Chapter 7
Single & Multi-Dimensional Arrays

COSC1046

Ping Luo

Algoma UNIVERSITY

Introduction

- Often, we will have to store a large number of values during the execution of a program.
- Using a large number of variables like number0, number1, ..., and number99 would overly complicate the program.
- Array is data structure which stores a fixed-size sequential collection of elements of the same type.

Array Basics

- An array is used to store a collection of data, but often we find it more useful to think of an array as a collection of variables of the same type.
- Instead of declaring individual variables, we declare one array variable such as numbers and use numbers[0], numbers[1], . . . , and numbers[99] to represent individual variables.

Array Basics

```
elementType[] arrayRefVar;
or: elementType arrayRefVar[]; // Allowed, but not preferred

Example:
1.    double[] myList;
      myList = new double[10];
2.    double[] myList = new double[10];
```

Array Basics

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.

myList.length; //no () here

Array Basics

myList[0] = 5.6;

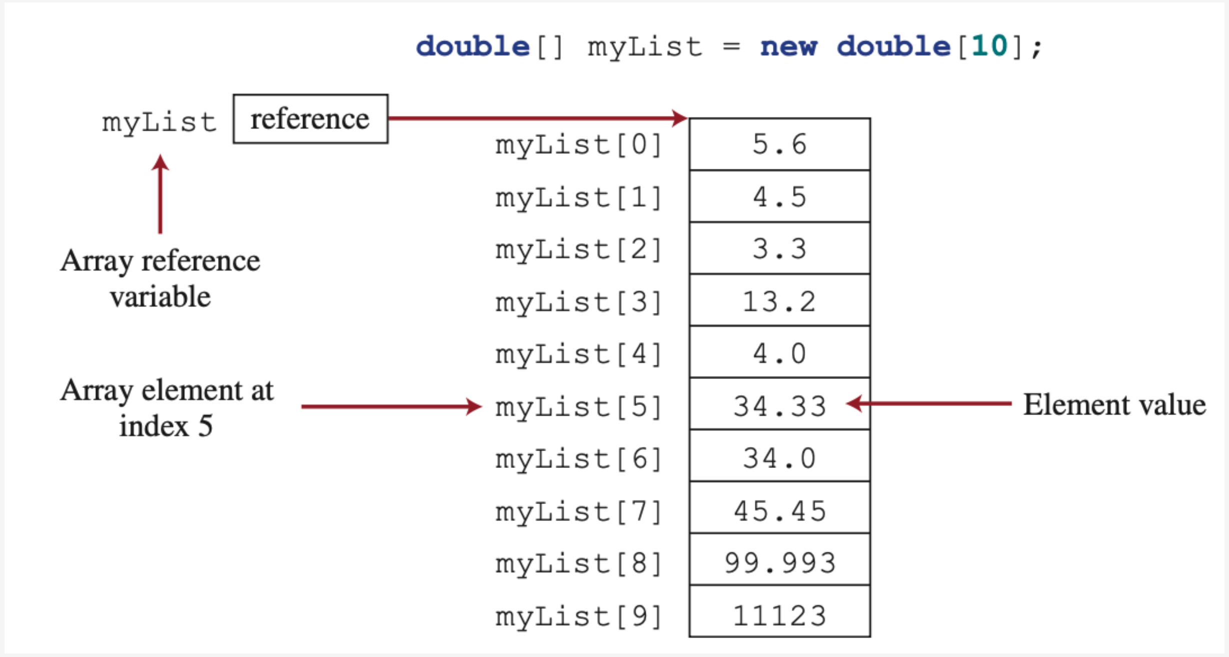
myList[1] = 4.5;

myList[2] = 3.3;

…

myList[8] = 99.993;

myList[9] = 11123;

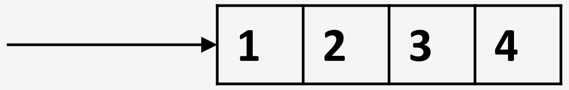


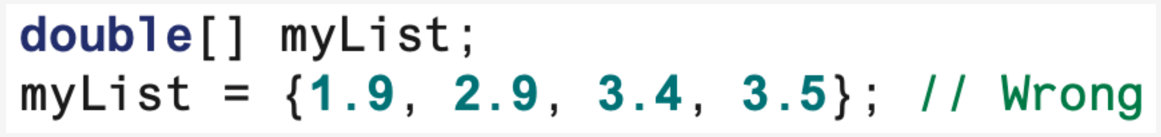
```
double[] myList = new double[10];
```

Array Basics

elementType[] arrayRefVar = {value0, value1, ..., valuek};

double[] myList = {1, 2, 3, 4};





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

Array and Loop

- All of the elements in an array are of the same type. They can be evenly processed in the same fashion repeatedly using a loop.
- Since the size of the array is known, it is natural to use a for loop.

Array and Loop

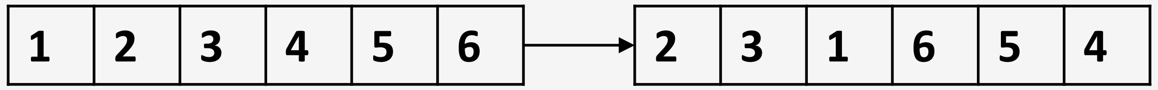
- All of the elements in an array are of the same type. They are evenly processed in the same fashion repeatedly using a loop.
- Since the size of the array is known, it is natural to use a for loop.

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

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

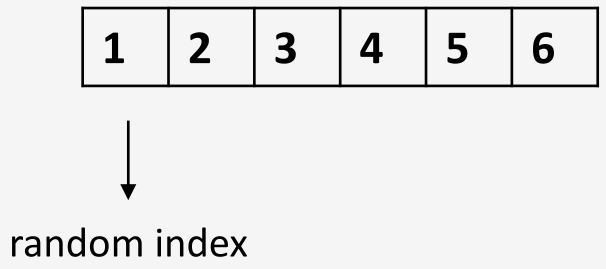
Case Study: Random shuffling

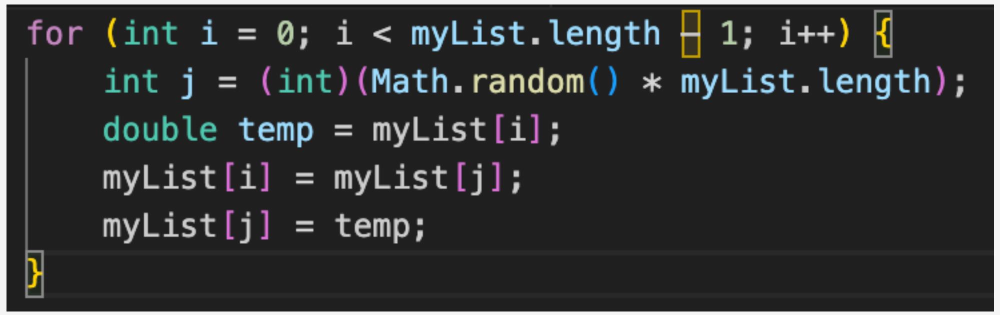
- In many applications, you need to randomly reorder the elements in an array. This is called shuffling.
- Shuffling is widely used in AI.



Case Study: Random shuffling

- In many applications, you need to randomly reorder the elements in an array. This is called shuffling.
- Shuffling is widely used in AI.

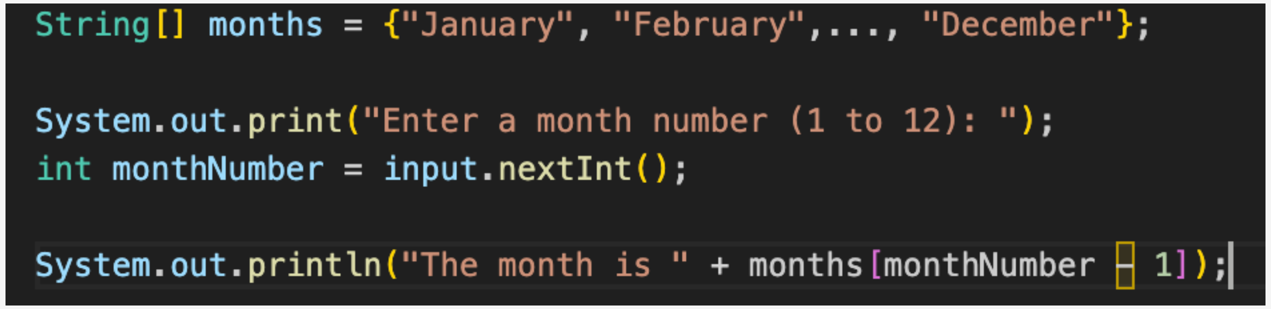




```java
for (int i = 0; i < myList.length - 1; i++) {
    int j = (int)(Math.random() * myList.length);
    double temp = myList[i];
    myList[i] = myList[j];
    myList[j] = temp;
}
```

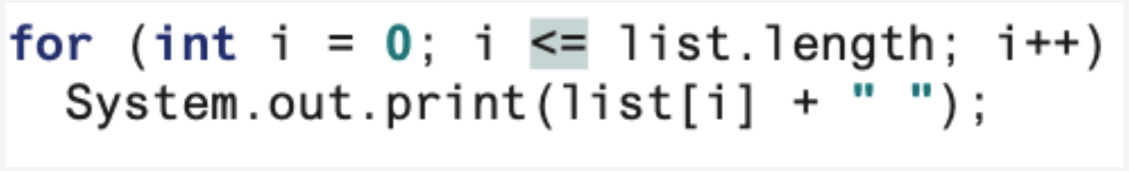
Array - Simplifying Coding

```
String monthName;
switch (monthNumber) {
   case 1:  monthName = "January"; break;
   case 2:  monthName = "February"; break;
   case 3:  monthName = "March"; break;
   case 4:  monthName = "April"; break;
   case 5:  monthName = "May"; break;
   case 6:  monthName = "June"; break;
   case 7:  monthName = "July"; break;
   case 8:  monthName = "August"; break;
   case 9:  monthName = "September"; break;
   case 10: monthName = "October"; break;
   case 11: monthName = "November"; break;
   case 12: monthName = "December"; break;
   default: monthName = "Invalid month"; break;
}
```



```
String[] months = {"January", "February",..., "December"};

System.out.print("Enter a month number (1 to 12): ");
int monthNumber = input.nextInt();

System.out.println("The month is " + months[monthNumber - 1]);
```

Caution



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

ArrayIndexOutOfBoundsException

Foreach Loop

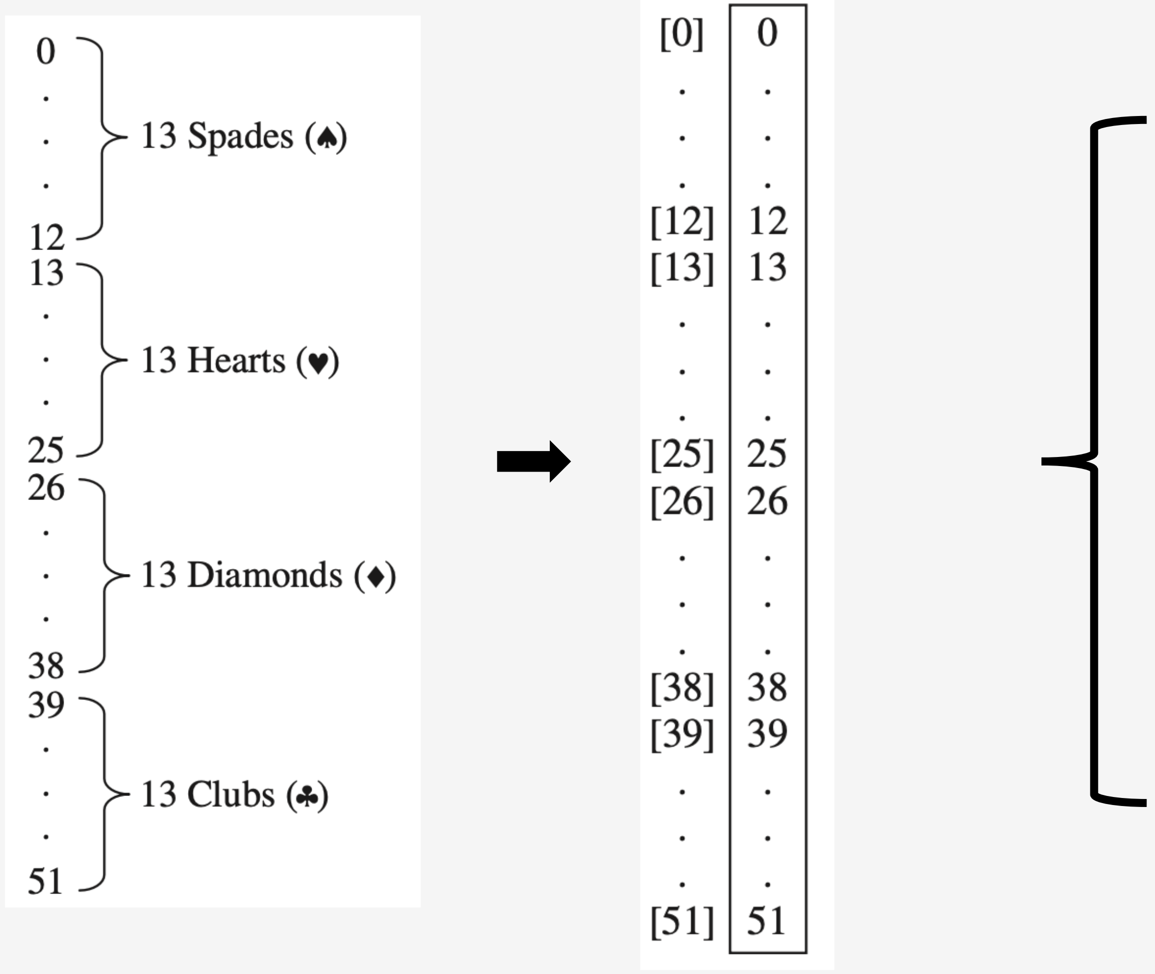
- Java supports a convenient for loop, known as a foreach loop, which enables us to traverse the array sequentially without using an index variable.

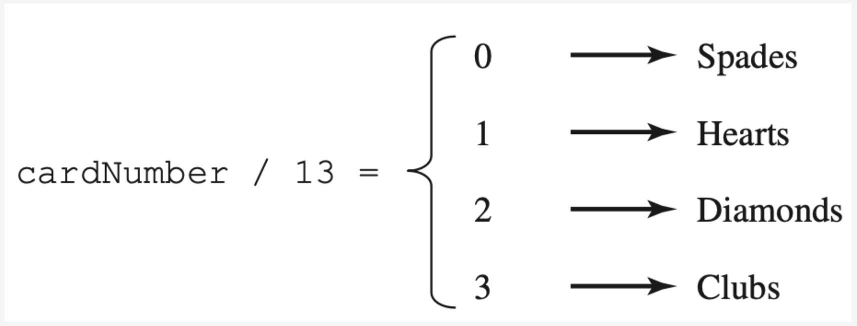
```
for (double e: myList) {
    System.out.println(e);
}
```

Case Study: Deck of Cards

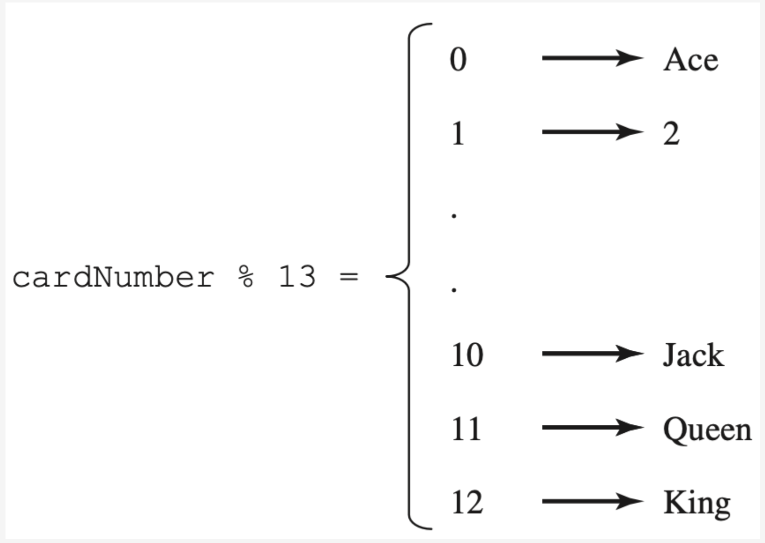
- Create a program that will randomly shuffle a deck of cards, and then display the first 4 cards.
  1. create an array to store the 52 cards;
  2. shuffle the cards (array);
  3. display the first 4 cards;

Case Study: Deck of Cards





$$\texttt{cardNumber / 13} = \begin{cases} 0 \longrightarrow \text{Spades} \\ 1 \longrightarrow \text{Hearts} \\ 2 \longrightarrow \text{Diamonds} \\ 3 \longrightarrow \text{Clubs} \end{cases}$$



$$\texttt{cardNumber \% 13} = \begin{cases} 0 \longrightarrow \text{Ace} \\ 1 \longrightarrow 2 \\ . \\ . \\ 10 \longrightarrow \text{Jack} \\ 11 \longrightarrow \text{Queen} \\ 12 \longrightarrow \text{King} \end{cases}$$

Case Study: Deck of Cards

```java
public class DeckOfCards {
  public static void main(String[] args) {
    int[] deck = new int[52];
    String[] suits = {"Spades", "Hearts", "Diamonds", "Clubs"};
    String[] ranks = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9",
                      "10", "Jack", "Queen", "King"};
    // Initialize cards (1 point)

    // Shuffle the cards (2 points)

    // Display the first four cards (2 points)
    for (int i = 0; i < 4; i++) {
      _____________
      _____________
      System.out.println("Card number " + deck[i] + ": "
      + rank + " of " + suit);
    }
  }
}
```

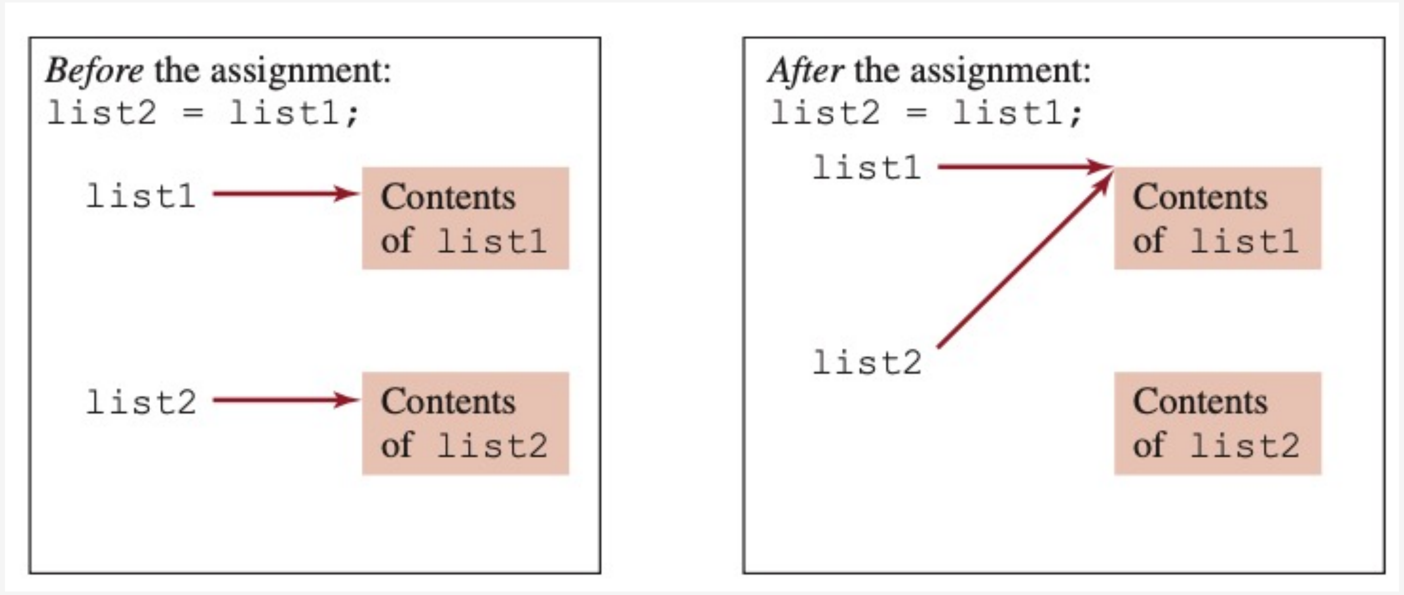
Copying Arrays

It is common to copying arrays and compare the original array with manipulated arrays.

Copying Arrays

list2 = list1;

Wrong!



Copying Arrays

1. Use a loop to copy individual elements one by one.
2. Use the static arraycopy method in the System class.
3. Use the clone method to copy arrays;

Copying Arrays

1. **Use a loop to copy individual elements one by one.**
2. Use the static arraycopy method in the System class.
3. Use the clone method to copy arrays;

Copying Arrays

1. Use a loop to copy individual elements one by one.
2. **Use the static arraycopy method in the System class.**
3. Use the clone method to copy arrays;

Copying Arrays

- arraycopy(**sourceArray**, srcPos, **targetArray**, tarPos, length);
- The parameters srcPos and tarPos indicate the starting positions in sourceArray and targetArray, respectively.

```
System.arraycopy(list1, 0, list2, 0, list1.length);
```

Copying Arrays

- arraycopy(sourceArray, srcPos, targetArray, tarPos, length);
- The parameters srcPos and tarPos indicate the starting positions in sourceArray and targetArray, respectively.

```
System.arraycopy(list1, 0, list2, 0, list1.length);
```

- The arraycopy method does not allocate memory space for the target array. The target array must have already been created with its memory space allocated. After the copying takes place, targetArray and sourceArray have the same content but independent memory locations.

Copying Arrays

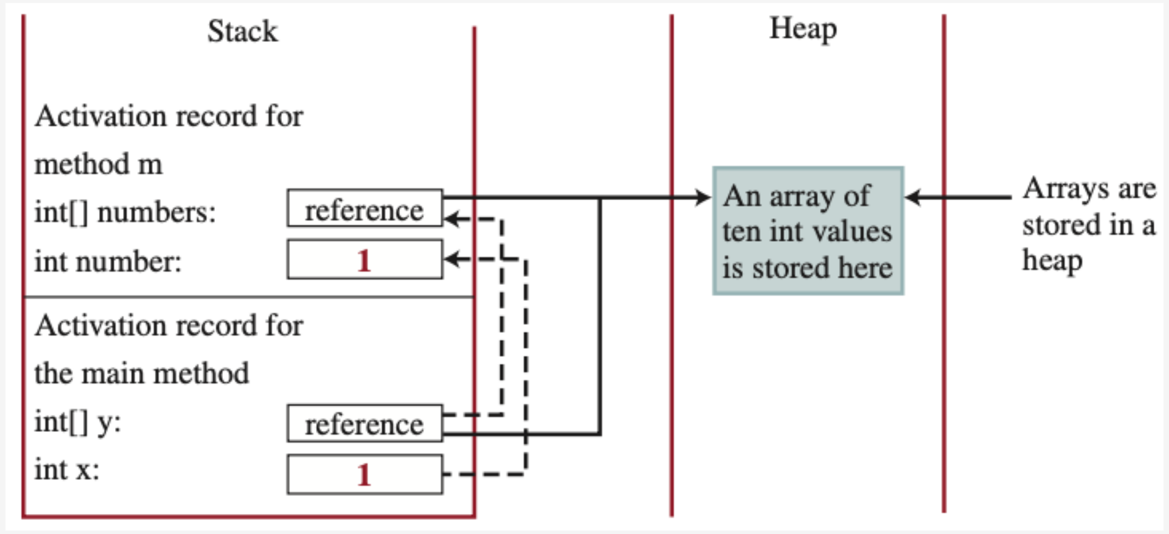
1. Use a loop to copy individual elements one by one.
2. Use the static arraycopy method in the System class.
3. **Use the clone method to copy arrays; this will be introduced in Chapter 13, Abstract Classes and Interfaces.**

Copying Arrays

```
int[] array = {23, 43, 55, 12};

int[] copiedArray = array.clone();
```

Passing Arrays to Methods

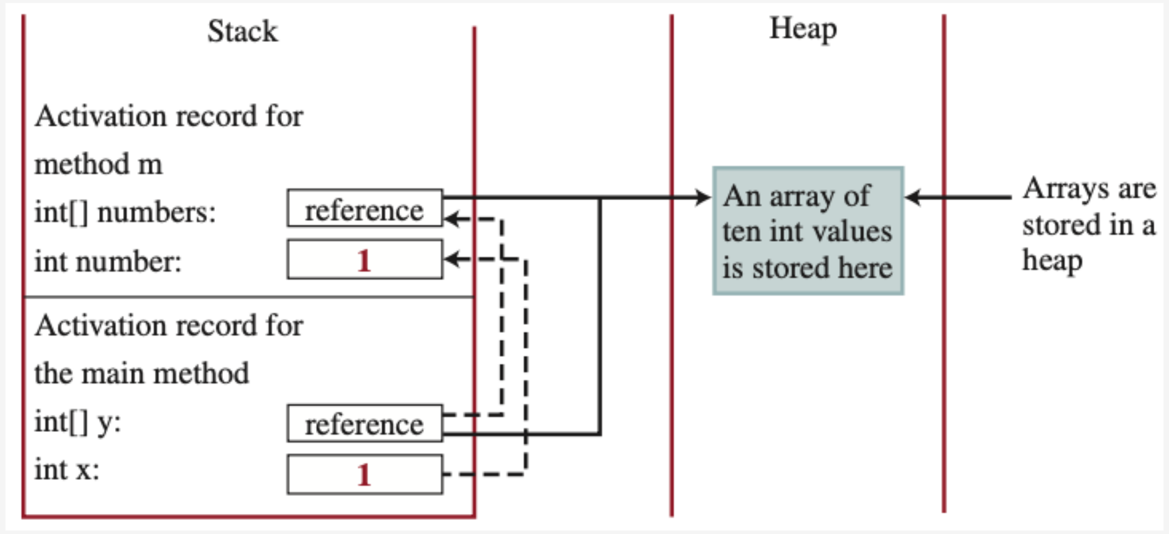
Java uses pass-by-value to pass arguments to a method, how about array? NO!


Stack
Heap
Activation record for method m
int[] numbers:
reference
int number:
1
Activation record for the main method
int[] y:
reference
int x:
1
An array of ten int values is stored here
Arrays are stored in a heap

Passing Arrays to Methods

Java uses pass-by-value to pass arguments to a method, how about array?

NO!

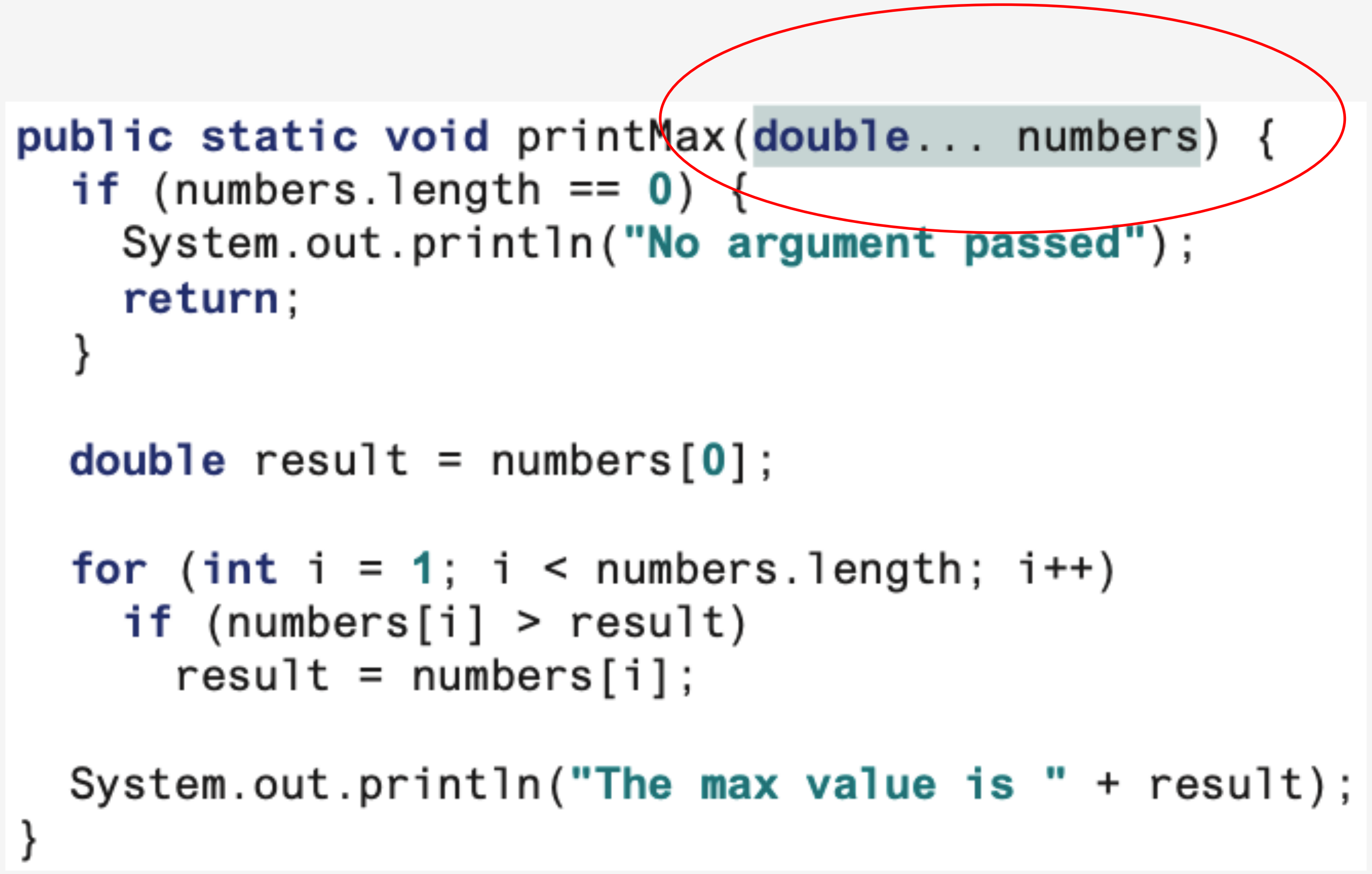

Stack
Heap
Activation record for method m
int[] numbers:
reference
int number:
1
Activation record for the main method
int[] y:
reference
int x:
1
An array of ten int values is stored here
Arrays are stored in a heap

If an array is changed in the method, you will see the change outside the method.

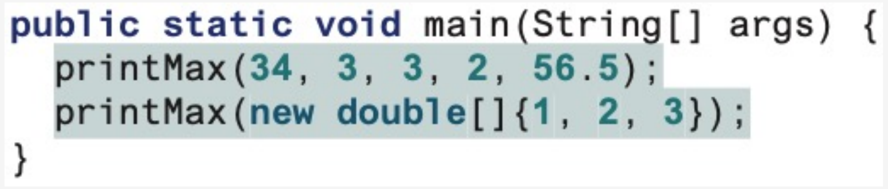
Variable-Length Argument Lists

- A variable number of arguments of the same type can be passed to a method and treated as an array.
- The parameter in the method is declared as follows:
  typeName... parameterName

Variable-Length Argument Lists



```java
public static void printMax(double... numbers) {
  if (numbers.length == 0) {
    System.out.println("No argument passed");
    return;
  }

  double result = numbers[0];

  for (int i = 1; i < numbers.length; i++)
    if (numbers[i] > result)
      result = numbers[i];

  System.out.println("The max value is " + result);
}
```



```java
public static void main(String[] args) {
  printMax(34, 3, 3, 2, 56.5);
  printMax(new double[]{1, 2, 3});
}
```

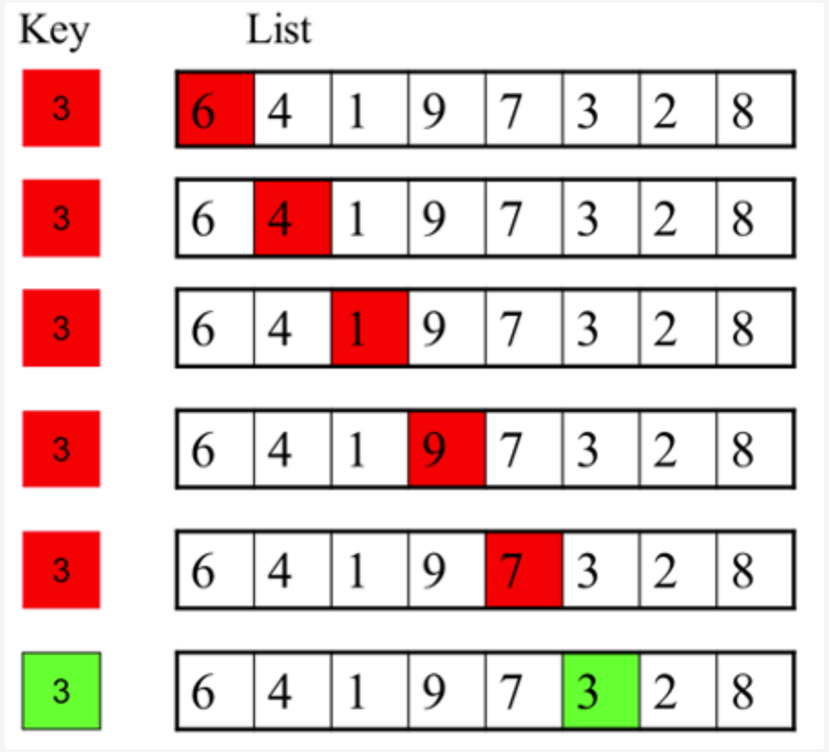
Array Algorithms

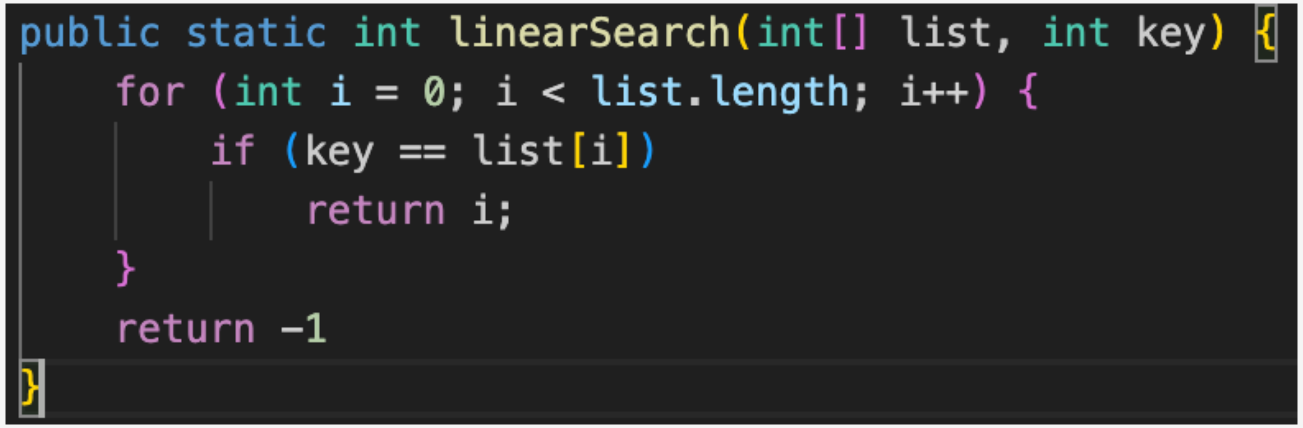
- Searching Array
- Sorting Array

Searching Arrays

- Searching is the process of looking for a specific element in an array—for example, discovering whether a certain score is included in a list of scores.
- Searching is a **common task** in computer programming. Many algorithms and data structures are devoted to searching.
- Linear search and Binary search.

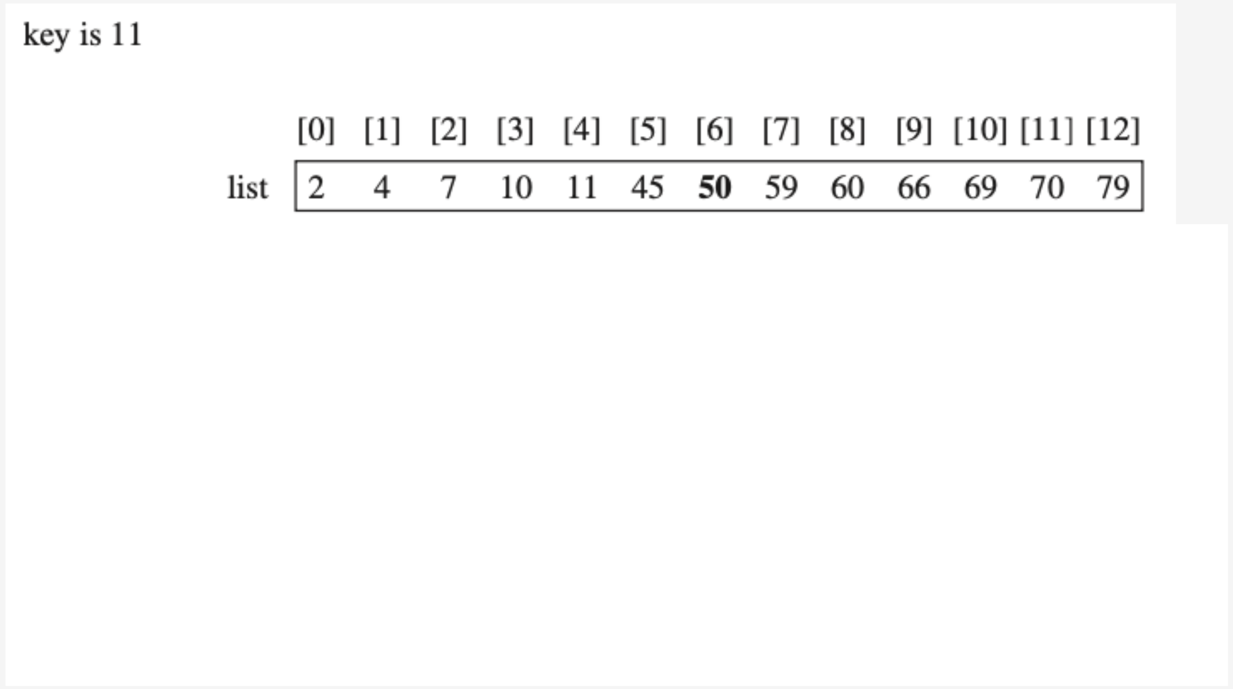
Linear Search




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

Binary Search

If an array is sorted, binary search is more efficient than linear search for finding an element in the array.

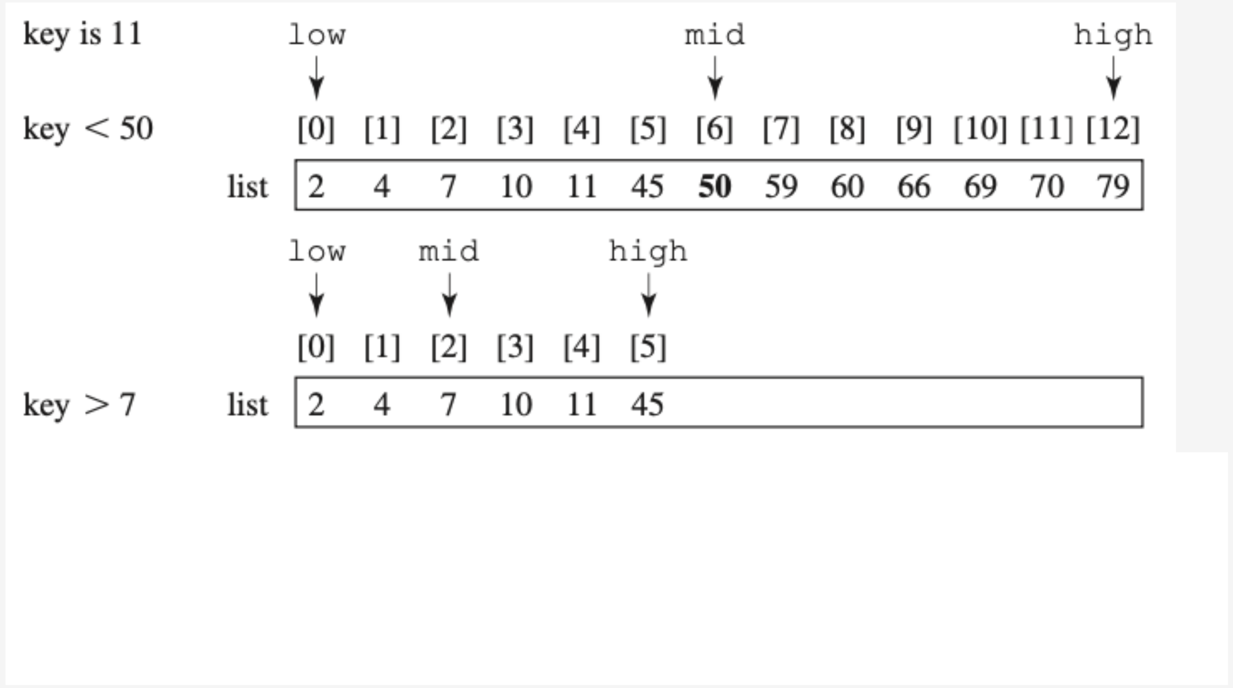


key is 11

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
list	2	4	7	10	11	45	**50**	59	60	66	69	70	79

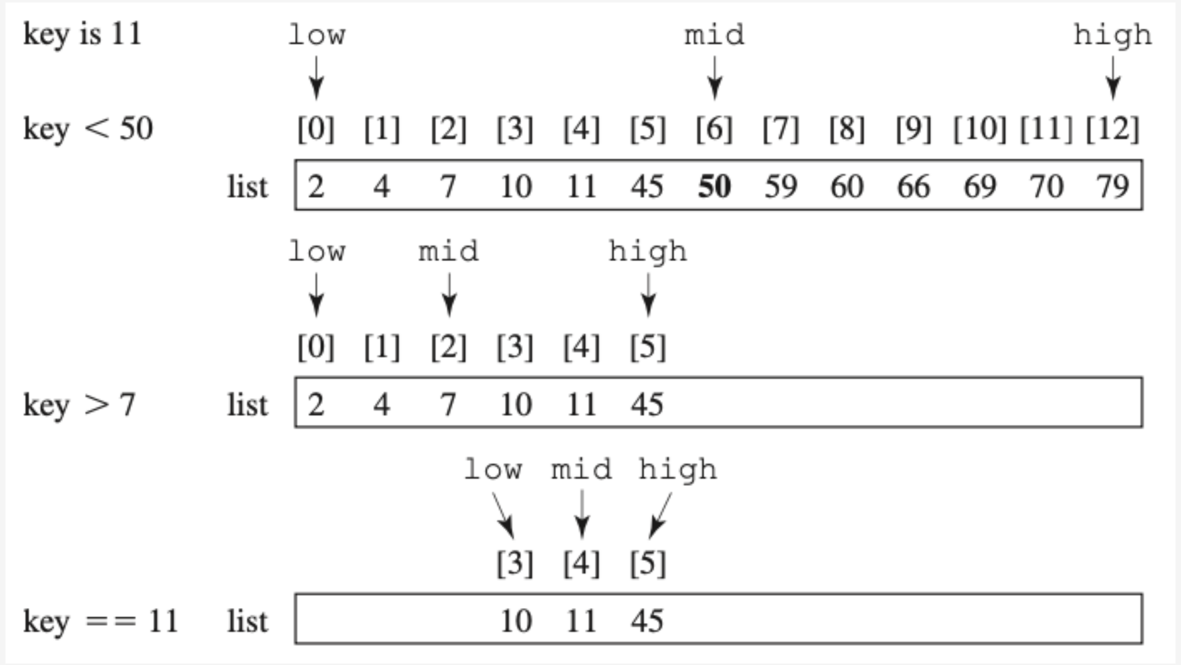
Binary Search

If an array is sorted, binary search is more efficient than linear search for finding an element in the array.

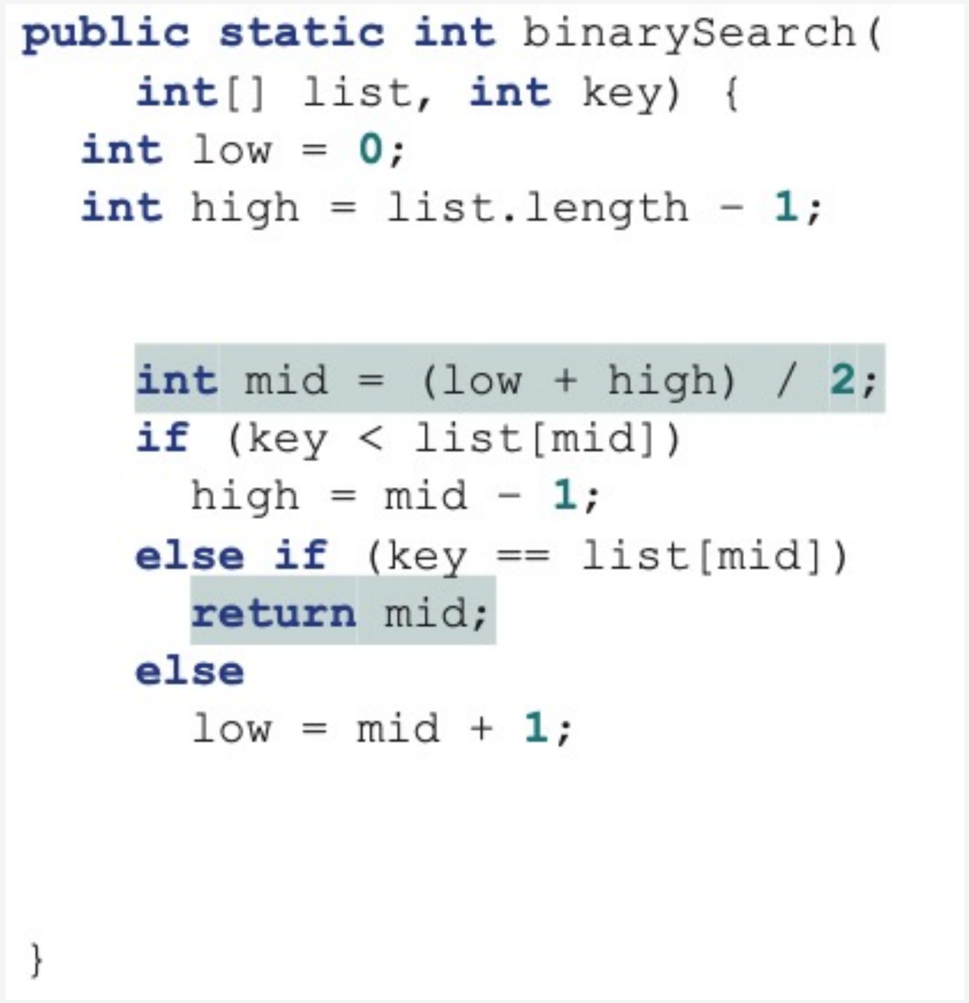


Binary Search

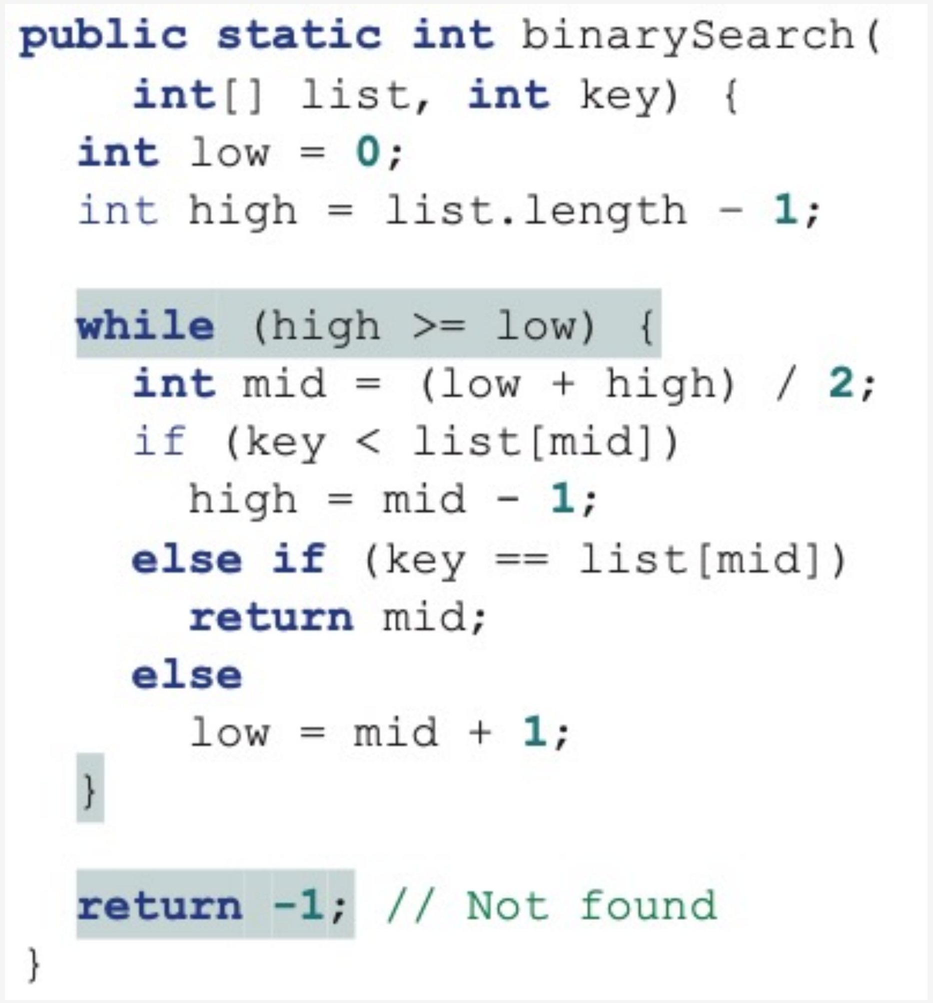
If an array is sorted, binary search is more efficient than linear search for finding an element in the array.



Binary Search



```java
public static int binarySearch(
    int[] list, int key) {
  int low = 0;
  int high = list.length - 1;


    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;



}
```



```java
public static int binarySearch(
    int[] list, int key) {
  int low = 0;
  int high = list.length - 1;

  while (high >= low) {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }

  return -1; // Not found
}
```

Sorting Arrays

Sorting, like searching, is a common task in computer programming.

Many different algorithms have been developed for sorting.

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Merge Sort
5. Quick Sort
6. Heap Sort
7. Shell Sort
8. Radix Sort
9. Counting Sort
10. Bucket Sort

Sorting Arrays

Sorting, like searching, is a common task in computer programming.

Many different algorithms have been developed for sorting.

1. **Bubble Sort**
2. **Selection Sort**
3. Insertion Sort
4. Merge Sort
5. Quick Sort
6. Heap Sort
7. Shell Sort
8. Radix Sort
9. Counting Sort
10. Bucket Sort

Bonus!

Selection Sort

https://liveexample.pearsoncmg.com/dsanimation/SelectionSortNew.html

Selection Sort

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

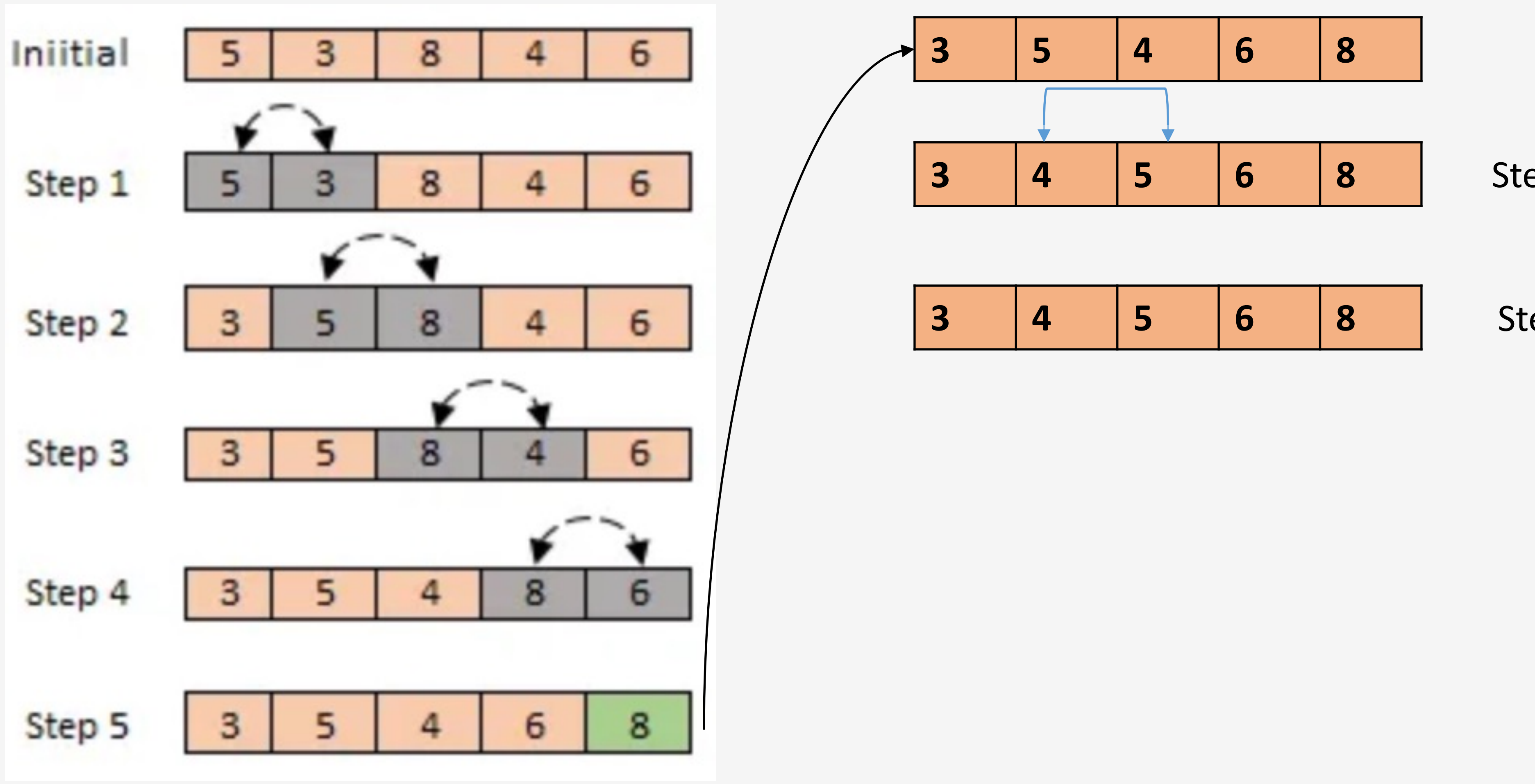
1. select smallest element

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

2. swap the smallest with list[i]

```
if (currentMinIndex != i) {
  list[currentMinIndex] = list[i];
  list[i] = currentMin;
}
```

Bubble Sort


Iniitial
5 3 8 4 6
Step 1
5 3 8 4 6
Step 2
3 5 8 4 6
Step 3
3 5 8 4 6
Step 4
3 5 4 8 6
Step 5
3 5 4 6 8
3 5 4 6 8
3 4 5 6 8
Step6
3 4 5 6 8
Step7

Arrays.sort

- The java.util.Arrays class contains various static methods for sorting and searching arrays.
- You can use the sort or parallelSort method to sort a whole array or a partial array.

Arrays.sort

- The java.util.Arrays class contains various static methods for sorting and searching arrays.
- You can use the sort or parallelSort method to sort a whole array or a partial array.

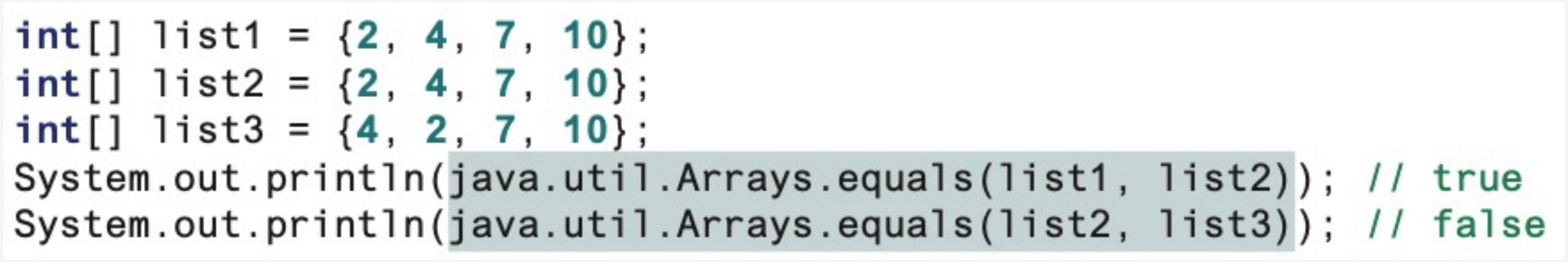
```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);
java.util.Arrays.parallelSort(numbers);
```

Arrays.binarySearch

int[] **list1** = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};

java.util.Arrays.binarySearch(**list1**, 11)

Arrays.equals



```
int[] list1 = {2, 4, 7, 10};
int[] list2 = {2, 4, 7, 10};
int[] list3 = {4, 2, 7, 10};
System.out.println(java.util.Arrays.equals(list1, list2)); // true
System.out.println(java.util.Arrays.equals(list2, list3)); // false
```

Final Exam Schedule

Date: April 13th, 2024

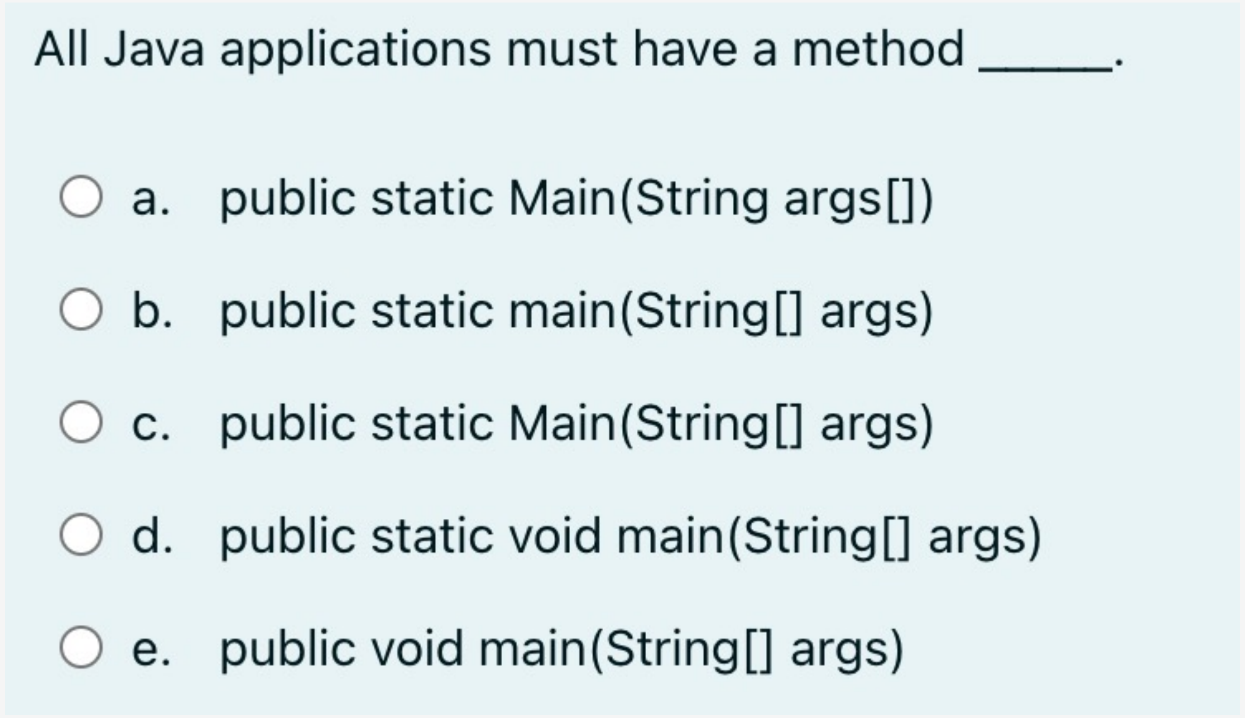
Time: 14:00 - 17:00

Location: E-302

Description: Paper Based - In Person

Passing Strings to the main Method

public static void main(String[] args)

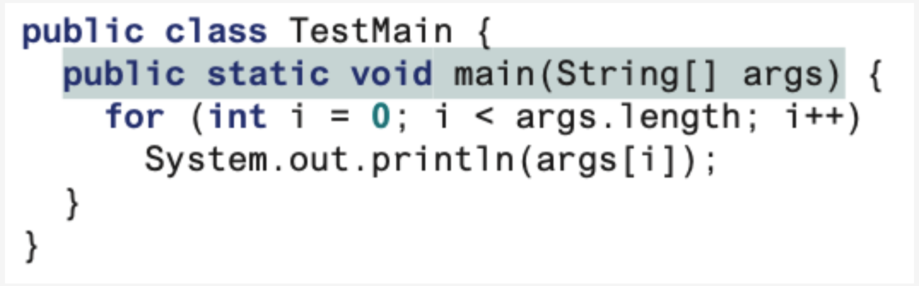


All Java applications must have a method _____.

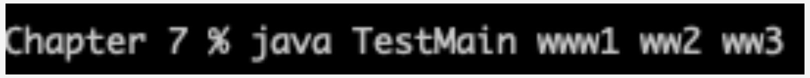
- ○ a. public static Main(String args[])
- ○ b. public static main(String[] args)
- ○ c. public static Main(String[] args)
- ○ d. public static void main(String[] args)
- ○ e. public void main(String[] args)

Passing Strings to the main Method

public static void main(String[] args)

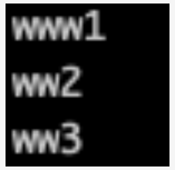


```java
public class TestMain {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++)
      System.out.println(args[i]);
  }
}
```



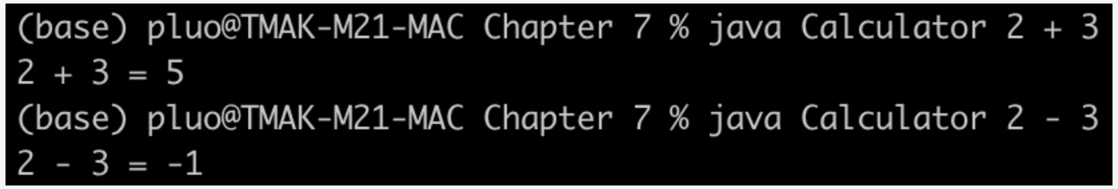
```
Chapter 7 % java TestMain www1 ww2 ww3
```

Output:

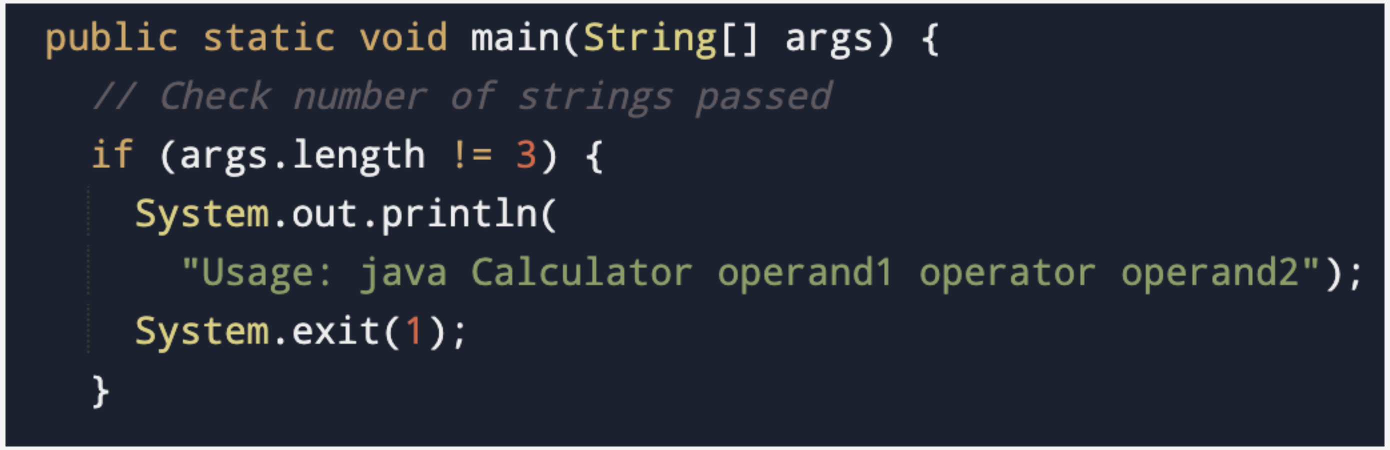


```
www1
ww2
ww3
```

Case Study: Calculator



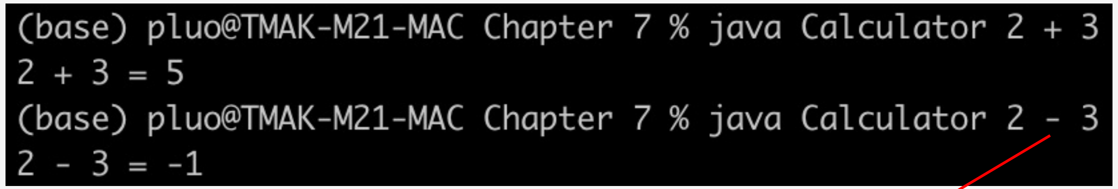
```
(base) pluo@TMAK-M21-MAC Chapter 7 % java Calculator 2 + 3
2 + 3 = 5
(base) pluo@TMAK-M21-MAC Chapter 7 % java Calculator 2 - 3
2 - 3 = -1
```



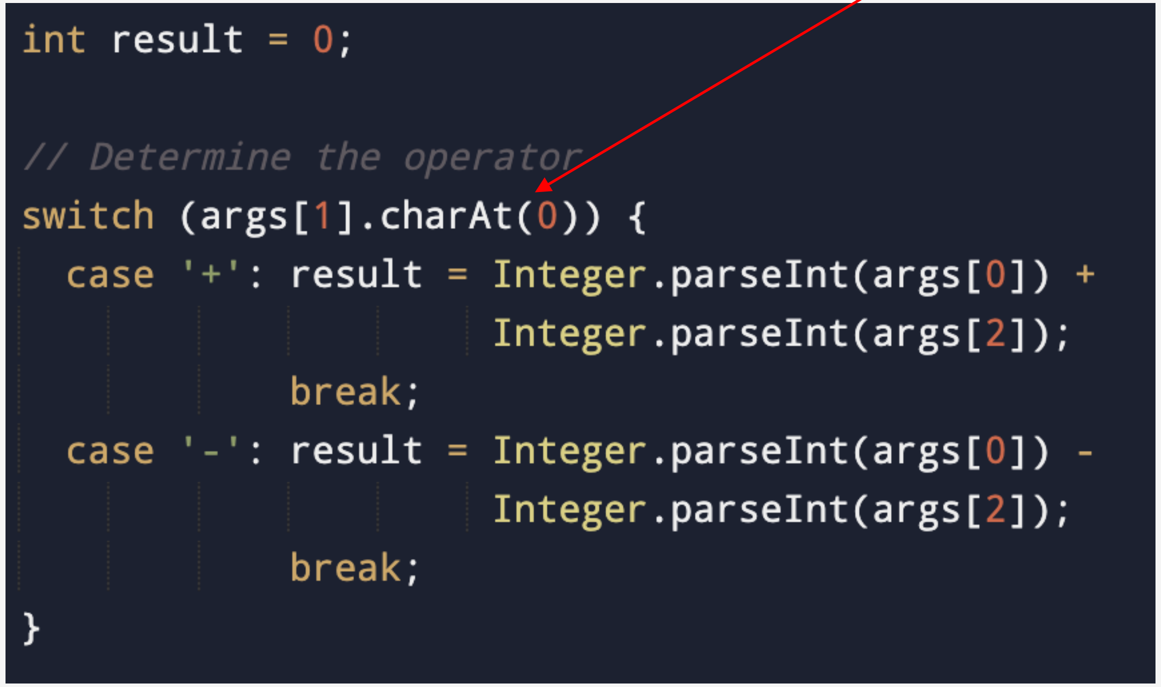
```java
public static void main(String[] args) {
  // Check number of strings passed
  if (args.length != 3) {
    System.out.println(
      "Usage: java Calculator operand1 operator operand2");
    System.exit(1);
  }
```

System.exit(1) will terminate the program

Case Study: Calculator



```
(base) pluo@TMAK-M21-MAC Chapter 7 % java Calculator 2 + 3
2 + 3 = 5
(base) pluo@TMAK-M21-MAC Chapter 7 % java Calculator 2 - 3
2 - 3 = -1
```



```java
int result = 0;

// Determine the operator
switch (args[1].charAt(0)) {
  case '+': result = Integer.parseInt(args[0]) +
                     Integer.parseInt(args[2]);
            break;
  case '-': result = Integer.parseInt(args[0]) -
                     Integer.parseInt(args[2]);
            break;
}
```

Integer.parseInt(args[0]) converts a digital string into an integer. The string must consist of digits. If not, the program will terminate abnormally.

Multidimensional Array

Multidimensional Array

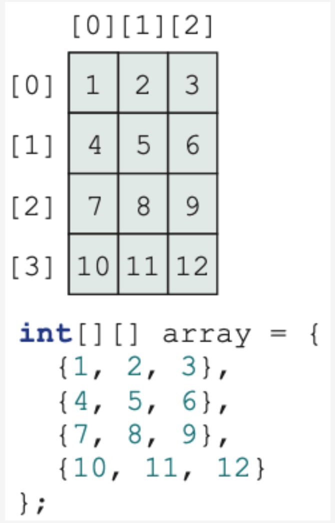
Data in a table or a matrix can be represented using a two-dimensional array.

1	**2**	**3**	**4**	**5**	**6**
1	0	0	0	0	0
2	2	0	0	0	0
3	0	0	0	0	0

Multidimensional Array

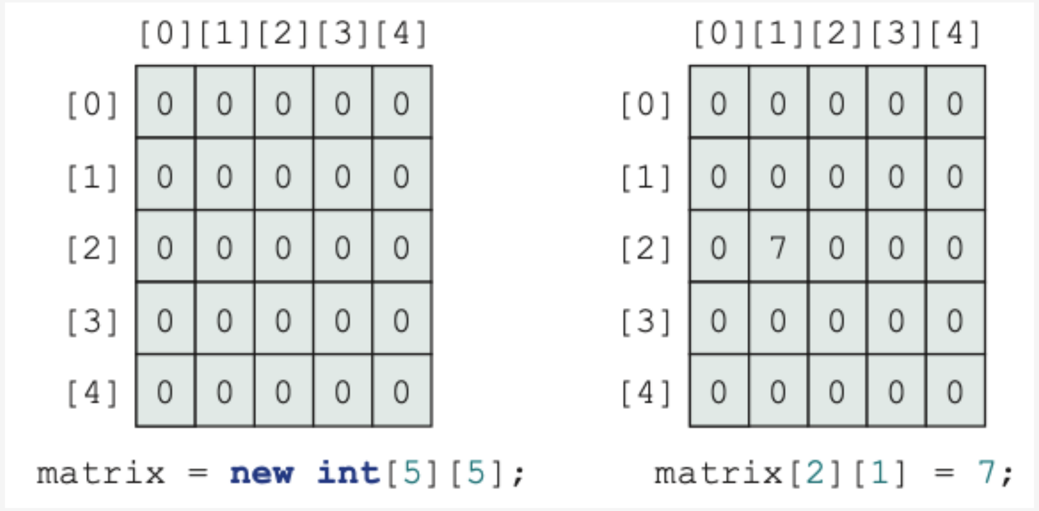
```
elementType[][] arrayRefVar;
or
elementType arrayRefVar[][]; // Allowed, but not preferred
```

```
int[][] matrix;
matrix = new int [4][6]
```



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

Multidimensional Array



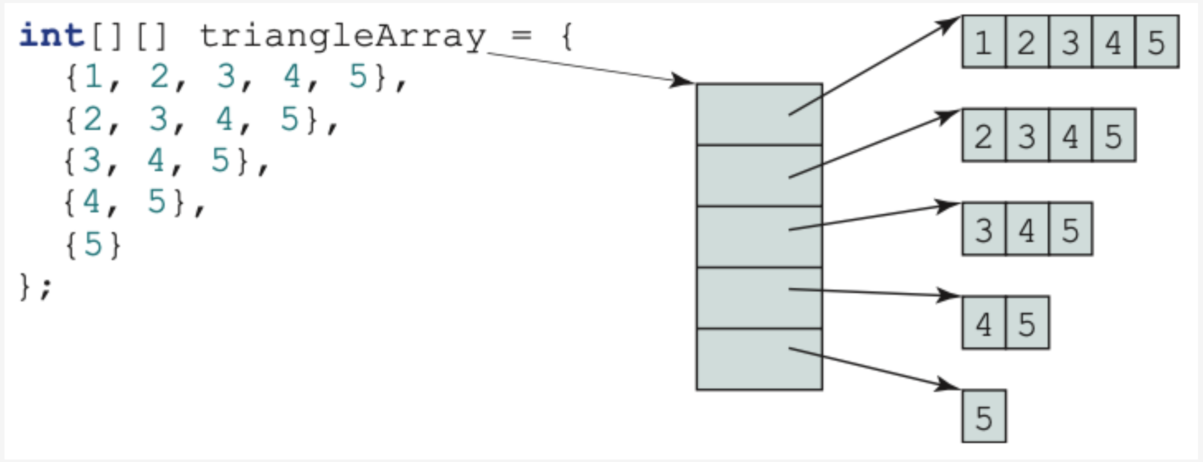
Caution
It is a common mistake to use `matrix[2, 1]`

Multidimensional Array

- A two-dimensional array is actually an array in which each element is a one-dimensional array.
- matrix[0].length: 5
- matrix[1].length: 5
- matrix[2].length: 5

Multidimensional Array

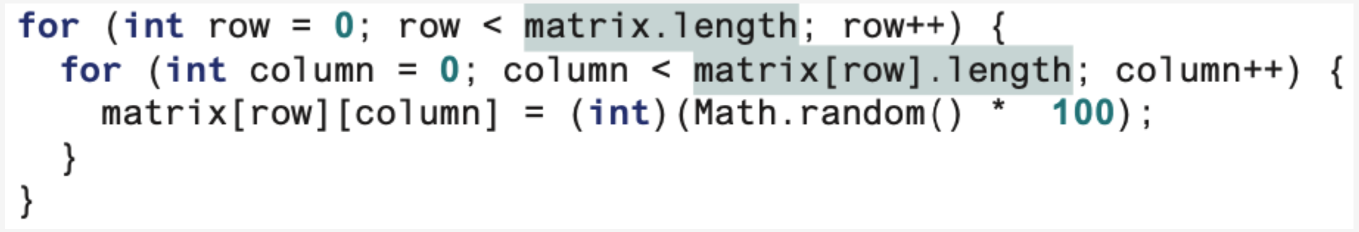
- Since each row in a two-dimensional array is itself an array, the rows can have different lengths!!! -----> Ragged Array



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

```
int[][] triangleArray = new int[5][];
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];
```

Multidimensional Array - Nested Loops



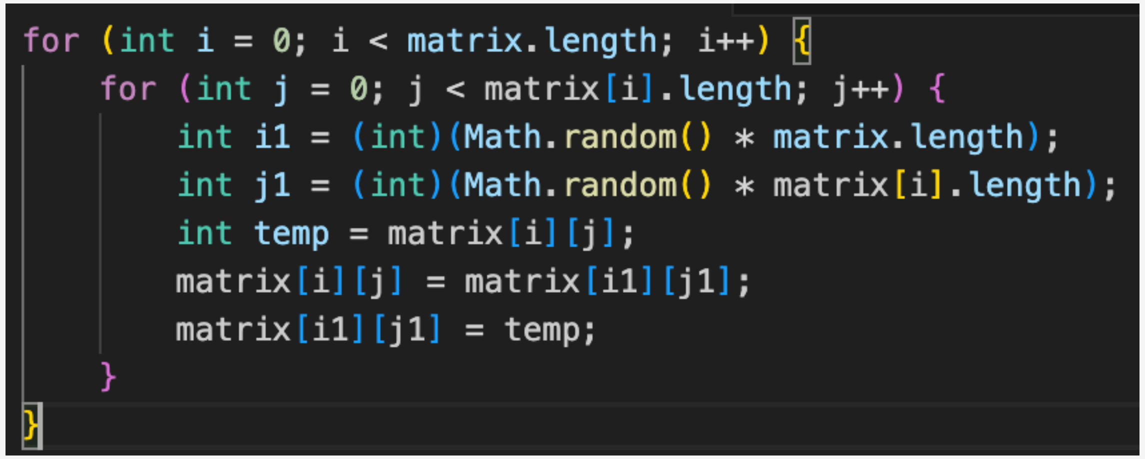
```java
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = (int)(Math.random() *  100);
  }
}
```

$$\begin{array}{c|cccc|}
 & 1 & 2 & \dots & n \\
1 & a_{11} & a_{12} & \dots & a_{1n} \\
2 & a_{21} & a_{22} & \dots & a_{2n} \\
3 & a_{31} & a_{32} & \dots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
m & a_{m1} & a_{m2} & \dots & a_{mn}
\end{array}$$

Case Study: PassTwoDimensionalArray

https://liveexample.pearsoncmg.com/html/PassTwoDimensionalArray.html

Multidimensional Array - Shuffling



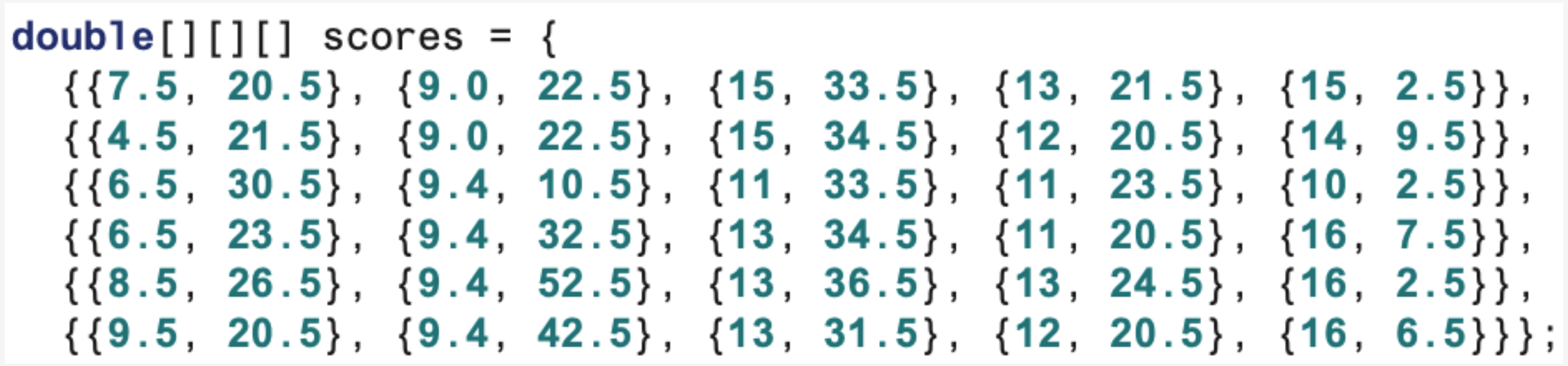
```java
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        int i1 = (int)(Math.random() * matrix.length);
        int j1 = (int)(Math.random() * matrix[i].length);
        int temp = matrix[i][j];
        matrix[i][j] = matrix[i1][j1];
        matrix[i1][j1] = temp;
    }
}
```

Case Study: Grade Questions

Write a program that grades multiple-choice tests.

Multidimensional Array

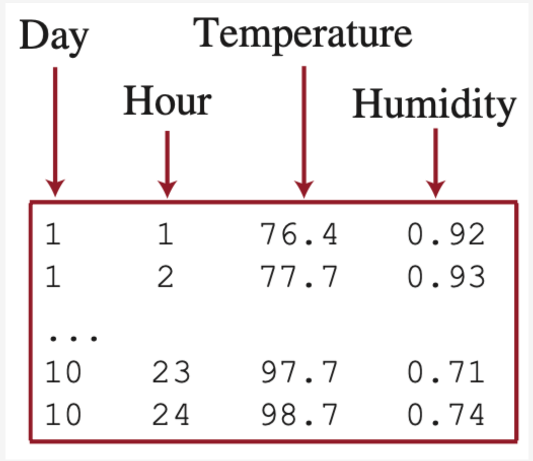
double[][][] scores = new double[6][5][2];



```
double[][][] scores = {
  {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
  {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
  {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
  {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
  {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
  {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

Case Study: Daily Temperature and Humidity

A meteorology station records the temperature and humidity every hour of every day and stores the data for the past 10 days in a text file named Weather.txt. Write a program that calculates the average daily temperature and humidity for the 10 days.

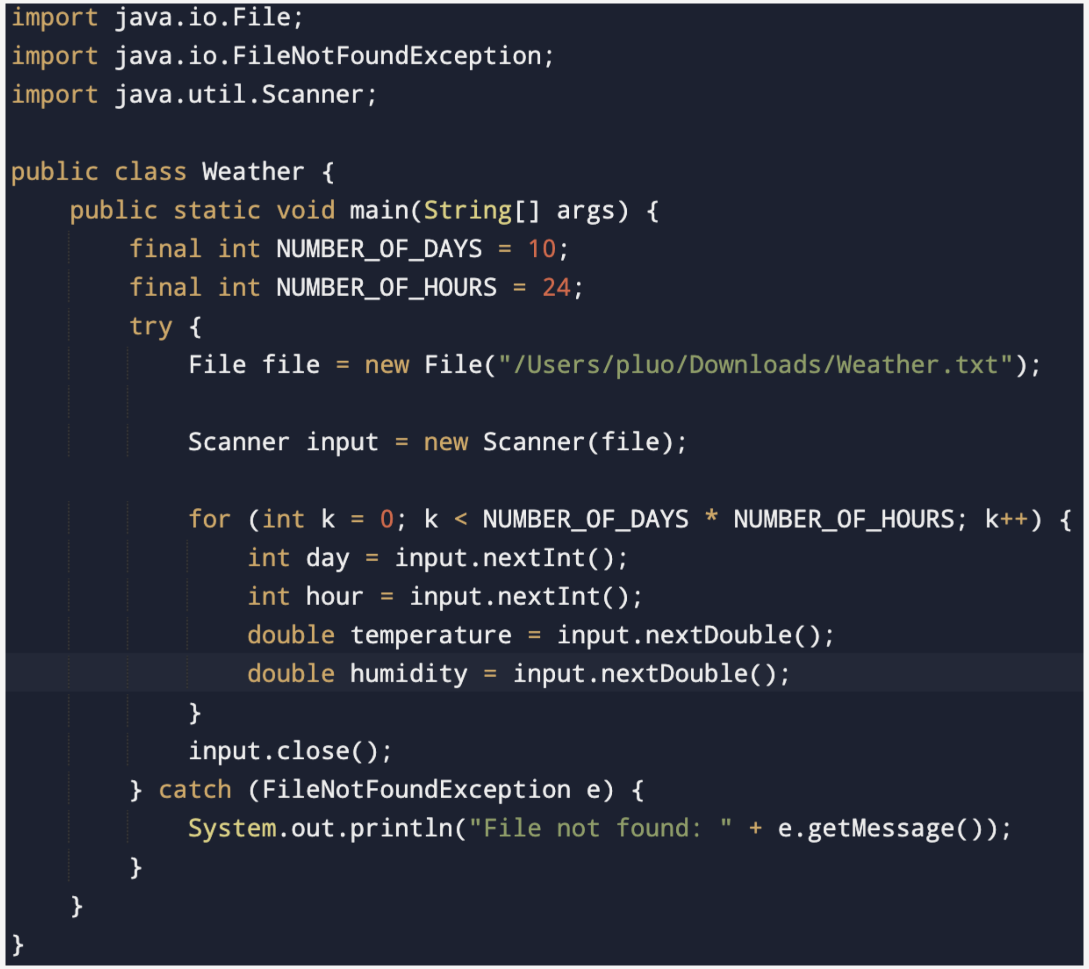


```
1      1     76.4    0.92
1      2     77.7    0.93
...
10     23    97.7    0.71
10     24    98.7    0.74
```

Case Study: Daily Temperature and Humidity

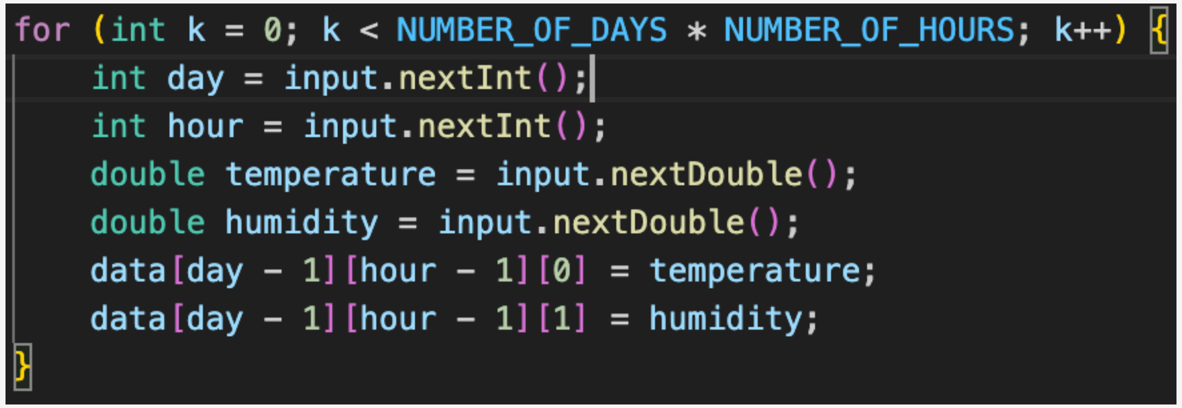
1. Read the Weather.txt file
2. Save the data into a multidimensional array
3. Compute average temperature & humidity
4. Output results

Case Study: Daily Temperature and Humidity



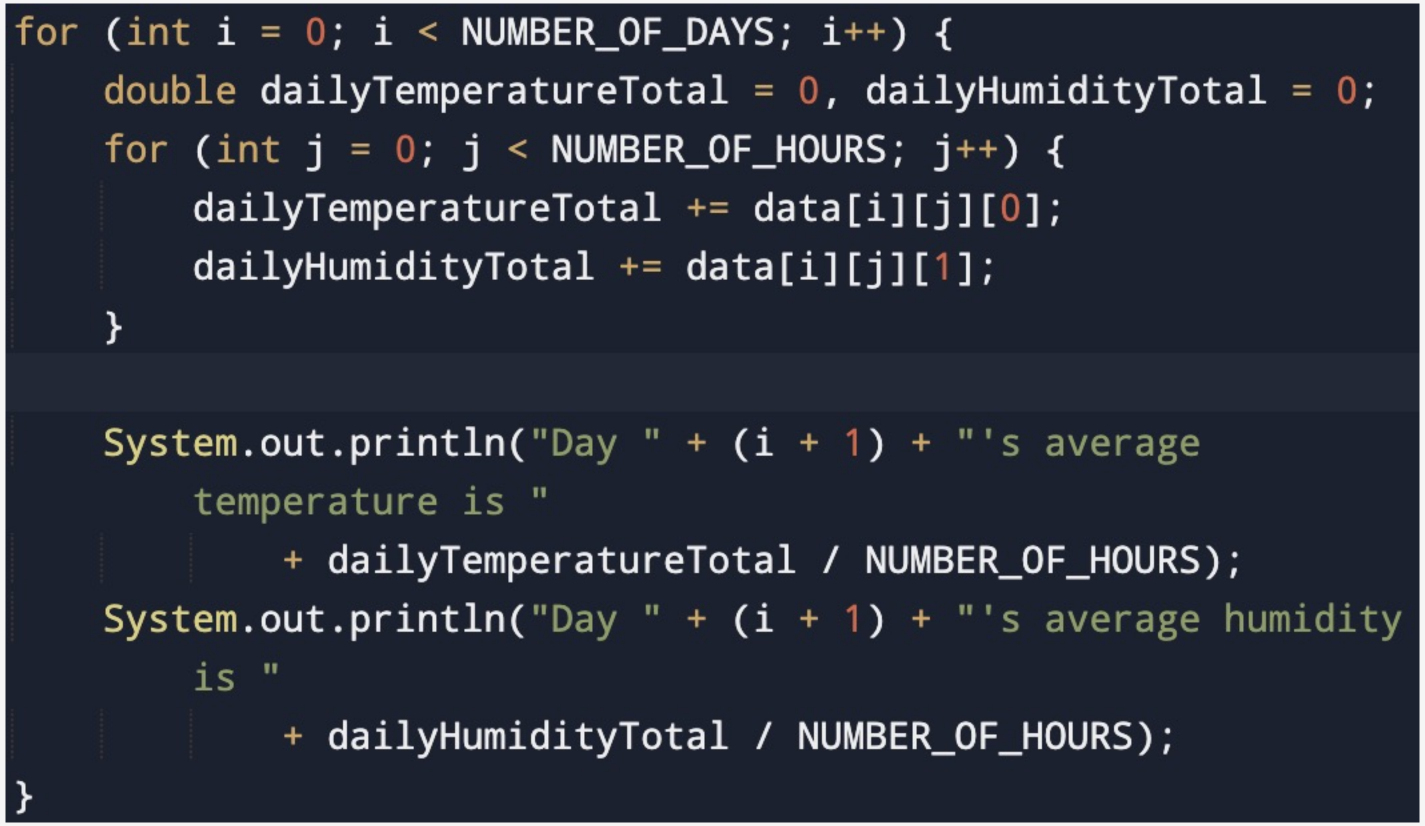
```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Weather {
    public static void main(String[] args) {
        final int NUMBER_OF_DAYS = 10;
        final int NUMBER_OF_HOURS = 24;
        try {
            File file = new File("/Users/pluo/Downloads/Weather.txt");

            Scanner input = new Scanner(file);

            for (int k = 0; k < NUMBER_OF_DAYS * NUMBER_OF_HOURS; k++) {
                int day = input.nextInt();
                int hour = input.nextInt();
                double temperature = input.nextDouble();
                double humidity = input.nextDouble();
            }
            input.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        }
    }
}
```

```java
double[][][] data =
new double[NUMBER_OF_DAYS][NUMBER_OF_HOURS][2];
```



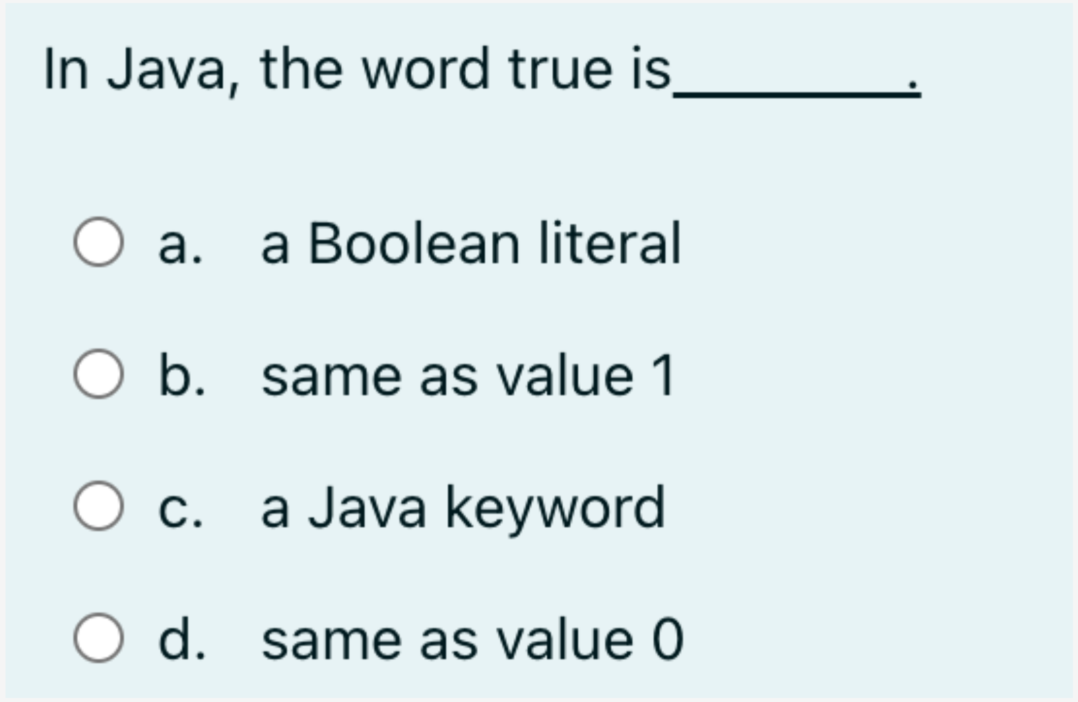
```java
for (int k = 0; k < NUMBER_OF_DAYS * NUMBER_OF_HOURS; k++) {
    int day = input.nextInt();
    int hour = input.nextInt();
    double temperature = input.nextDouble();
    double humidity = input.nextDouble();
    data[day - 1][hour - 1][0] = temperature;
    data[day - 1][hour - 1][1] = humidity;
}
```

Case Study: Daily Temperature and Humidity



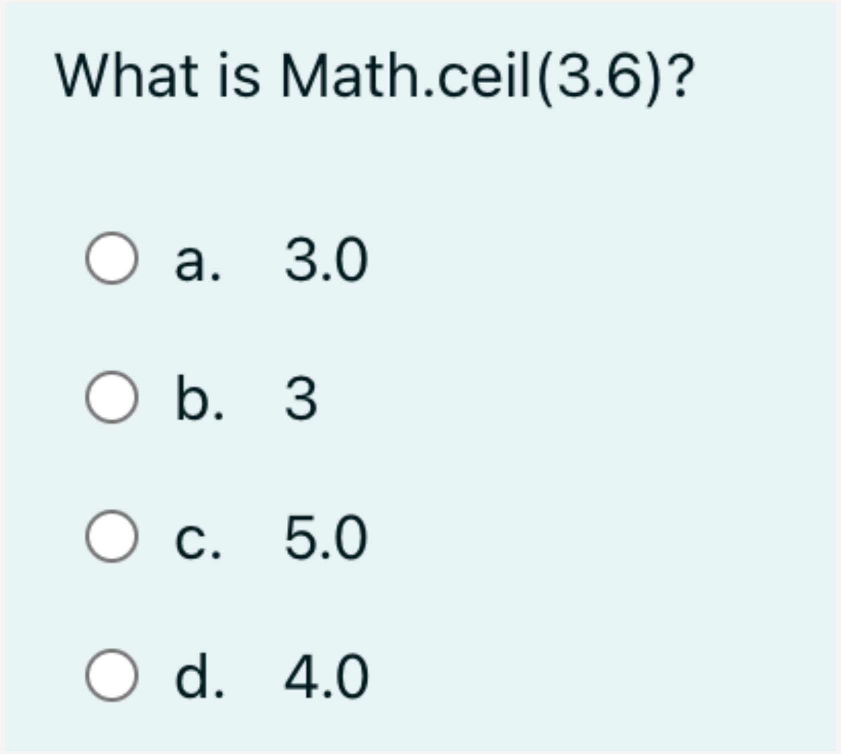
```java
for (int i = 0; i < NUMBER_OF_DAYS; i++) {
    double dailyTemperatureTotal = 0, dailyHumidityTotal = 0;
    for (int j = 0; j < NUMBER_OF_HOURS; j++) {
        dailyTemperatureTotal += data[i][j][0];
        dailyHumidityTotal += data[i][j][1];
    }

    System.out.println("Day " + (i + 1) + "'s average
        temperature is "
            + dailyTemperatureTotal / NUMBER_OF_HOURS);
    System.out.println("Day " + (i + 1) + "'s average humidity
        is "
            + dailyHumidityTotal / NUMBER_OF_HOURS);
}
```

Midterm Exam Questions



In Java, the word true is________.

- ○ a. a Boolean literal
- ○ b. same as value 1
- ○ c. a Java keyword
- ○ d. same as value 0



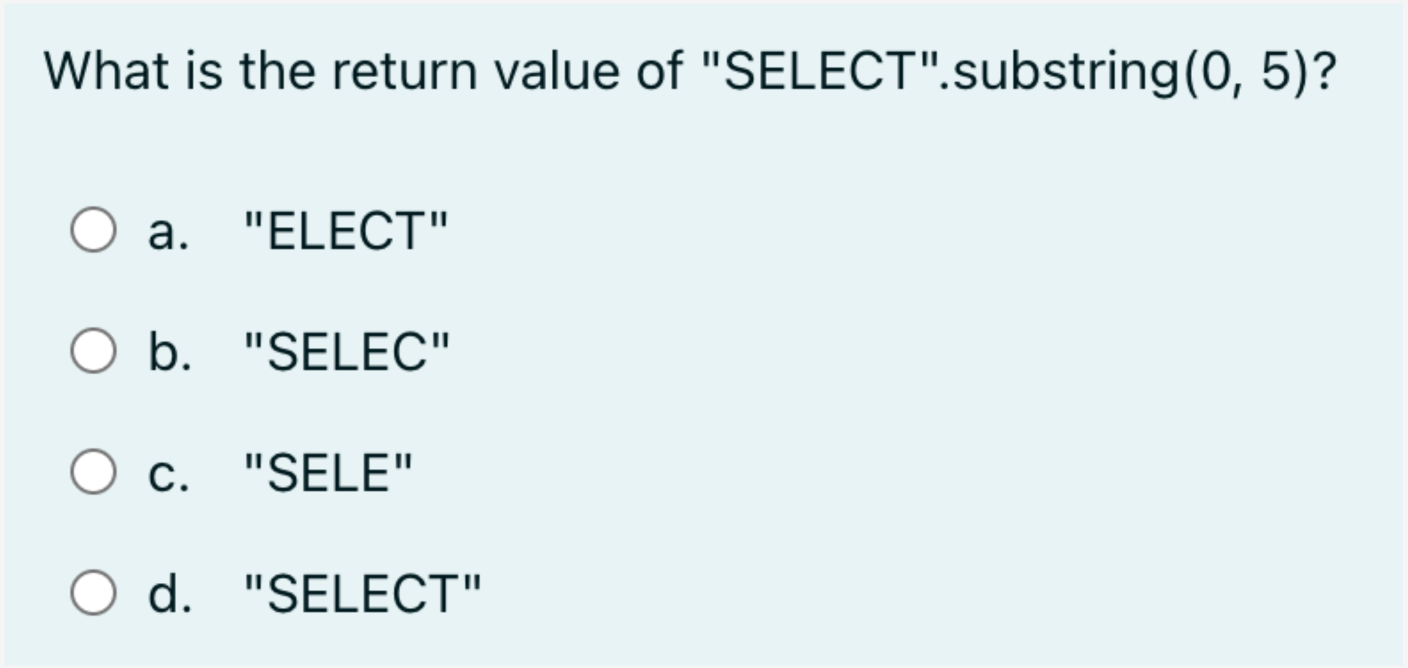
What is Math.ceil(3.6)?

○ a. 3.0

○ b. 3

○ c. 5.0

○ d. 4.0



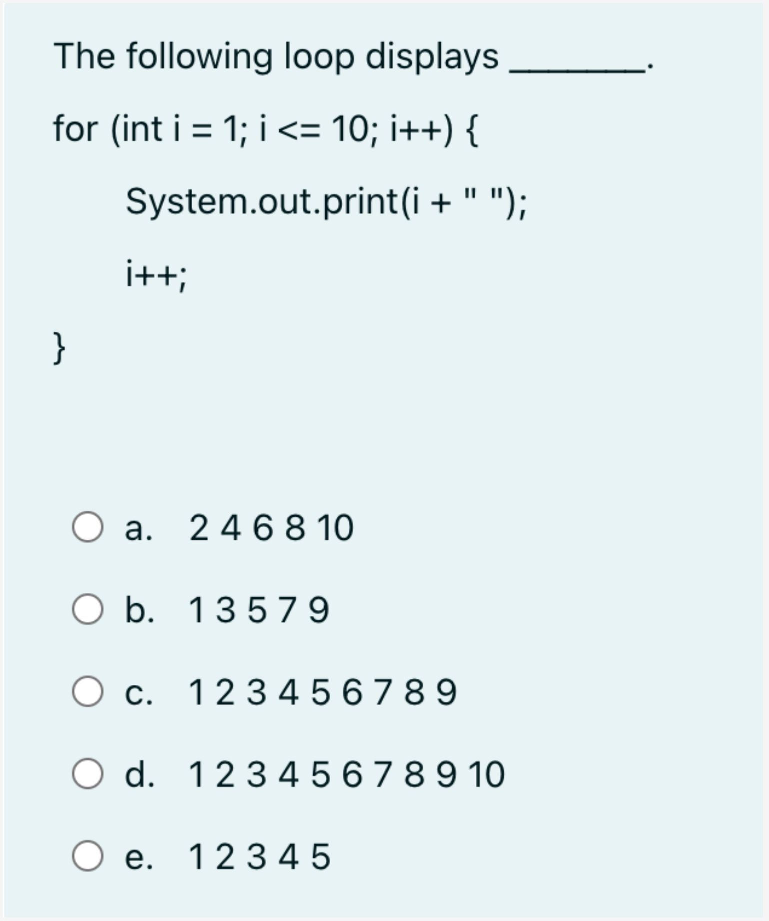
What is the return value of "SELECT".substring(0, 5)?

○ a. "ELECT"

○ b. "SELEC"

○ c. "SELE"

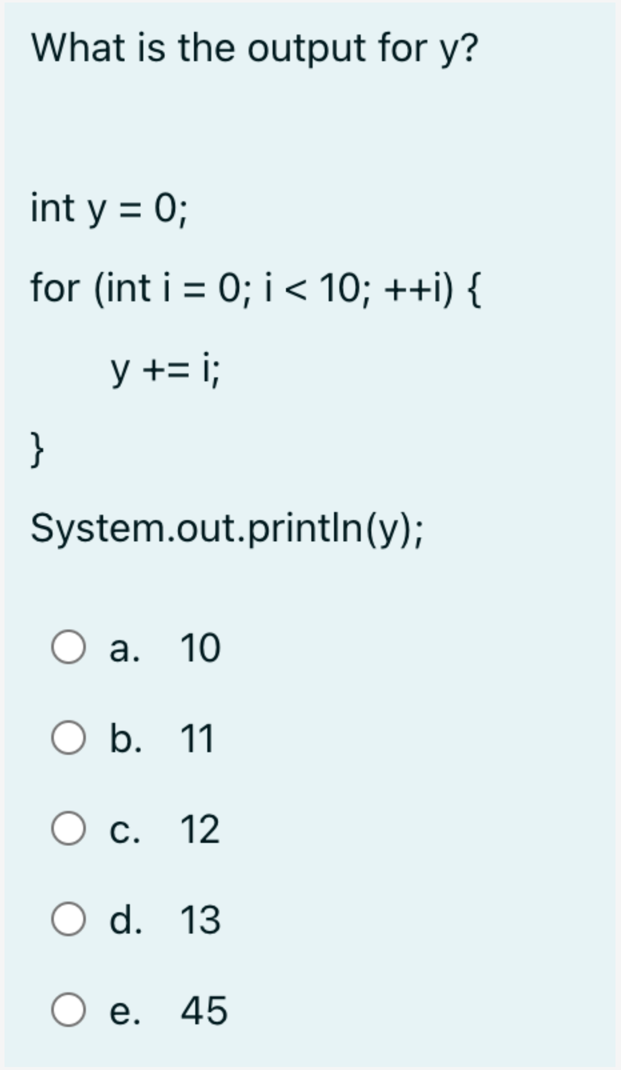
○ d. "SELECT"



The following loop displays ________.

```
for (int i = 1; i <= 10; i++) {
    System.out.print(i + " ");
    i++;
}
```

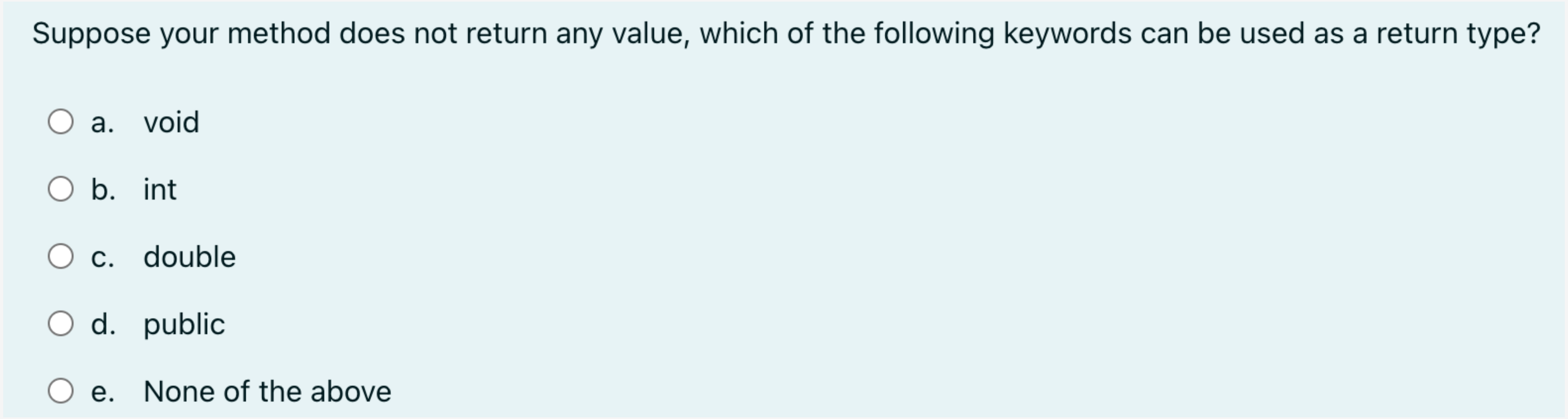
- ○ a. 2 4 6 8 10
- ○ b. 1 3 5 7 9
- ○ c. 1 2 3 4 5 6 7 8 9
- ○ d. 1 2 3 4 5 6 7 8 9 10
- ○ e. 1 2 3 4 5



What is the output for y?

```
int y = 0;
for (int i = 0; i < 10; ++i) {
    y += i;
}
System.out.println(y);
```

- ○ a. 10
- ○ b. 11
- ○ c. 12
- ○ d. 13
- ○ e. 45



Suppose your method does not return any value, which of the following keywords can be used as a return type?

○ a. void

○ b. int

○ c. double

○ d. public

○ e. None of the above

Q&A