Arrays

CSE 114: Introduction to Object-Oriented Programming

Paul Fodor

Stony Brook University

http://www.cs.stonybrook.edu/~cse114

Contents

- Motivation: Opening Problem
- Declaring Array Variables
- Creating Arrays
- length and Indexed Variables
- Default values
- Shortcut initialization
- Common algorithms with arrays: initialization, printing, sum, min, max, shuffling, shifting, copying
- Enhanced for-Loops (for-each loops)
- Passing and returning arrays to/from methods (Pass By Value)
  - The Heap Segment and the Call Stack Memory
  - Returning an Array from a Method, reverse an array
- Searching Arrays: Linear and Binary Search
- Sorting Arrays: Selection Sort and Insertion Sort

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Opening Problem

- Read one hundred numbers (e.g., temperatures, units sold), compute their average, and find out how many numbers are above the average
  - Problems:
    - you need to read all the numbers before you can compute the average and compare them
    - storing 100 numbers
    - variables with one number each?

      NO.
  - Solution:
    - arrays: one variable for all 100 numbers

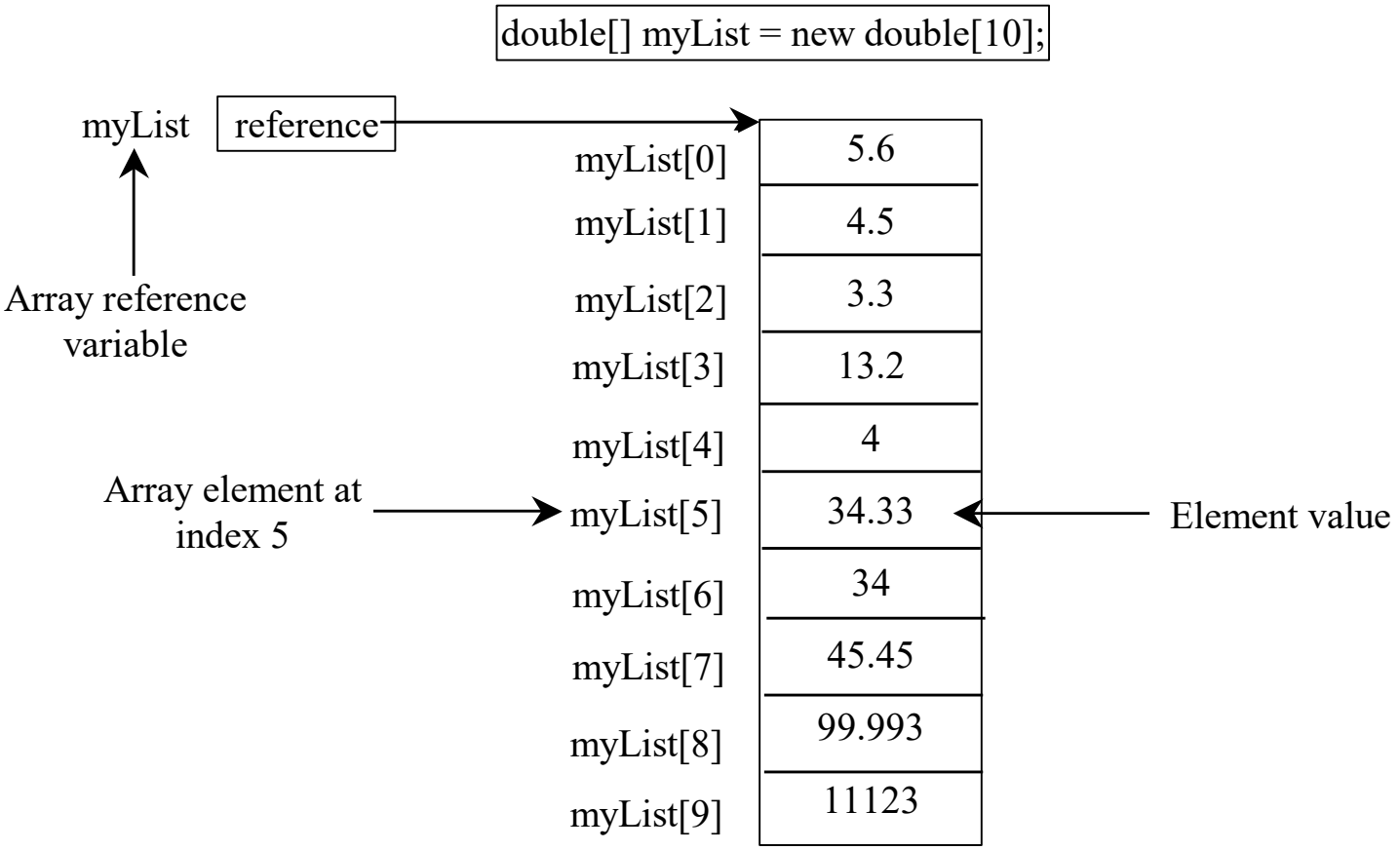
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

```
import java.util.Scanner;
public class Test {
  public static void main(String[] args) {
    double[] myList;
    int n = 100;
    myList = new double[n];
    Scanner input = new Scanner(System.in);
    System.out.print("Enter "+myList.length+" doubles: ");
    double sum = 0;
    for (int i = 0; i < myList.length; i++) {
      myList[i] = input.nextDouble();
      sum += myList[i];
    }
    double avg = sum / myList.length;
    int counter = 0;
    for (double v : myList)
      if (v > avg)
        counter++;
    System.out.println(counter
      +" are greater than the average "+avg);
  }
}
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Introducing Arrays

An *array* is a data structure that represents a collection of the same type of data.



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Declaring Array Variables

- `elementDatatype[] arrayRefVar;`

  Example:

  ```
  double[] myList;
  ```

- `elementDatatype arrayRefVar[];`

  // This style is allowed in Java, but **NOT** preferred

  Example:

  ```
  double myList[];
  ```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Creating Arrays

```
arrayRefVar = new elementDatatype[arraySize];
```

- Examples:

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

double[] myList2;
int size = 10;
myList2 = new double[size];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Declaring and Creating in One Step

- `elementDatatype[] arrayRefVar = new elementDatatype[arraySize];`
  - Example:

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

Old syntax:

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


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

The Length of an Array

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

  `arrayRefVar.length`

- Example:

  `myList.length` returns 10

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Default Values

- When an array is created, its elements are assigned the default value of

  `0` for the numeric primitive data types,

  `'\u0000'` for char types, and

  `false` for boolean types, and

  `null` for reference types

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Indexed Variables

- The array elements are accessed through the index.
  - The array indices are *0-based*, i.e., it starts from `0` to `arrayRefVar.length - 1`.
  - Each element in the array is represented using the following syntax, known as an *indexed variable*:

```
arrayRefVar[index]
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Using Indexed Variables

After an array is created, an indexed variable can be used in the same way as a regular variable.

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

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

Examples:

```
myList[0] = 1;
myList[1] = 2;
myList[2]= myList[0] + myList[1];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Array Initializers

- Declaring, creating, initializing in one step:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

  - This shorthand syntax must be in one statement!

This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];

myList[0] = 1.9;

myList[1] = 2.9;

myList[2] = 3.4;

myList[3] = 3.5;
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Array Initializers

- Another syntax:

```
double[] myList = new double[]
          {1.9, 2.9, 3.4, 3.5};


double[] myList2;
myList2 = new double[]{1.9, 2.9, 3.4, 3.5};
// it is required if the assignment is
//  separate from the declaration
```

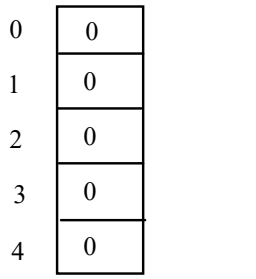

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



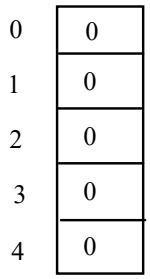
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



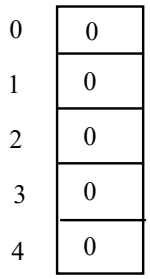
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



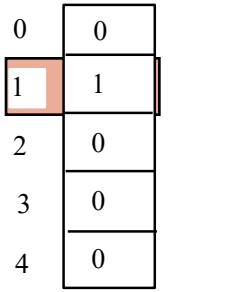
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace Program with Arrays

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



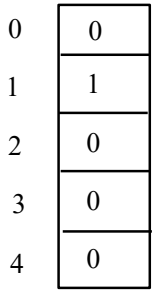
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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

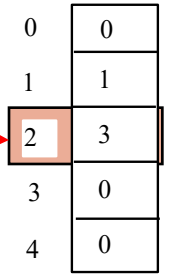
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



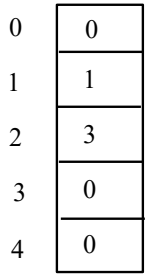
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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

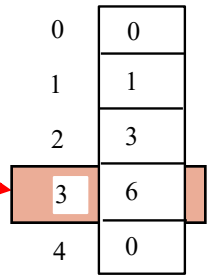
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



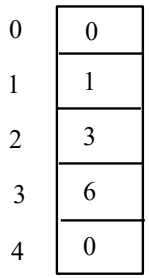
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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 (=4) is still less than 5

After the third iteration

0	0
1	1
2	3
3	6
4	0

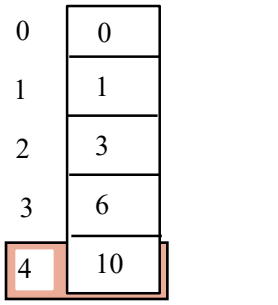
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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 5

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace Program with Arrays

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

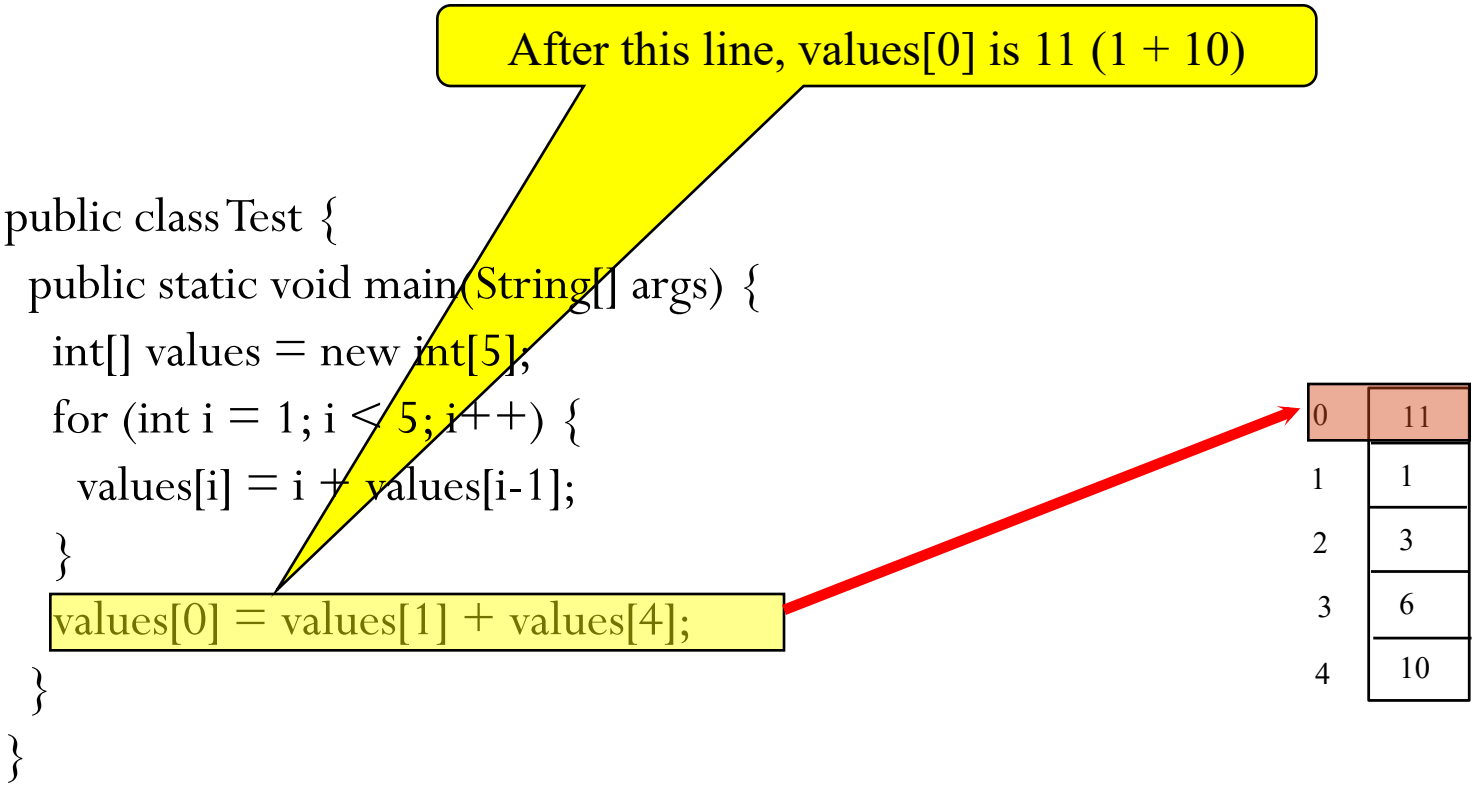
```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

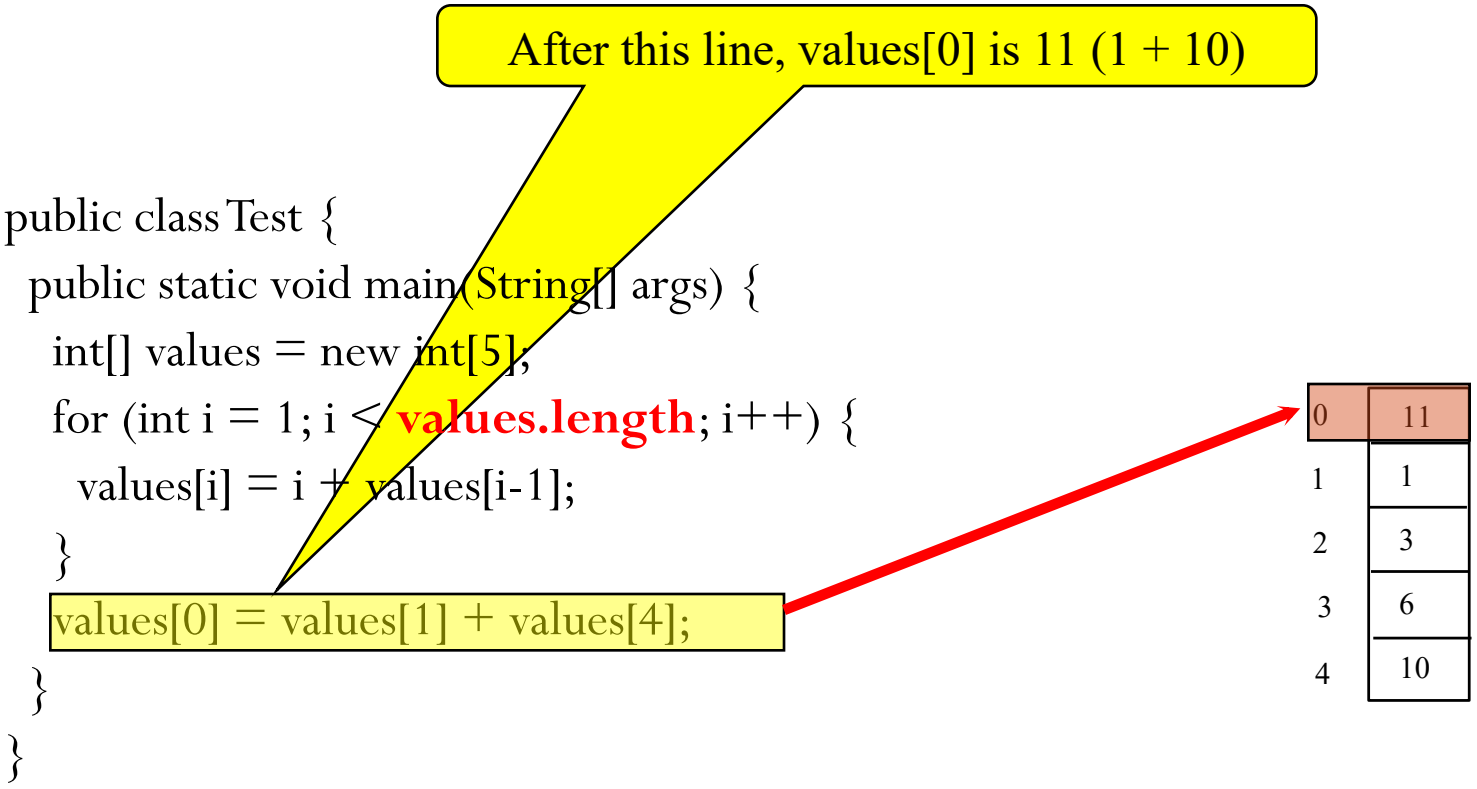
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace Program with Arrays



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace Program with Arrays



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Initializing arrays with input values

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

Scanner input = new Scanner(System.in);

System.out.print("Enter " + myList.length
   + " values: ");

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

  myList[i] = input.nextDouble();
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Initializing arrays with random values

```
double[] myList = new double[10];
for(int i = 0; i < myList.length; i++)
  myList[i] = Math.random() * 100;
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Printing arrays

```
double[] myList = new double[10];
...
for(int i = 0; i < myList.length; i++)
  System.out.print(myList[i] + " ");
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Summing all elements

```
double[] myList = new double[10];
...
double total = 0;
for(int i = 0; i < myList.length; i++)
  total += myList[i];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Finding the largest element

```
double max = myList[0];
for(int i = 1; i < myList.length; i++)
  if (myList[i] > max)
        max = myList[i];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

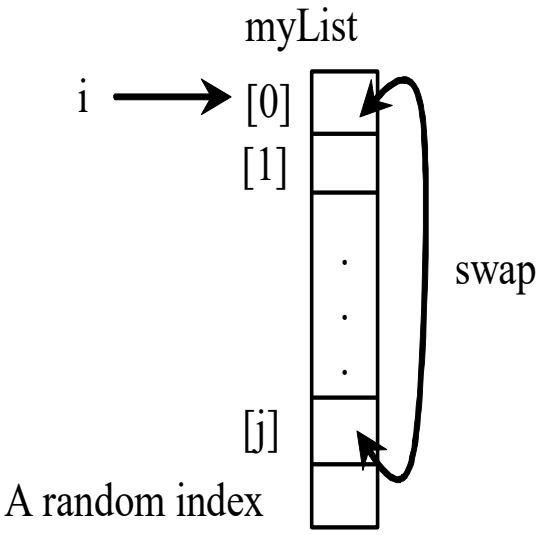
Finding the smallest element

```
double min = myList[0];
for(int i = 1; i < myList.length; i++)
  if (myList[i] < min)
      min = myList[i];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Random shuffling

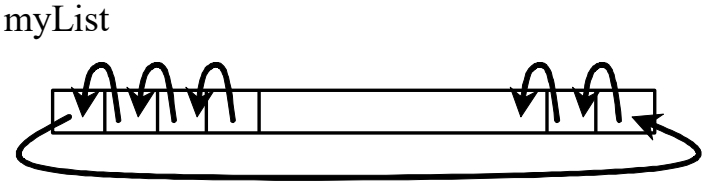
```
for (int i = 0; i < myList.length; i++) {
  // Generate an index j randomly
  int j = (int)(Math.random()
    * myList.length);

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



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Shifting Left

```
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;
```



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Shifting Right

```
double temp = myList[myList.length - 1];
for(int i=myList.length-1; i>0; i--)
     myList[i] = myList[i-1];
myList[0] = temp;
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Enhanced for Loop (for-each loop)

JDK 1.5 introduced a new **for-**loop that enables you to traverse the complete array sequentially without using an index variable.

- For example, the following code displays all elements in the array myList:

```
for (double value: myList)
  System.out.println(value);
```

In general, the syntax is

```
for (elementType var: arrayRefVar) {
  // Process the var
}
```

Note: You must use an index variable if you wish to change the elements in the array.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Enhanced for Loop (for-each loop)

for loop

```
double total = 0.;
for (int i = 0; i <
  myList.length; i++) {
  total += myList[i];
}
```

for-each loop

```
double total=0;
for (double d : myList) {
    total += d;
}
```

The for-each loop is much more general (you can use it with any Java Collection)

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Enhanced for Loop (for-each loop)

- Printing arrays

for loop

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

for-each loop

```
for (double d: myList) {
  System.out.print(d + " ");
}
```

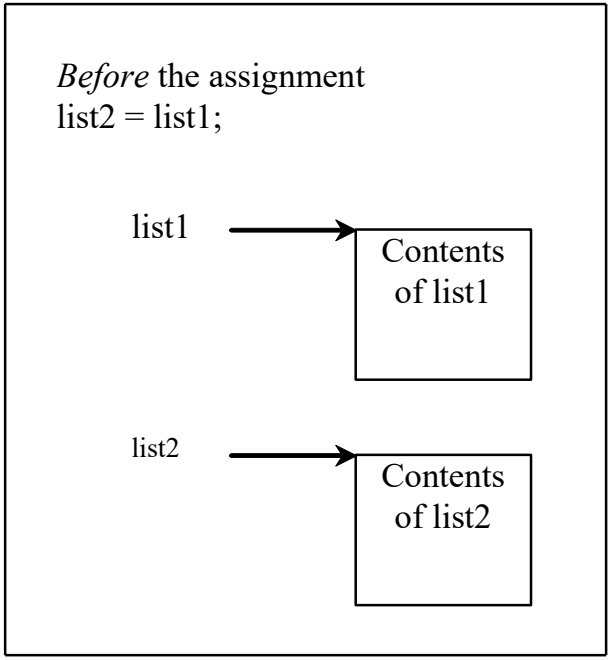

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

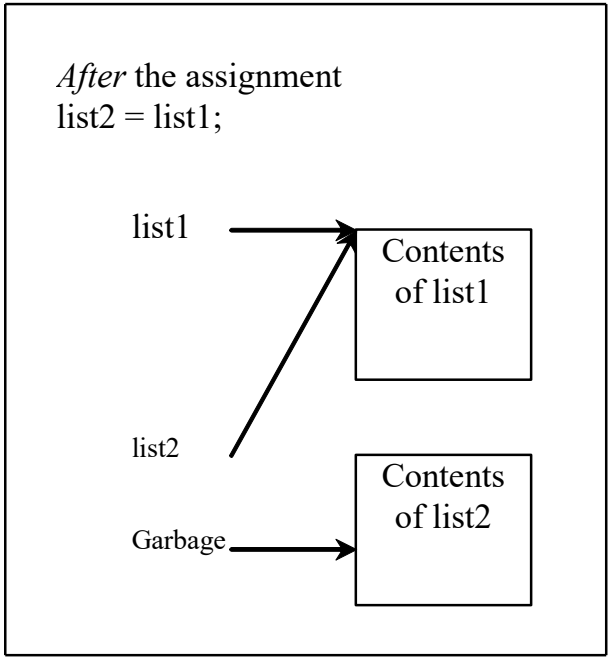
Copying Arrays

- Often, in a program, you need to duplicate an array or a part of an array.
- **Using the assignment statement (=), you re-direct the pointer:**

list2 = list1;

- **You don't copy with "=" !**





(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Copying Arrays

- Using a loop:

```
int[] sourceArray={2, 3, 1, 5, 10};
int[] targetArray=new int[sourceArray.length];
for (int i = 0; i < sourceArray.length; i++)
   targetArray[i] = sourceArray[i];
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

The arraycopy Utility

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

Example:

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

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

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


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

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);
```

OR

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

Anonymous array

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Anonymous Array

The statement

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

creates an array using the following syntax:

```
new dataType[]{literal0, literal1, ..., literalk}
```

There is no explicit reference variable for the array.

Such array is called an ***anonymous array***.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Pass By Value

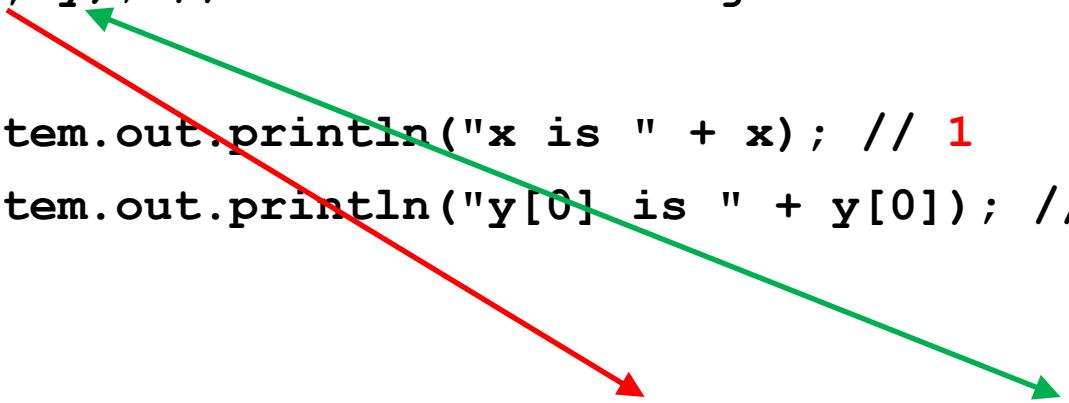
Java uses ***pass by value*** to pass arguments to a method.

- 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.
- Different from a parameter of a primitive type value where 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.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

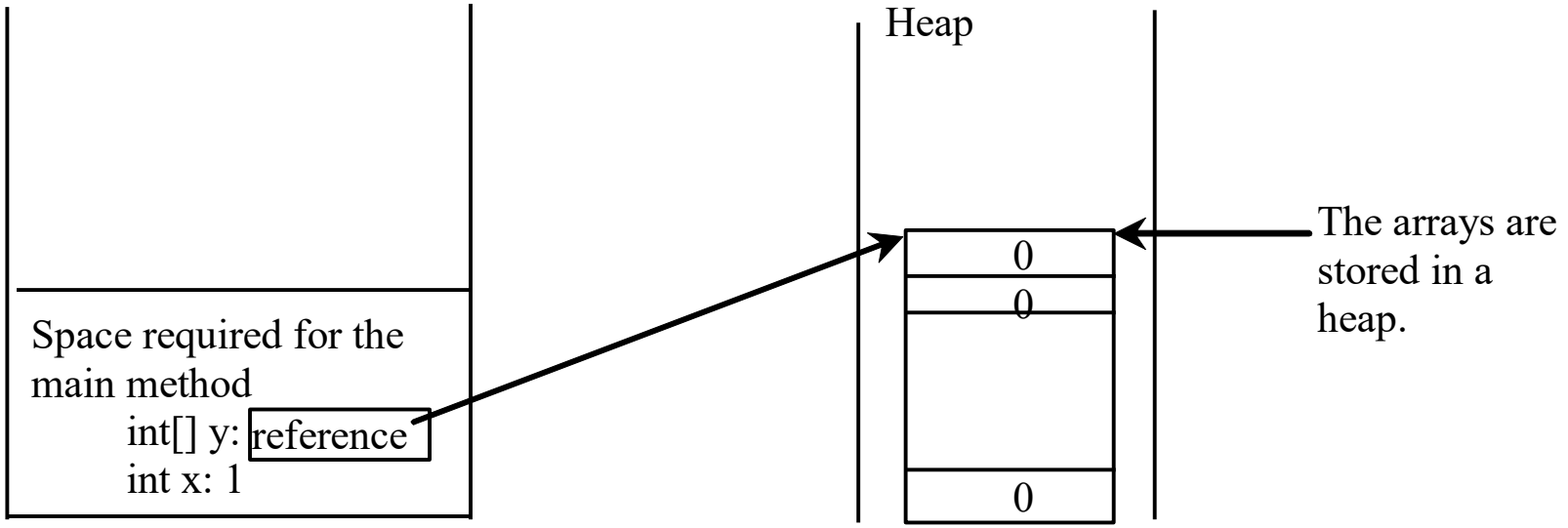
Simple Example

```
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); // 1
    System.out.println("y[0] is " + y[0]); // 5555
  }

  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]
  }
}
```



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

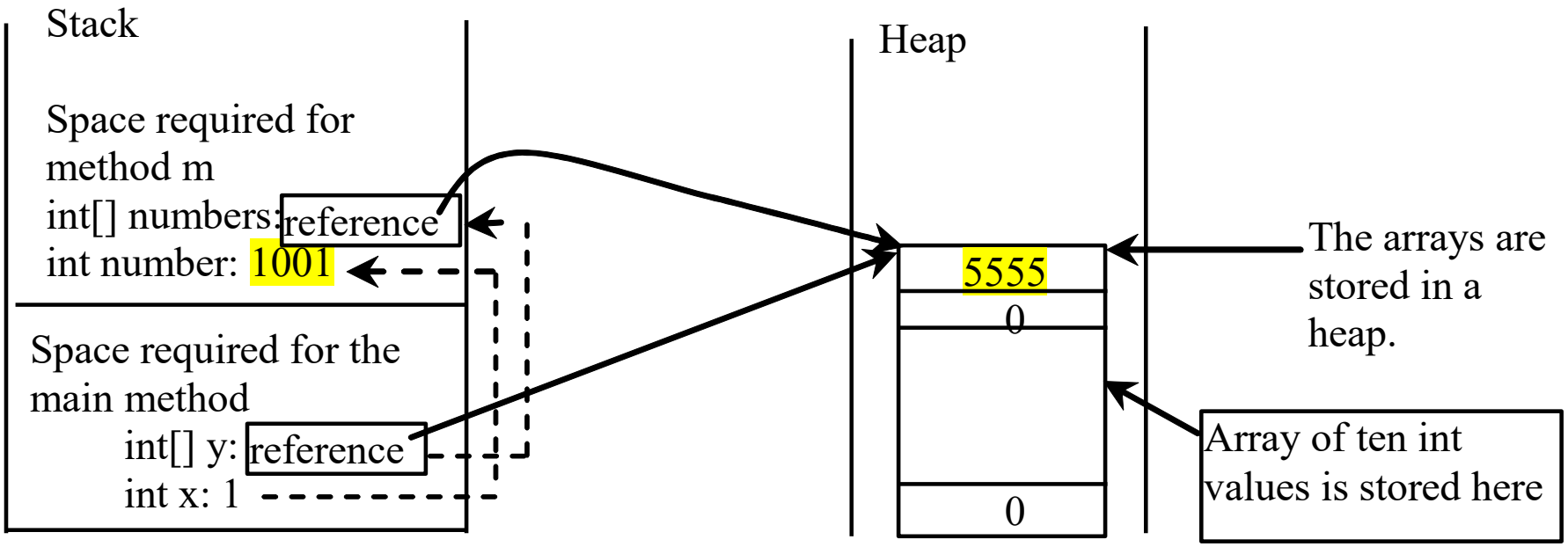
The Heap Segment



The JVM stores arrays and objects in an area of memory, called *heap*, which is used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

The Call Stack Memory

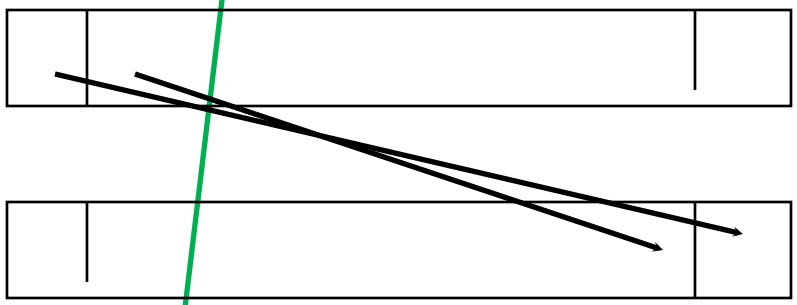


When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Returning an Array from a Method

```
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;
}
```




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

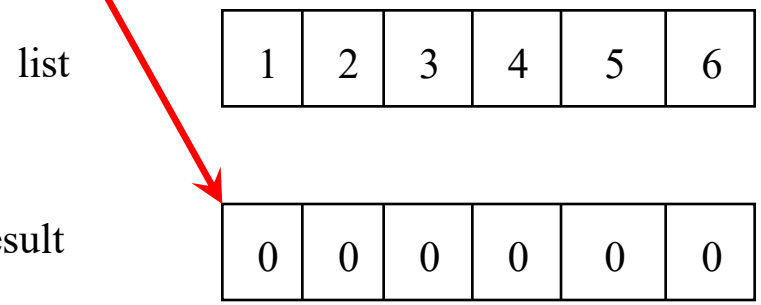

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method

```
int[] list1 = new int[]{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;
}
```



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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

list

1	2	3	4	5	6

result

0	0	0	0	0	0

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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

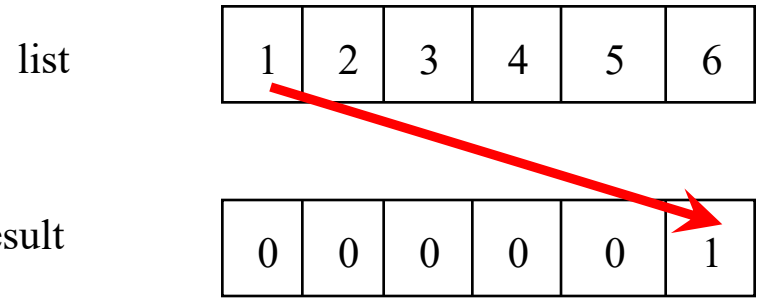
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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]



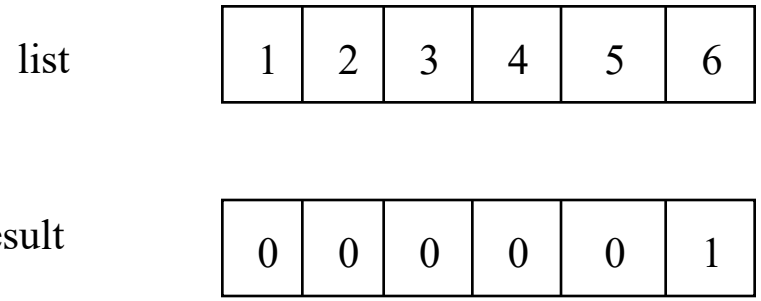
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 1 and j becomes 4



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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 (=1) is less than 6

list	1	2	3	4	5	6

result	0	0	0	0	0	1

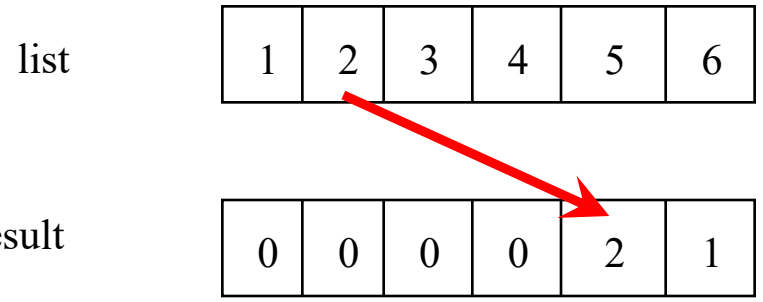
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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 = 1 and j = 4
Assign list[1] to result[4]



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 2 and j becomes 3

list	1	2	3	4	5	6

result	0	0	0	0	2	1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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 (=2) is still less than 6

list	1	2	3	4	5	6

result	0	0	0	0	2	1

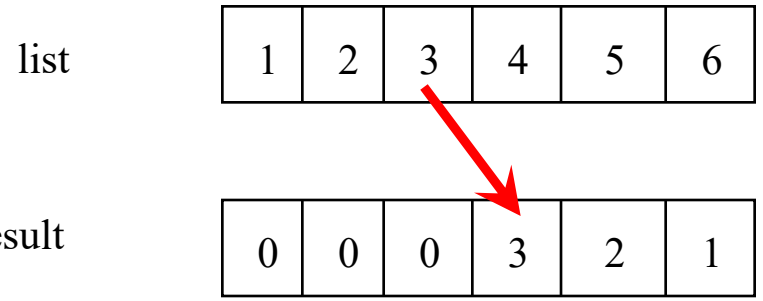
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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 = 2 and j = 3
Assign list[i] to result[j]



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 3 and j becomes 2

list	1	2	3	4	5	6

result	0	0	0	3	2	1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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) is still less than 6

list	1	2	3	4	5	6

result	0	0	0	3	2	1

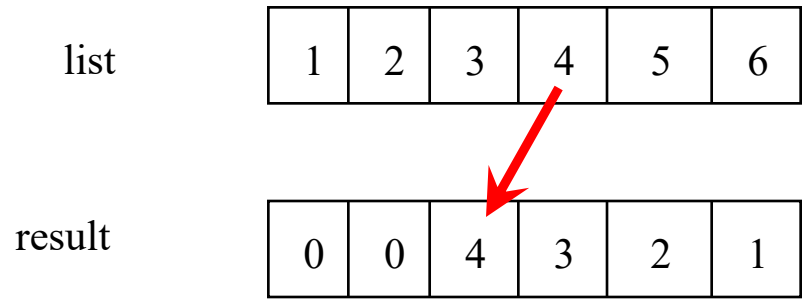
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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]



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 4 and j becomes 1

list	1	2	3	4	5	6

result	0	0	4	3	2	1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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) is still less than 6

list	1	2	3	4	5	6

result	0	0	4	3	2	1

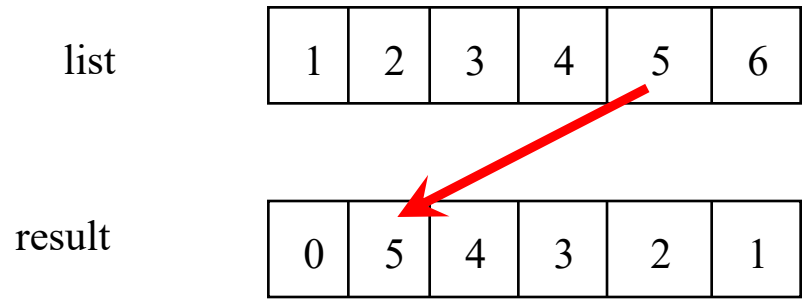
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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]



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 5 and j becomes 0

list	1	2	3	4	5	6

result	0	5	4	3	2	1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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) is still less than 6

list	1	2	3	4	5	6

result	0	5	4	3	2	1

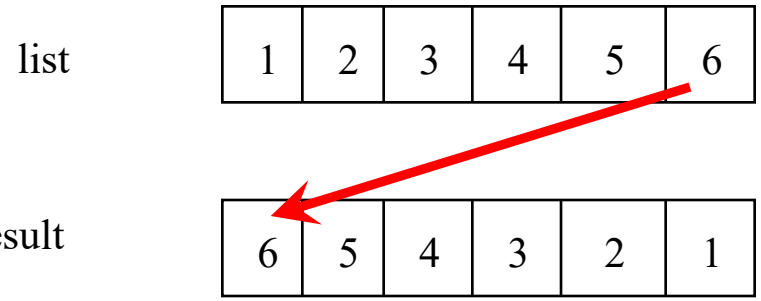
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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]



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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;
}
```

After this, i becomes 6 and j becomes -1

list	1	2	3	4	5	6

result	6	5	4	3	2	1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Trace the reverse Method, cont.

```
int[] list1 = new int[]{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 (=6) < 6 is false. So exit the loop.

list	1	2	3	4	5	6

result	6	5	4	3	2	1

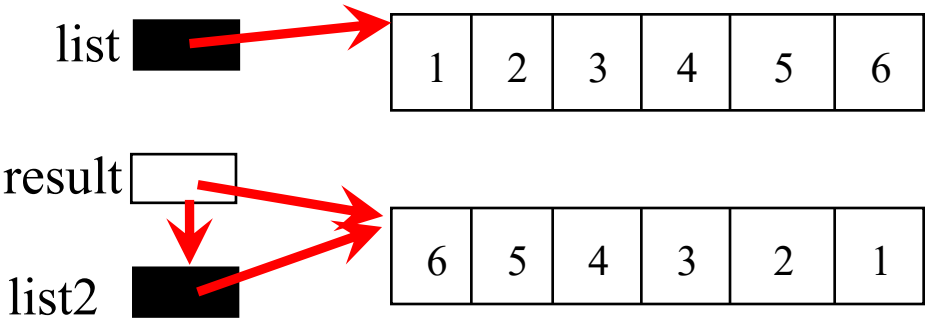
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Returning an Array from a Method

```
int[] list1 = new int[]{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;
}
```

Return result



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Use the same Array in a method

```
public static void reverse(int[] list) {
  int temp;
  for (int i = 0, j = list.length - 1;
       i < list.length/2; i++, j--) {
    temp = list[j];
    list[j] = list[i];
    list[i] = temp;
  }
}
```

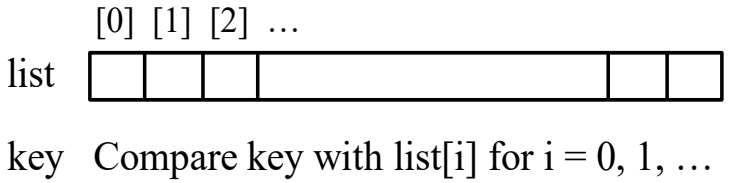
```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
reverse(list1);
System.out.print(list1[0]);
```

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Searching Arrays

- Searching is the process of looking for a specific element in an array

```
public static int linearSearch(int[] list, int key)
```



- return the index of the first occurrence of the key in list
- if not found, return -1

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Linear Search Example

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

From Idea to Solution

```
public static int linearSearch(int[] list, int key) {
  for(int i = 0; i < list.length; i++)
    if (key == list[i])
      return i;
  return -1;
}
```

```
int[] list = {6,4,1,9,7,3,2,8};
int i = linearSearch(list, 3);  // returns 5
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, 4); // returns 1
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Binary Search

- If an array is already ordered, then it is cheaper to find an element
  - Assume that the array is in ascending order. e.g., 1, 2, 3, 4, 6, 7, 8, 9

The binary search first compares the key (e.g., 8) with the element in the middle of the array.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

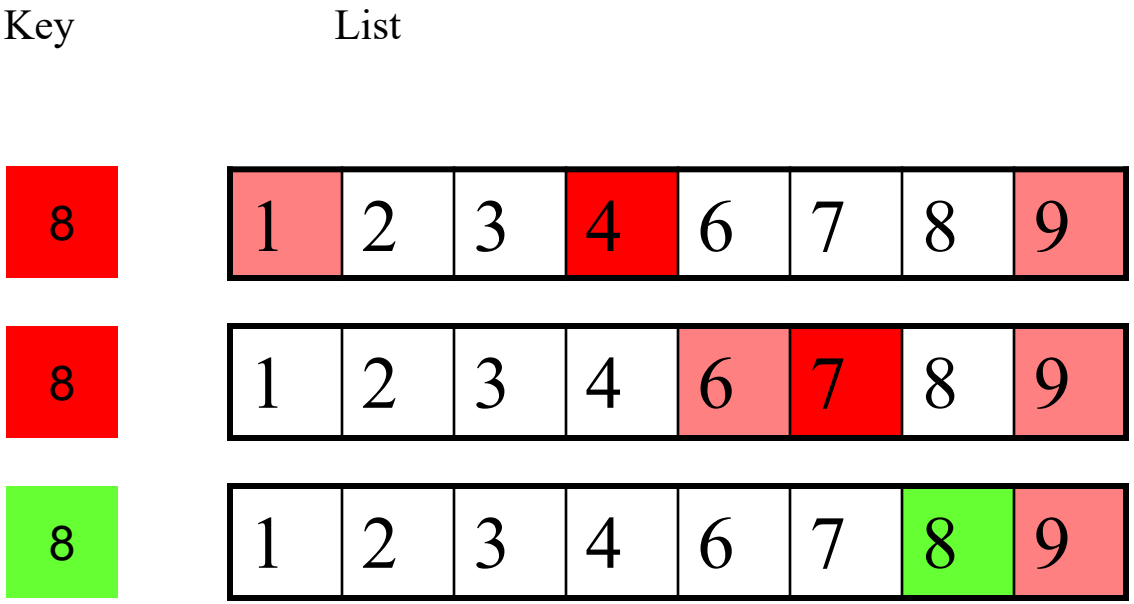
Binary Search

Consider the following three cases:

- If the key is less than the middle element, you only need to search the key in the **first half** of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Binary Search



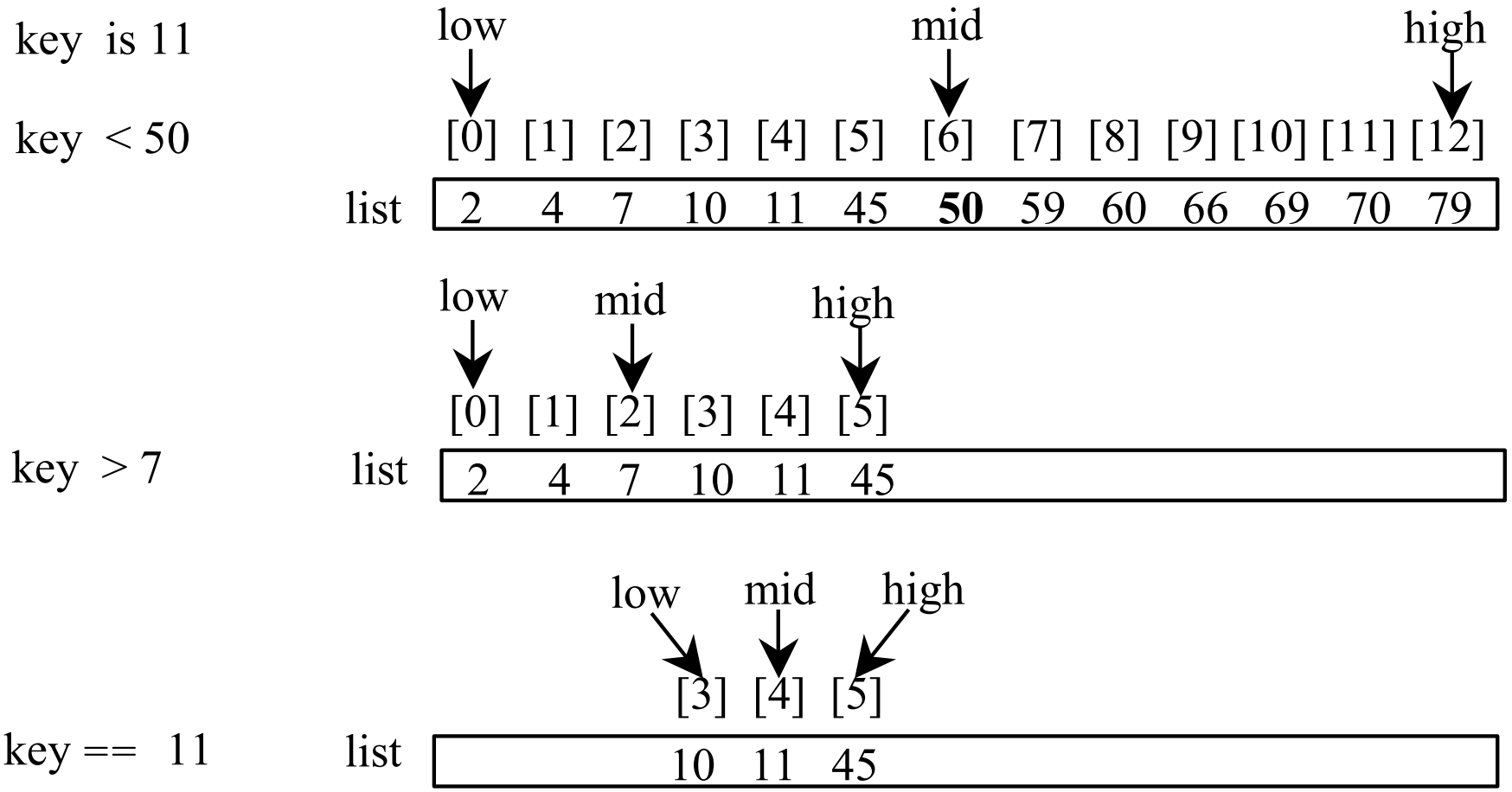
(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

From Idea to Solution

```
/** Use binary search to find the key in the list */
public static int binarySearch(int[] list, int key) {
  int low = 0;
  int high = list.length - 1;
  int mid;
  while(low <= high) {
    mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }
  return -1 - low;
}
```

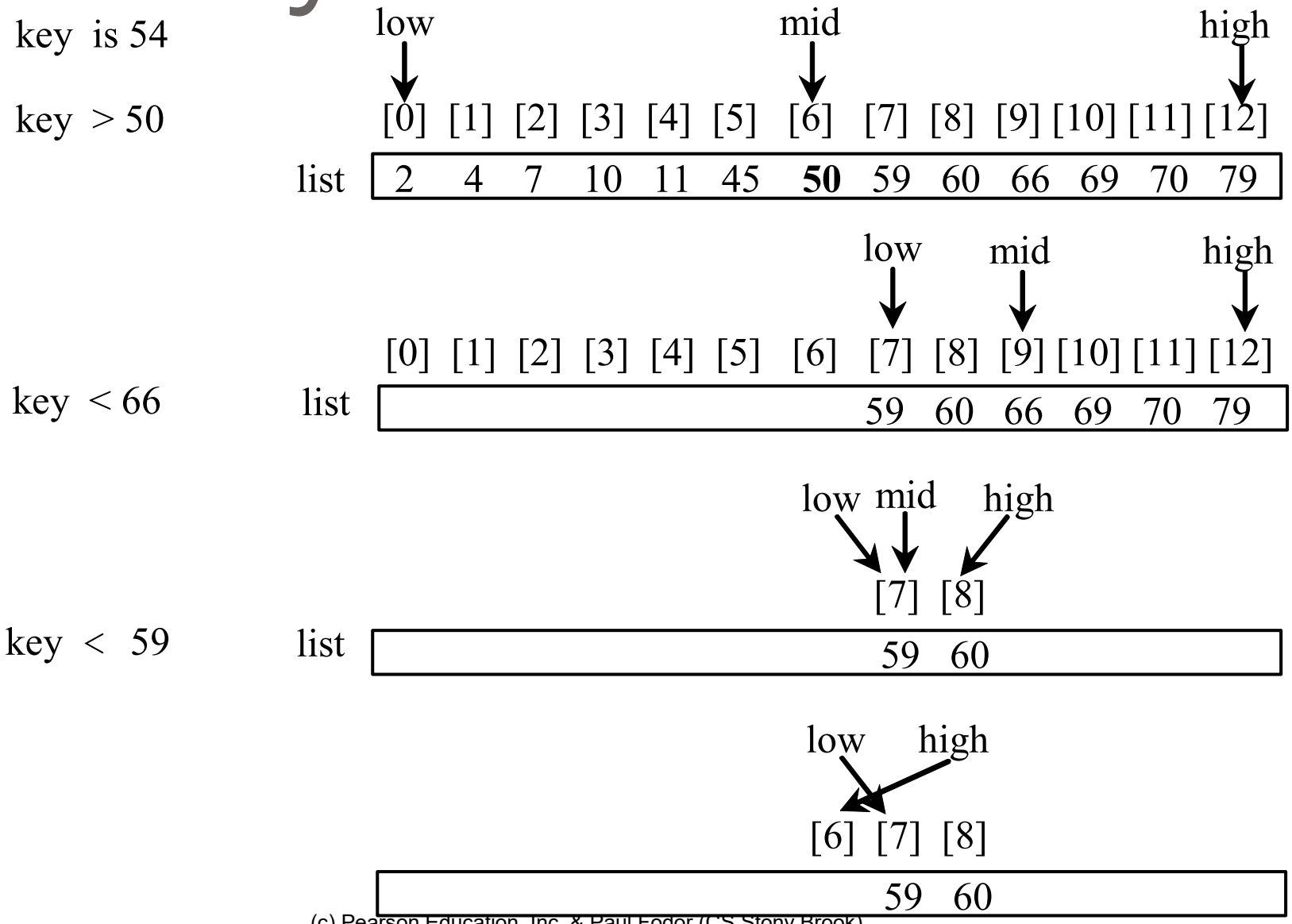

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Binary Search



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Binary Search



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

The Arrays.binarySearch Method

- Java provides several overloaded binarySearch methods for searching a key in an array of int, double, char, short, long, and float in the java.util.Arrays class.

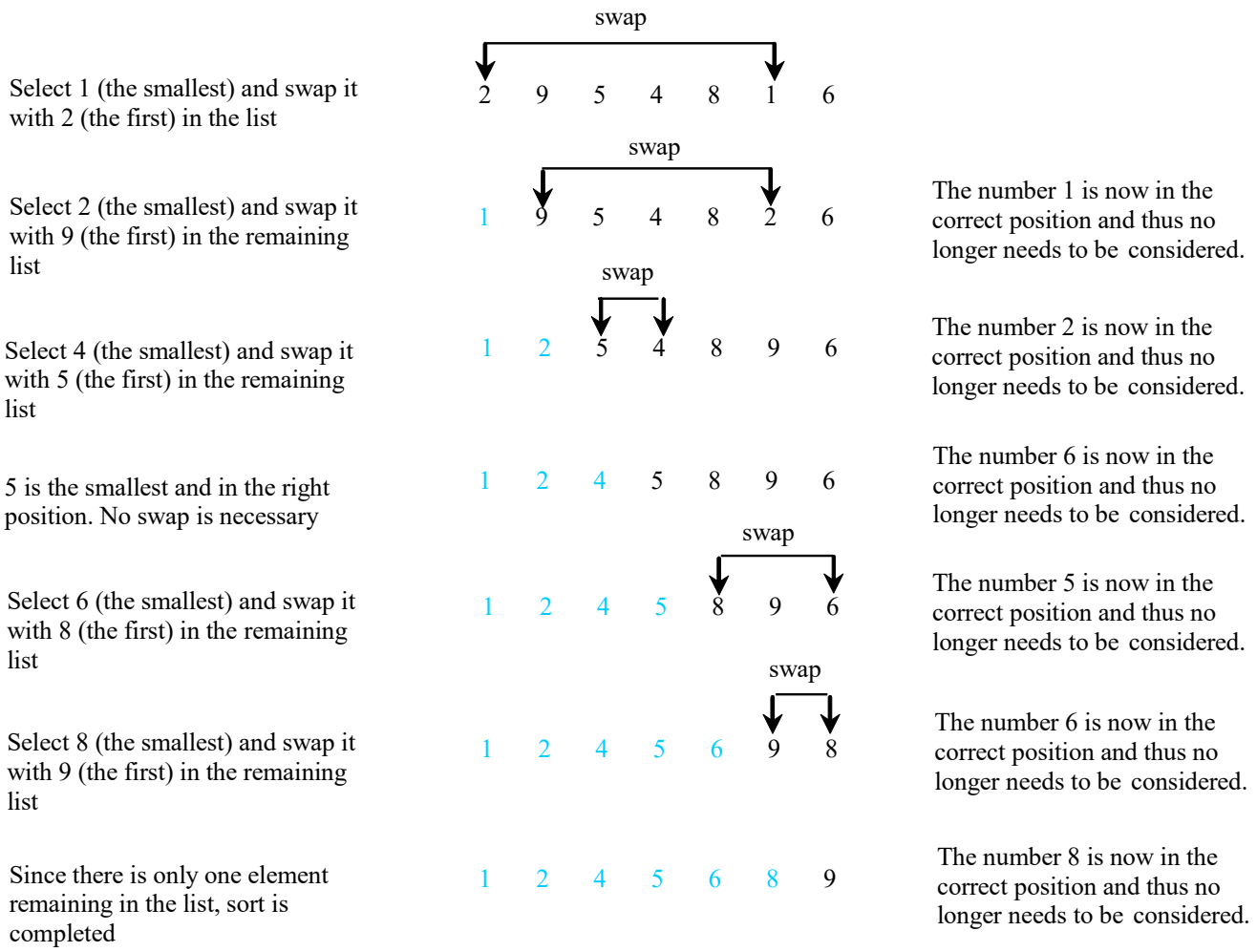
```
int[] list = {1, 2, 3, 4, 6, 7, 8, 9};
System.out.println("Index is " +
  java.util.Arrays.binarySearch(list, 6));
```

Return is 4

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Selection Sort

Selection sort finds the smallest number in the list and places it first. It then finds the smallest number in the remaining list and places it second, and so on until the list contains only a single number. Sort the list {2, 9, 5, 4, 8, 1, 6} using selection sort would be:



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

From Idea to Solution

```
for (int i = 0; i < list.length-1; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i+1..listSize-1]
}
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

From Idea to Solution

```
for (int i = 0; i < list.length-1; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i+1..listSize-1]
}
```

Expand

```
double currentMin = list[i];
int currentMinIndex = i;
for (int j = i+1; j < list.length; j++) {
  if (currentMin > list[j]) {
    currentMin = list[j];
    currentMinIndex = j;
  }
}
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Wrap it in a Method

```
/** A selectionSort method for sorting numbers */
public static void selectionSort(double[] list) {
    for(int i = 0; i < list.length-1; i++) {
      // Find the minimum in the list[i..list.length-1]
      double currentMin = list[i];
      int currentMinIndex = i;
      for (int j = i + 1; j < list.length; j++)
        if (currentMin > list[j]) {
          currentMin = list[j];
          currentMinIndex = j;
        }
      // Swap list[i] with list[currentMinIndex] if necessary;
      if (currentMinIndex != i) {
        list[currentMinIndex] = list[i];
        list[i] = currentMin;
      }
    }
  }
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

Insertion Sort

From index 1 on, the prefix is sorted and we want to insert the current element in its right sorted place (by starting from the position and comparing the elements with temp, for every element greater than temp, shift it right.

```
int[] myList = {2, 9, 5, 4, 8, 1, 6}; // Unsorted
```

Step 1: Initially, the sorted sublist contains the first element in the list. Insert 9 to the sublist.

Step2: The sorted sublist is {2, 9}. Insert 5 to the sublist.

Step 3: The sorted sublist is {2, 5, 9}. Insert 4 to the sublist.

Step 4: The sorted sublist is {2, 4, 5, 9}. Insert 8 to the sublist.

Step 5: The sorted sublist is {2, 4, 5, 8, 9}. Insert 1 to the sublist.

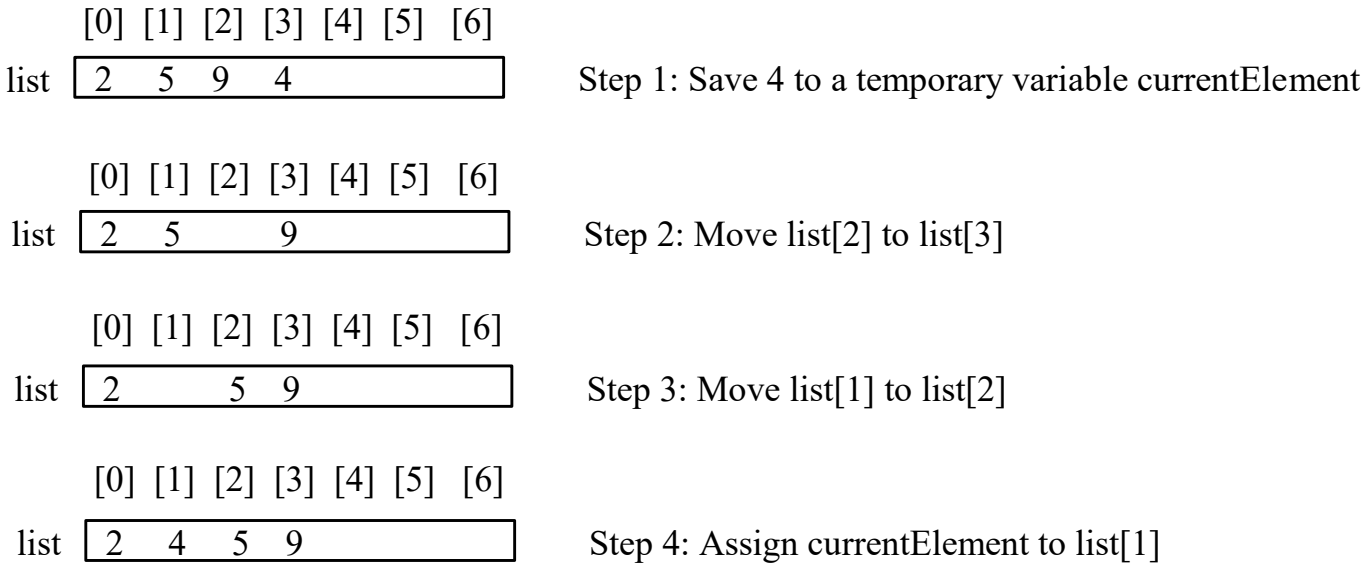
Step 6: The sorted sublist is {1, 2, 4, 5, 8, 9}. Insert 6 to the sublist.

Step 7: The entire list is now sorted

(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

How to Insert?

One iteration:



(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

From Idea to Solution

```
public static void insertionSort(double[] a){
  for(int i=1; i < a.length; i++){
    double temp = a[i];
    int j;
    for(j = i-1; j >= 0; j--)
      if(temp < a[j])
        a[j+1] = a[j];
      else
        break;
    a[j+1] = temp;
  }
}
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

```
public class Test {
  public static void insertionSort(double[] a) {
    for (int i = 1; i < a.length; i++) {
      double temp = a[i];
      int j;
      for (j = i - 1; j >= 0; j--)
        if (temp < a[j])
          a[j + 1] = a[j];
        else
          break;
      a[j + 1] = temp;
    }
  }
  public static void main(String[] args) {
    double[] a = { 5.0, 4.0, 3.0, 2.0, 1.0 };
    insertionSort(a);
    for (double v : a)
      System.out.print(v + " ");
  }
}
1.0 2.0 3.0 4.0 5.0
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)

The Arrays.sort Method

- Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the **`java.util.Arrays`** class.
  - For example, the following code sorts an array of numbers and an array of characters:

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);

char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```


(c) Pearson Education, Inc. & Paul Fodor (CS Stony Brook)