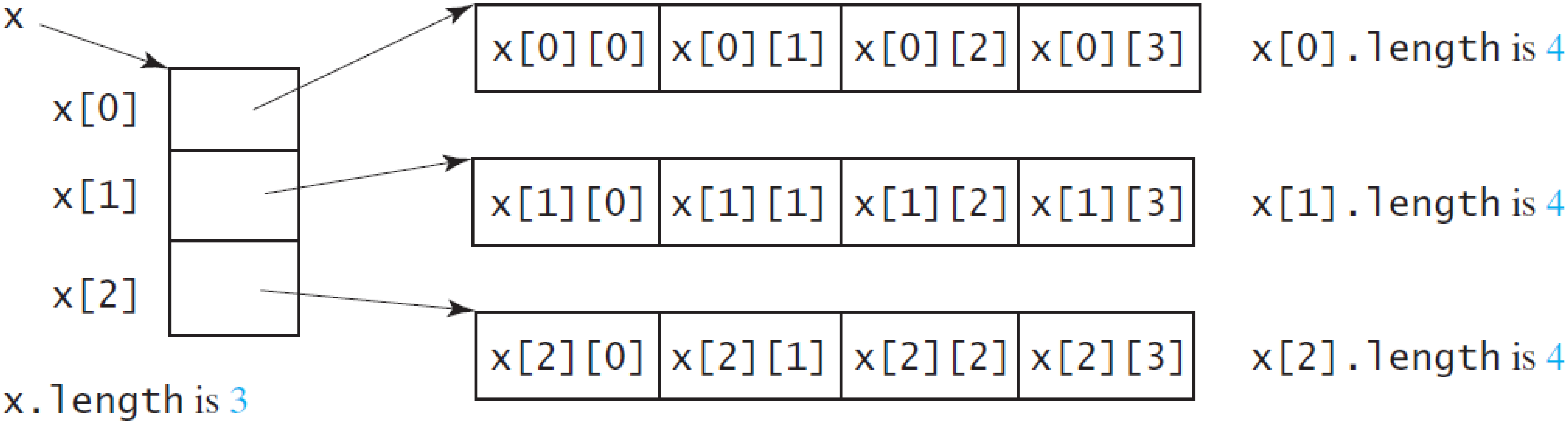
matrix[0].length? 5

array.length? 4

array[0].length? 3

Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```



x

x[0]

x[1]

x[2]

x.length is 3

x[0][0]	x[0][1]	x[0][2]	x[0][3]	x[0].length is 4
x[1][0]	x[1][1]	x[1][2]	x[1][3]	x[1].length is 4
x[2][0]	x[2][1]	x[2][2]	x[2][3]	x[2].length is 4

Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

Initializing arrays with input values

```
java.util.Scanner input = new Scanner(System.in);


System.out.println("Enter " + matrix.length + " rows and
   " +  matrix[0].length + " columns: ");


for (int row = 0; row < matrix.length; row++) {

  for (int column = 0; column < matrix[row].length;
   column++) {

    matrix[row][column] = input.nextInt();

  }

}
```

Initializing arrays with random values

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length;
    column++) {
    matrix[row][column] = (int)(Math.random() * 100);
  }
}
```

Printing arrays

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length;
    column++) {
     System.out.print(matrix[row][column] + " ");
  }


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