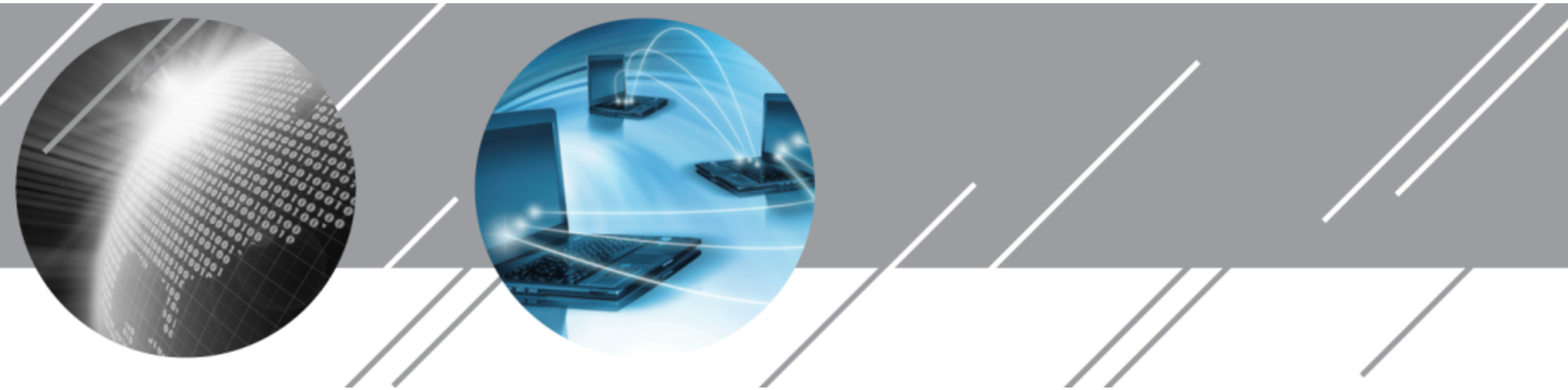
Object Oriented Programming

Introduction to Java

Ch. 7. Arrays



Dept. of Software, Gachon University
Ahyoung Choi, Spring 2021

7.1 Array Basics

Arrays

- An ***array*** is a collection of items of the same type
- Think of as collection of variables of same type
  - Like **a list** of different variables,
    but with a nice, compact way to name them
- An array is a special kind of object in Java

- **Loops repeat things temporally;**
  arrays repeat things spatially

Creating and Accessing Arrays

- Creating an array with 7 variables of type double

```
double[] temperature = new double[7];
```

- To access
  - **temperature[i] ,** where 0 <= i < 7
  - Variables such as temperature[0] that have an integer expression in square brackets are known as:
    - indexed variables, subscripted variables, array elements, or simply elements

Creating and Accessing Arrays

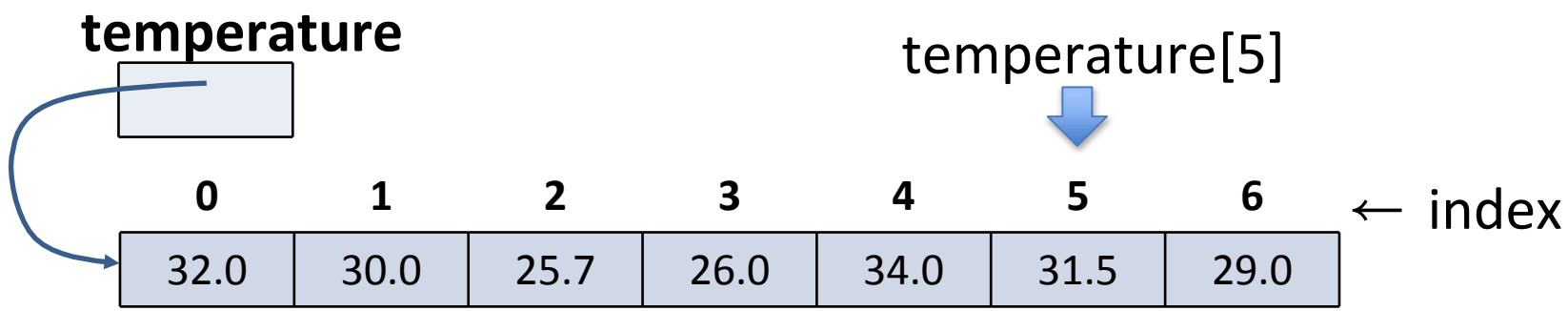
- Syntax:
  - BaseType[] arrayName = new BaseType[length];
  - BaseType can be either a primitive type or a class (String, etc.)
- Examples:
  - int[] pressure = new int[100];
  - int[] pressure;
    pressure = new int[100];
  - int pressure[] = new int[100]; // possible but, discouraged!
  - public static final int readings = 100;
    int[] pressure = new int[readings]; // constant length
  - int numScores = keyboard.nextInt();
    int[] scores = new int[numScores]; // dynamic allocation

Creating and Accessing Arrays

- Create an array with given length saved in constants
  - `public static final int NUMBER_OF_READINGS = 100;`
  - `int[] pressure = new int[NUMBER_OF_READINGS];`
  - Then,
    ```
    for (int i = 0; i < NUMBER_OF_READINGS ; i++)
        scoreSum += keyboard.nextInt();
    ```
- Create an array with user input length
  - `System.out.println("How many scores?");`
  - `int numScores = keyboard.nextInt();`
  - `int[] scores = new int[numScores];`

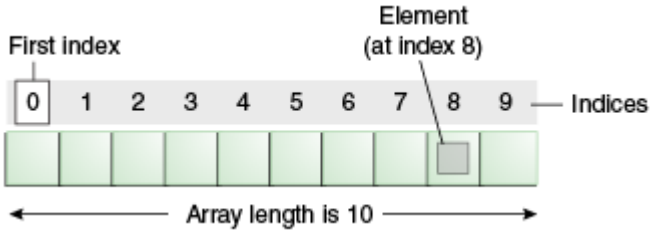
Index

- An **index (**or subscript) is an integer expression inside the square brackets that indicates an array element
  - To access an element use: **ArrayName[index]**
- Array indices begin at **0**
  - The reason is that the array name represents a memory address, and the ith element can be accessed by the address plus i



Array Details

- Array terminology



Array name

```
temperature[n + 2]
```

Index (also called a subscript)

```
temperature[n + 2]
```

Indexed variable (also called an array element, an element, or a subscripted variable)

```
temperature[n + 2]
```

Value of the indexed variable (also called an element of the array)

```
temperature[n + 2] = 32;
```

Finding Length of An Existing Array

- An array is **a special kind of object**
- As an object an array has only one public instance variable
  - Variable `length`
  - Contains number of elements in the array
- `length` is equal to the length of the array
  - `Pet[] pets = new Pet[20];`
  - `pets.length` has the value 20

- You cannot change the value of *length* because it is **final**

Finding Length of An Existing Array

- Do not be OUT OF BOUND!
  - Indices must be in bound
    - double[] entry = new double[5]; // from [0] to [4]
    - entry[5] = 3.7; // ERROR! Index out of bound

Question: Array Indexing

```
public static final int NUMBER_OF_EMPLOYEES = 100;
. . .
int[] hours = new int[NUMBER_OF_EMPLOYEES];
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter hours worked for each employee:");
for (int index = 0; index < hours.length; index++);
{
   System.out.println("Enter hours for employee " +
                      (index + 1));
   hours[index] = keyboard.nextInt();
}
```

VS

```
int[] hours = new int[NUMBER_OF_EMPLOYEES + 1];
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter hours worked for each employee:");
for (int index = 1; index <hours.length; index++);
{
   System.out.println("Enter hours for employee" + index);
   hours[index] = keyboard.nextInt();
}
```

It is OK to *waste* element 0

- There is no answer, but The bottom line is that all programmers collaborating on a project should use the same coding practices.

More About Array Indices

- Index of first array element is 0
- Last valid Index is `arrayName.length - 1`
- Array indices must be within bounds to be valid
  - When program tries to access outside bounds, run time error occurs

Initializing Arrays

- Initialize an array when it is declared
  - int[] scores = { 68, 97, 102 };
- Equivalent to:
  - int[] scores = new int[] { 68, 97, 102 };
  - int[] scores = new int[3];
    scores[0] = 68;
    scores[1] = 97;
    scores[2] = 102;
- You can use for-loop
  - for (int i = 0; i < 100; i ++) count[i] = 0;

Initializing Arrays

- For primitive types:
  - `int[] myIntArray = new int[3];`
  - `int[] myIntArray = {1,2,3};`
  - `int[] myIntArray = new int[]{1,2,3};`

- For classes, for example String,
  - `String[] myStringArray = new String[3];`
  - `String[] myStringArray = {"a","b","c"};`
  - `String[] myStringArray = new String[]{"a","b","c"};`

Lab: Creating and Accessing Arrays

- class ArrayOfTemperatures
  - Read temperatures from the user and shows which are above and which are below the average of all the temperatures.

```
How many temperatures do you have?
3
Enter 3 temperatures:
32
26.5
27
The average temperature is 28.5
The temperatures are
32.0 above average
26.5 below average
27.0 below average
Have a nice week.
```

```java
import java.util.Scanner;
public class ArrayOfTemperatures2
{
    public static void main (String [] args)
    {
        Scanner keyboard = new Scanner (System.in);
        System.out.println ("How many temperatures do you have?");
        int size = keyboard.nextInt ();
        double [] temperature = [      ?????    ];
        // Read temperatures and compute their average:
        System.out.println ("Enter " + temperature.length +
                " temperatures:");
        double sum = 0;
        for (int index = 0 ; index < [redacted] ; index++)
        {
            ???
        }
        double average = [redacted];
        System.out.println ("The average temperature is " +
                average);
        // Display each temperature and its relation to the average:
        System.out.println ("The temperatures are");
        for (int index = 0 ; index < [redacted] ; index++)
        {
            ???
        }
        System.out.println ("Have a nice week.");
    }
}
```

```
How many temperatures do you have?
3
Enter 3 temperatures:
32
26.5
27
The average temperature is 28.5
The temperatures are
32.0 above average
26.5 below average
27.0 below average
Have a nice week.
```

7.2 Arrays in Classes and Methods

Arrays of Objects

- You create **an array of objects (Array object)** like this:
  - `Student[] students = new Student[35];`
- **NOTE: Each of the elements of students is not yet an object !**
- You have to instantiate each individual one
  - `students[0] = new Student();`
  - `students[1] = new Student();`
- …or do this in a loop

Arrays of Objects

```
Student[] group1= new Student[3];
for (int i = 0; i < group1.length; i++) {
    group1[i] = new Student();
}
```

	[0]	[1]	[2]
group1	**0**	**0**	**0**

Arrays of Objects

```
Student[] group1= new Student[3];
for (int i = 0; i < group1.length; i++) {
    group1[i] = new Student();
}
```

	[0]	[1]	[2]
group1	**1045**	**2548**	**2836**

1045	No name 0
2548	No name 0
2836	No name 0

Lab: array object

- Class diagram for class **`SalesReporter`**

```
SalesReporter
- highestSales: double
- averageSales: double
- team: SalesAssociate[]
- numberOfAssociates: int

+ getData(): void
+ computeStats(): void
+ displayResults(): void
```

```
Average sales per associate is $32000.0
The highest sales figure is $50000.0

The following had the highest sales:
Name: Natalie Dressed
Sales: $50000.0
$18000.0 above the average.

The rest performed as follows:
Name: Dusty Rhodes
Sales: $36000.0
$4000.0 above the average.

Name: Sandy Hair
Sales: $10000.0
$22000.0 below the average.
```

SalesAssociate.java

Copy and use this file

```java
package arrayTest;
import java.util.Scanner;
public class SalesAssociate{
    private String name;
    private double sales;
    public SalesAssociate( )    {
        name = "No record";
        sales = 0;
    }
    public SalesAssociate(String initialName, double initialSales)    {
        set(initialName, initialSales);
    }
    public void set(String newName, double newSales)   {
        name = newName;
        sales = newSales;
    }
    public void readInput( )    {
        System.out.print("Enter name of sales associate: ");
        Scanner keyboard = new Scanner(System.in);
        name = keyboard.nextLine( );

        System.out.print("Enter associate's sales: $");
        sales = keyboard.nextDouble( );
    }
    public void writeOutput( )    {
        System.out.println("Name: " + name);
        System.out.println("Sales: $" + sales);
    }
    public String getName( )    {        return name;    }
    public double getSales( )    {        return sales;    }
}
```

```java
package arrayTest;
import java.util.Scanner;
  public class SalesReporter {
      private double highestSales;
      private double averageSales;
      private SalesAssociate[] team;  //The array object
      private int numberOfAssociates; //Same as team.length
    // Reads the number of sales associates and data for each one.
    public void getData( )    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter number of sales associates:");
        numberOfAssociates = keyboard.nextInt( );
        team = new SalesAssociate[numberOfAssociates + 1];//We won't use tea
[redacted]
}    }
    // Computes the average and highest sales figures.
    public void computeStats( )    {
        double nextSales = team[1].getSales( );
        highestSales = nextSales;
        double sum = nextSales;
[redacted]
        }
        averageSales = sum / numberOfAssociates;    }
    // Displays sales report on the screen.
    public void displayResults( )    {
        System.out.println("Average sales per associate is $" + a
        System.out.println("The highest sales figure is $" + high
        System.out.println("The following had the highest sales:"
        for (int i = 1; i <= numberOfAssociates; i++)  {
            double nextSales = team[i].getSales( );
            if (nextSales == highestSales) team[i].writeOutput( )
         }    }
  public static void main(String[] args) {
        SalesReporter clerk = new SalesReporter( );
        clerk.getData( );
        clerk.computeStats( );
        clerk.displayResults( ); } }
```

SalesReporter
– highestSales: double – averageSales: double – team: SalesAssociate[] - numberOfAssociates: int
+ getData(): void + computeStats(): void + displayResults(): void

Create instances of array objects

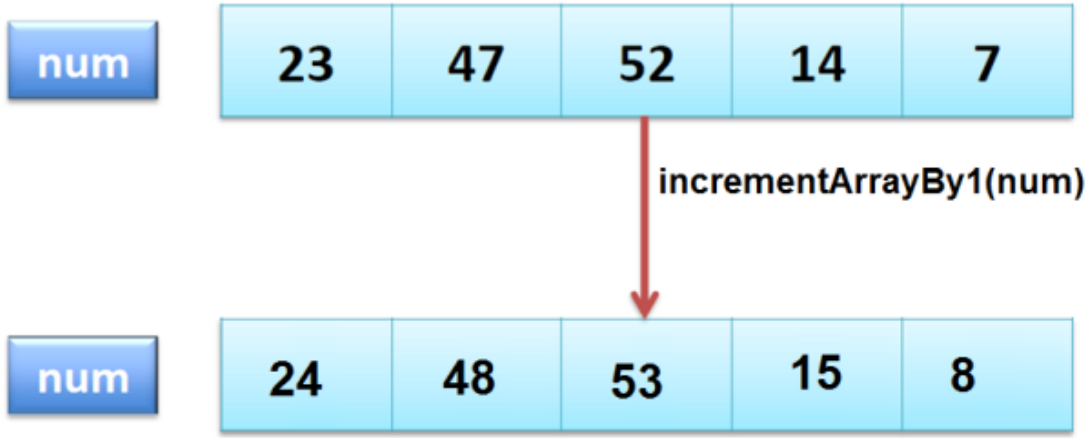
```
Average sales per associate is $32000.
The highest sales figure is $50000.0
The following had the highest sales:
Name: Natalie Dressed
Sales: $50000.0
$18000.0 above the average.
The rest performed as follows:
Name: Dusty Rhodes
Sales: $36000.0
$4000.0 above the average.
Name: Sandy Hair
Sales: $10000.0
$22000.0 below the average.
```

Arrays as Parameters

```
public static void incrementArray(int[] arr) {
    for(int i = 0; i < arr.length; i++) arr[i]++;
}

public void testArray() {
    int[] a = new int[] { 23, 47, 52, 14, 7 };
    incrementArray(a);
    System.out.println(a[3]);
}
```

Prints 15, why?



Arrays as Parameters

```
double[] a = new double[10];
double[] b = new double[30];

SampleClass.incrementArrayBy2(a);
SampleClass.incrementArrayBy2(b);
```

```
public class SampleClass
{
    public static void incrementArrayBy2(double[] anArray)
    {
        for (int i = 0; i <anArray.length; i++)
            anArray[i] = anArray[i] + 2;
    }
    <The rest of the class definition goes here.>
}
```

Array Parameters Do Not Specify the Array Length

Then, you can understand..

```
public static void main(String[] args)
```

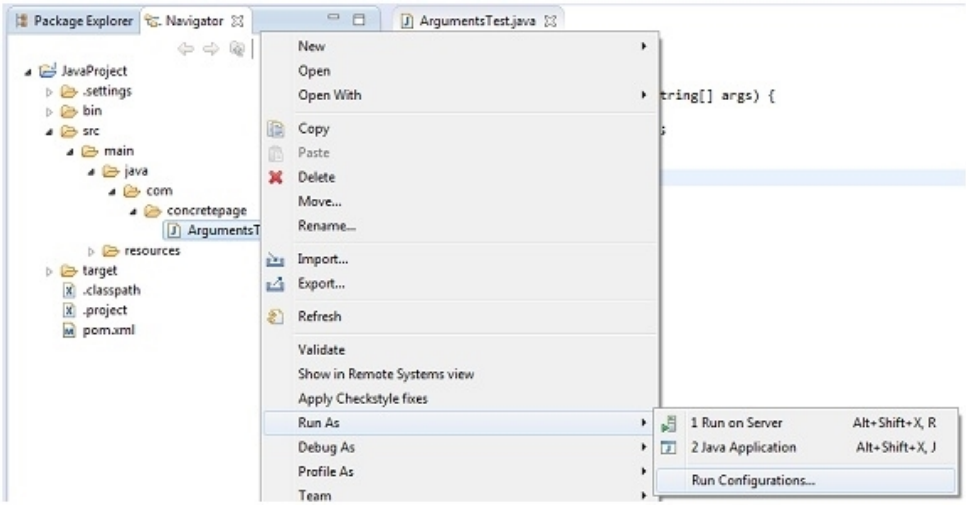
Argument for the Method `main`

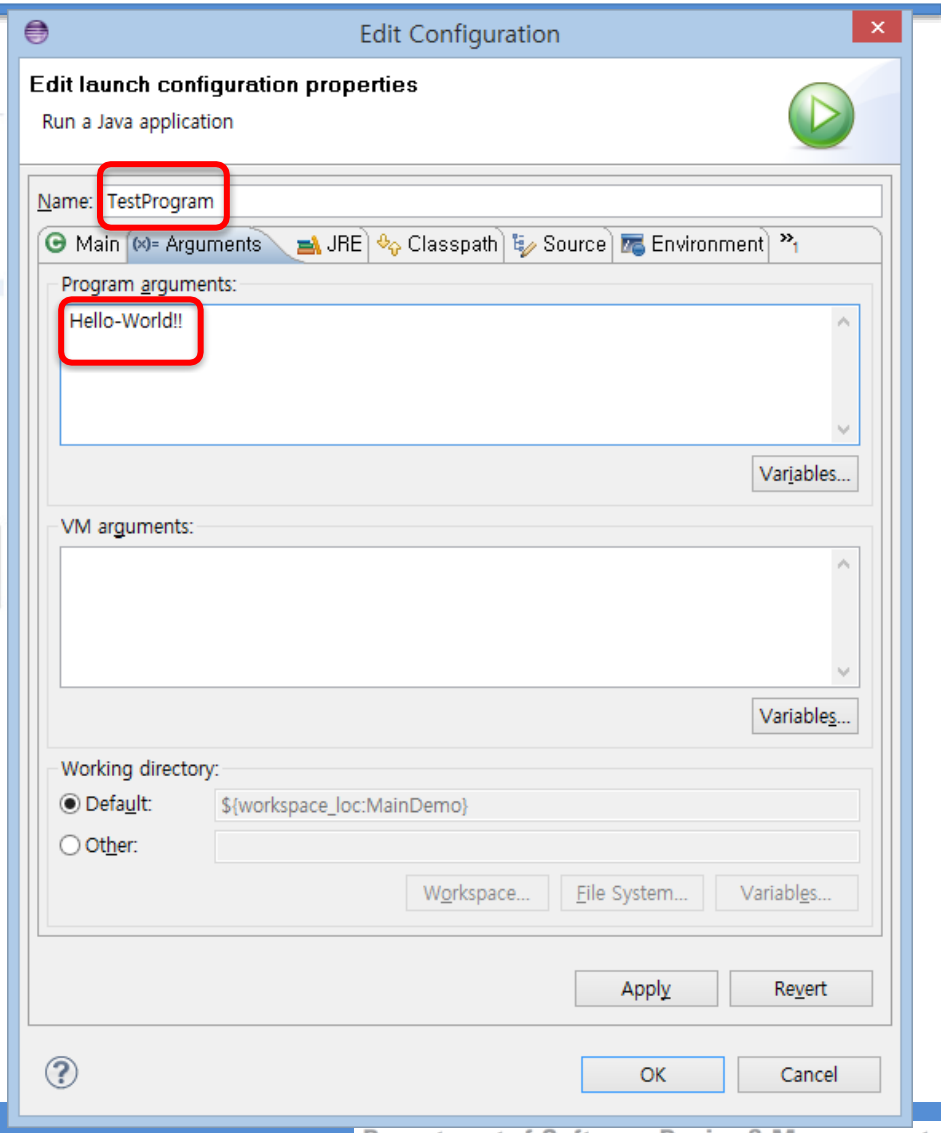
```
public static void main(String[] args)
```

- **Command-Line Arguments**
- When you run a program, you can provide additional strings (if you like) e.g., > `java TestProgram Sally Smith`
  - This allows the user to specify configuration information when the application is launched.

```
public class TestProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello" + args[0] + " " + args[1]);
    }
}
```

How to Pass Command Line Arguments to Java Program in Eclipse



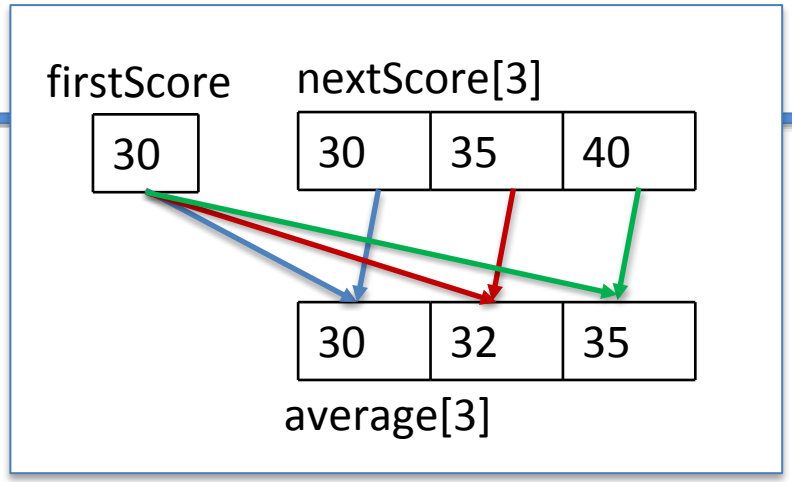


Arrays as Return Types

- Create an array and return it

```
public double[] buildArray(int len) {
    double[] retArray = new double[len];
    for (int i = 0; i < retArray.length; i++) {
        retArray[i] = i * 1.5;
    }
    return retArray;
}
```

Lab: return array




```
import java.util.Scanner;
public class ReturnArrayDemo{
    public static void main(String[] args)    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter your score on exam 1:");
        int firstScore = keyboard.nextInt( );
        int[] nextScore = new int[3];
        for (int i = 0; i < nextScore.length; i++)
             nextScore[i] = firstScore + 5 * i;

        double[] averageScore = getArrayOfAverages(firstScore, nextScore );

        for (int i = 0; i < nextScore.length; i++)        {
           System.out.println("If your score on exam "+ (i+2) + " is" + nextScore[i]);
           System.out.println("your average will be " +  averageScore[i]);
        }
    }
    public static double[] getArrayOfAverages(int firstScore, int[] nextScore)   {
        double[] temp = new double[nextScore.length];
        for (int i = 0; i < temp.length; i++)
            temp[i] = getAverage(firstScore, nextScore[i ]);
        return temp;
    }
    public static double getAverage(int n1, int n2)
    {
       return (n1 + n2) / 2.0;
    }
}
```

7.4 Sorting, Searching Arrays

Sorting

- Given an array of values, order them from lowest to highest (ascending order) or from highest to lowest (descending order)
  - Before sorting:

4	7	3	9	6	2	8

  - After sorting:

2	3	4	6	7	8	9

- Sorting is an extremely important question in computer science

Selection Sort

- Ascending order selection sort
  - Take the remaining array except the front
  - Find the minimum value in array
  - Swap the minimum value with the front
  - Then minimum value put in the front

  - Repeat until we meet the end of array

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
7	6	11	17	3	15	5	19	30	14

7	6	11	17	3	15	5	19	30	14

3	6	11	17	7	15	5	19	30	14

3	6	11	17	7	15	5	19	30	14

3	5	11	17	7	15	6	19	30	14

•

•

•

3	5	6	7	11	14	15	17	19	30

Hint: Swap

```
public static void swap(int i,int j, int [] a)
{
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

- This method will swap the value of a[i] and a[j]
- Remember that "a" is a memory address
  - None of a, i and j are changed in this code
  - Only a[i] and a[j] are changed – they are not local!

Lab: Selection Sort

- View implementation of selection sort, listing 7.10
  `class ArraySorter`
- View demo program, listing 7.11
  `class SelectionSortDemo`

```
Array values before sorting:
7 5 11 2 16 4 18 14 12 30
Array values after sorting:
2 4 5 7 11 12 14 16 18 30
```

test class

```
Array values before sorting:
7 5 11 2 16 4 18 14 12 30
Array values after sorting:
2 4 5 7 11 12 14 16 18 30
```

```java
public class SelectionSortDemo {
    public static void main(String[] args)    {
        int[] b = {7, 5, 11, 2, 16, 4, 18, 14, 12, 30};

        display (b, "before");
        ArraySorter.selectionSort(b);
        display (b, "after");
    }

    public static void display(int[] array, String when){
            System.out.println("Array values " + when + " sorting:");
            for (int i = 0; i < array.length; i++)
                System.out.print(array[i] + " ");
            System.out.println( );
    }
}
```

```
public class ArraySorter{
    public static void selectionSort(int[] anArray)    {
        [illegible]
    }
    }

    //   Returns the index of the smallest value in the portion of the array
    private static int getIndexOfSmallest(int startIndex, int[] a)    {
        [illegible]
        return indexOfMin;
    }

    // swap ith and jth element in an array
    private static void swap(int i, int j, int[] a)    {
        [illegible]
    }
}
```

Other Sorting Algorithms

- Sorting algorithms
  - Selection sort is the simplest
  - Many other efficient algorithms for large arrays
- Java Class Library provides for efficient sorting
  - The class *java.util.Arrays* provides multiple (overloaded) static sort methods
  - Arrays.sort(anArray);
  - Arrays.sort(anArray, first, last);

Insertion Sort

- Assignment: Survey this algorithm!
  - No grading, self-study!

7.5 Multidimensional Arrays

2D Arrays

- Array (or 1D array) gives you a list of variables
  - int[] score = new int[5] gives you score[0], score[1], …
- 2D array gives you a table of variables
  - int[][] t = new int[3][4];

Row index 3

Column index 2

Indices	0	1	2	3	4	5
0	$1050	$1055	$1060	$1065	$1070	$1075
1	$1103	$1113	$1124	$1134	$1145	$1156
2	$1158	$1174	$1191	$1208	$1225	$1242
3	$1216	$1239	$1262	$1286	$1311	$1335
4	$1276	$1307	$1338	$1370	$1403	$1436
5	$1340	$1379	$1419	$1459	$1501	$1543
6	$1407	$1455	$1504	$1554	$1606	$1659
7	$1477	$1535	$1594	$1655	$1718	$1783
8	$1551	$1619	$1689	$1763	$1838	$1917
9	$1629	$1708	$1791	$1877	$1967	$2061

`table[3][2]` *has a value of* `1262`

Declaring and Creating 2D Arrays

- Two pairs of square brackets means 2D
  - `int[][] table = new int[3][4];`
- or
  - `int[][] table;`
  - `table = new int[3][4];`

table[0][0]	table[0][1]	table[0][2]	table[0][3]
table[1][0]	table[1][1]	table[1][2]	table[1][3]
table[2][0]	table[2][1]	table[2][2]	table[2][3]

Two-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.

- Example:

row col

```
rating[0][2] = 2
rating[1][3] = 8
```

rating

movie (second index)

reviewer (first index)	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Accessing 2D arrays

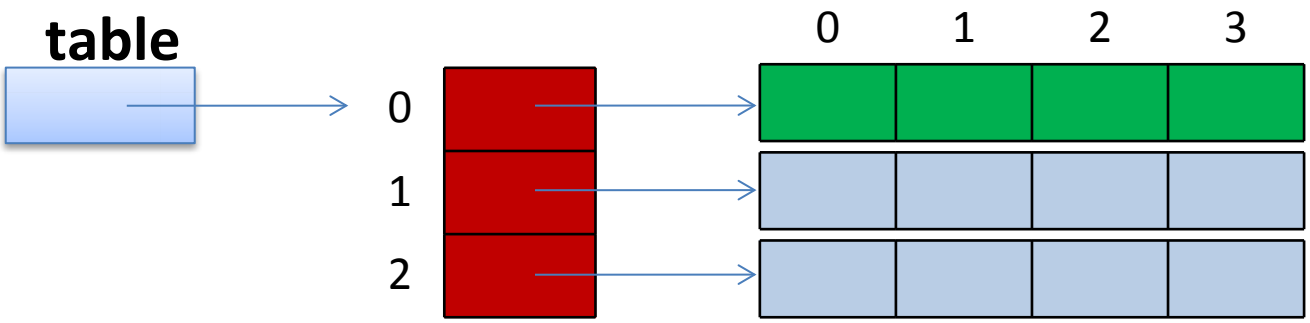
- We use a nested loop to access 2D arrays
  - cf. We used a loop to access 1D arrays

```
int[][] table = new int[4][3];
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 3; j++) {
        table[i][j] = i * 3 + j;
        System.out.println(table[i][j]);
    }
}
```

```
for (int r = 0; r < table.length; r++) {
    for (int c = 0; c < table[r].length; c++) {
        table[r][c] = r * 5 + c;
        System.out.println(table[r][c]);
    }
}
```

Length of 2D arrays

- Java's representation of a 2D array: array of array
- E.g., int[][] table = new int[3][4];
- Array *table* is actually 1D array of type int[]
  - table.length → 3;
  - table[0].length → 4



Array of Array!

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- `int[] scores = new int[5];`
  - scores is a one-dimensional array
  - base type is `int`
- `int[][] table = new int[4][3];`
  - table is still in fact a one-dimensional array
  - base type is `int[]`
- Array table is actually 1 dimensional of type `int[]`
  - It is an array of arrays

Initializing 2D arrays

```
int[][] hours = new int[5][3];
hours[0][0] = 8; hours[0][1] = 0; hours[0][2] = 9;
hours[1][0] = 8; hours[1][1] = 0; hours[1][2] = 9;
hours[2][0] = 8; hours[2][1] = 8; hours[2][2] = 8;
hours[3][0] = 8; hours[3][1] = 8; hours[3][2] = 4;
hours[4][0] = 8; hours[4][1] = 8; hours[4][2] = 8;
```

```
int[][] hours = {
    { 8, 0, 9 },
    { 8, 0, 9 },
    { 8, 8, 8 },
    { 8, 8, 4 },
    { 8, 8, 8 }
};
```

Multidimensional Arrays

- You can have more than two dimensions
  - `int[][][] cube = new int[4][3][4];`

- Use more nested loops to access all elements
  - `for (int i…)`
    - `for (int j…)`
      - `for (int k…)`

Ragged Arrays

- Not necessary for all rows to be of the same length
- Example:

```
int[][] b;
b = new int[3][];
b[0] = new int[5]; //First row,  5 elements
b[1] = new int[7]; //Second row, 7 elements
b[2] = new int[4]; //Third row,  4 elements
```

```
int[][] rating = { {3,5,7,9}, {4,2},
                   {5,7,8,6}, {6} };
```

3	5	7	9
4	2		
5	7	8	6
6			

Multidimensional-Array Parameters and Returned Values

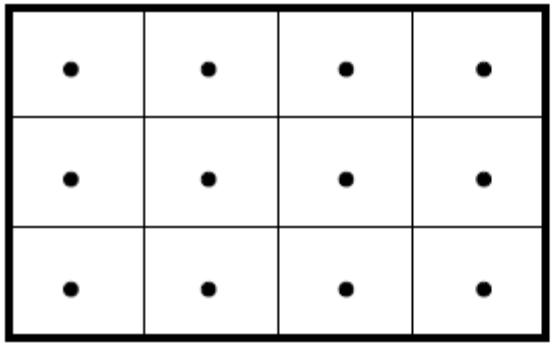
```
Balances for Various Interest Rates Compounded Annually
(Rounded to Whole Dollar Amounts)
Years 5.00% 5.50% 6.00% 6.50% 7.00% 7.50%
1 $1050 $1055 $1060 $1065 $1070 $1075
2 $1103 $1113 $1124 $1134 $1145 $1156
3 $1158 $1174 $1191 $1208 $1225 $1242
4 $1216 $1239 $1262 $1286 $1311 $1335
5 $1276 $1307 $1338 $1370 $1403 $1436
6 $1340 $1379 $1419 $1459 $1501 $1543
7 $1407 $1455 $1504 $1554 $1606 $1659
8 $1477 $1535 $1594 $1655 $1718 $1783
9 $1551 $1619 $1689 $1763 $1838 $1917
10 $1629 $1708 $1791 $1877 $1967 $2061
```

```
public class InterestTable2 {
    public static final int ROWS = 10;
    public static final int COLUMNS = 6;
    public static void main (String [] args )     {
        int [] [] table = new int [ROWS] [COLUMNS];
        for (int row = 0 ; row < ROWS ; row++)
            for (int column = 0 ; column < COLUMNS ; column++)
                table [row] [column] = getBalance (1000.00, row + 1, (5 + 0.5 * column));
        System.out.println ("Balances for Various Interest Rates " + "Compounded Annually");
        System.out.println ("(Rounded to Whole Dollar Amounts)");
        System.out.println ("Years 5.00% 5.50% 6.00% 6.50% 7.00% 7.50%");
        showTable (table);
    }
  public static void showTable (int [] [] anArray)   {
        for (int row = 0 ; row < ROWS ; row++)    {
            System.out.print ((row + 1) + " ");
            for (int column = 0 ; column < COLUMNS ; column++)  System.out.print ("$" + anArray [row]
[column] + " ");
        }
    }
    public static int getBalance (double startBalance, int years, double rate )       {
        double runningBalance = startBalance;
        for (int count = 1 ; count <= years ; count++)     runningBalance = runningBalance * (1 + rate /
100);
        return (int) (Math.round (runningBalance));
    }
}
```

2D Array of Object References

- Recall that creating an array of object references fills the array with null values
- Example:

Student[][] class1= new Student[3][4];

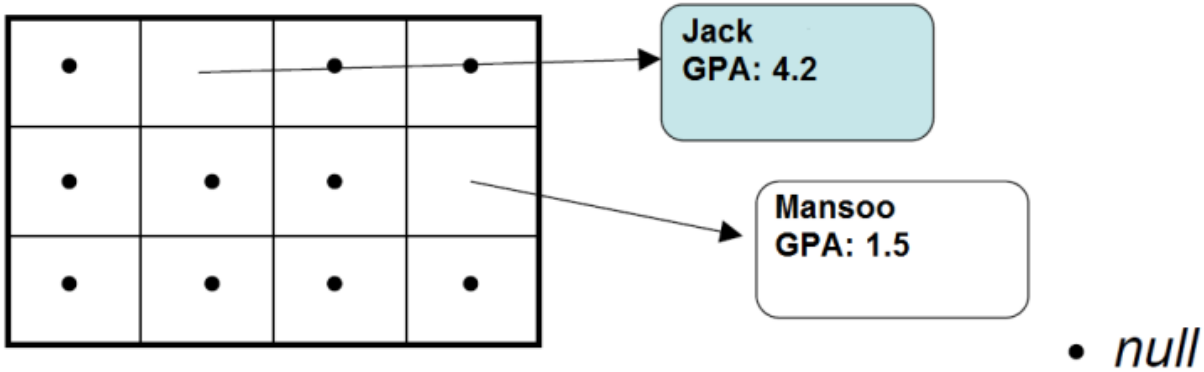


• *null*

2D Array of Object References

- Need to create the objects and assign the references to the array elements
- Example:

class1[0][1] = new Student("Jack", 4.2);
class1[1][3] = new Student("Mansoo", 1.5);



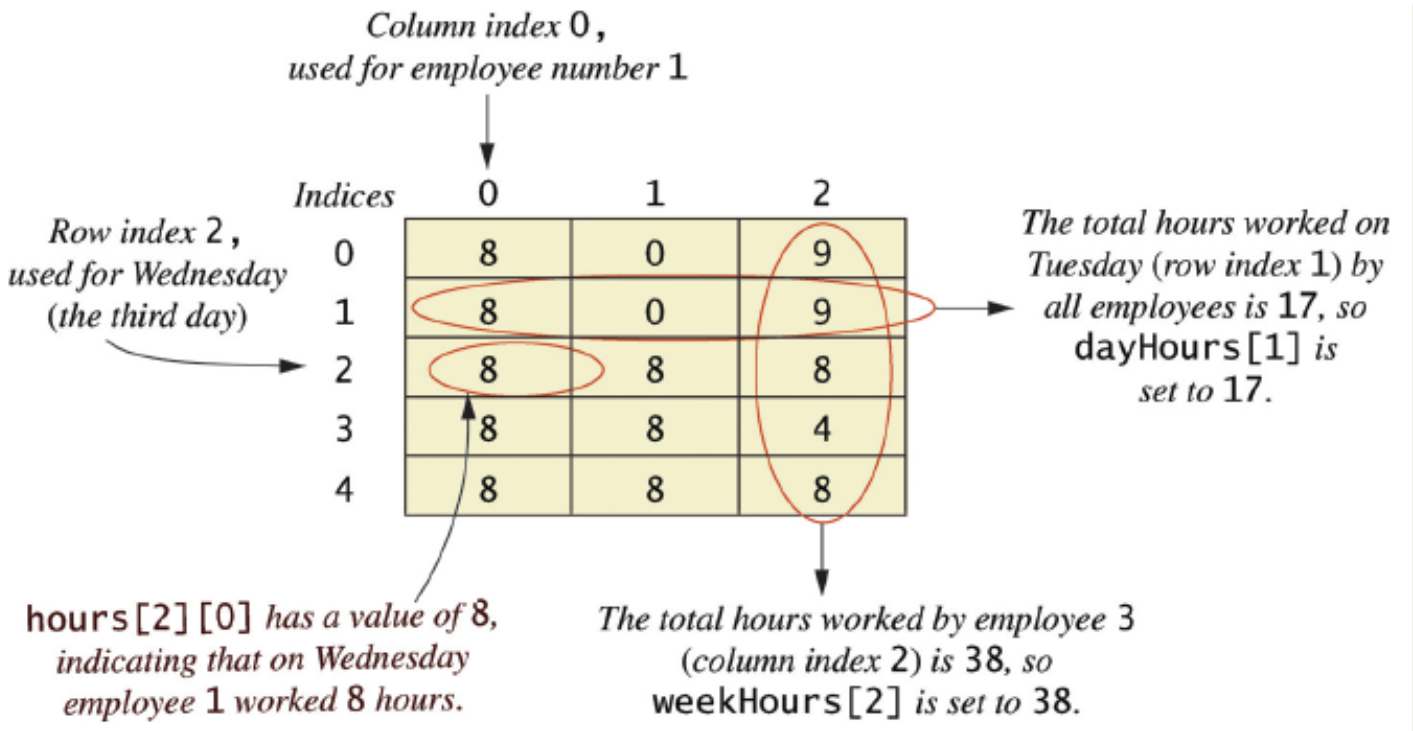
Lab: Employee Time Records

- Reads hours worked for each employee on each day of the work week into the two-dimensional array hours.
- Computes the total weekly hours for each employee and the total daily hours for all employees combined.
- Define
  - Two-dimensional array stores hours worked
    - For each employee
    - For each of 5 days of work week
  - Array is private instance variable of class
- View sample program, listing 7.14 `class TimeBook`

Employee	1	2	3	Totals
Monday	8	0	9	17
Tuesday	8	0	9	17
Wednesday	8	8	8	24
Thursday	8	8	4	20
Friday	8	8	8	24
Total =	40	24	38	

Programming Example

- Figure 7.8 Arrays for the class **`TimeBook`**



```java
public class TimeBook{
    private int numberOfEmployees;
    private int[][] hours;  //hours[i][j] has the hours for employee j on day i.
    private int[] weekHours;//weekHours[i] has the week's hours worked for
    private int[] dayHours; //dayHours[i] has the total hours worked by all
    private static final int NUMBER_OF_WORKDAYS = 5;
    private static final int MON = 0;
    private static final int TUE = 1;
    private static final int WED = 2;
    private static final int THU = 3;
    private static final int FRI = 4;

    public static void main(String[] args) {
        final int NUMBER_OF_EMPLOYEES = 3;
        TimeBook book = new TimeBook(NUMBER_OF_EMPLOYEES);
        book.setHours( );
        book.update( );
        book.showTable( );
    }
    public TimeBook(int theNumberOfEmployees)    {
```

// 생성자: 고용자 숫자 입력 후 배열 생성

```
    }
    public void setHours( ) {
        hours[0][0] = 8;  hours[0][1] = 0;  hours[0][2] = 9;
        hours[1][0] = 8;  hours[1][1] = 0;  hours[1][2] = 9;
        hours[2][0] = 8;  hours[2][1] = 8;  hours[2][2] = 8;
        hours[3][0] = 8;  hours[3][1] = 8;  hours[3][2] = 4;
        hours[4][0] = 8;  hours[4][1] = 8;  hours[4][2] = 8;
    }
```

// 근무시간 설정, 입력 없이 초기화

```
    public void update( )    {
        computeWeekHours( );
        computeDayHours( );
    }
```

// 주당 총 근무시간 계산
// 인당 총 근무시간 계산

.. 이어서...

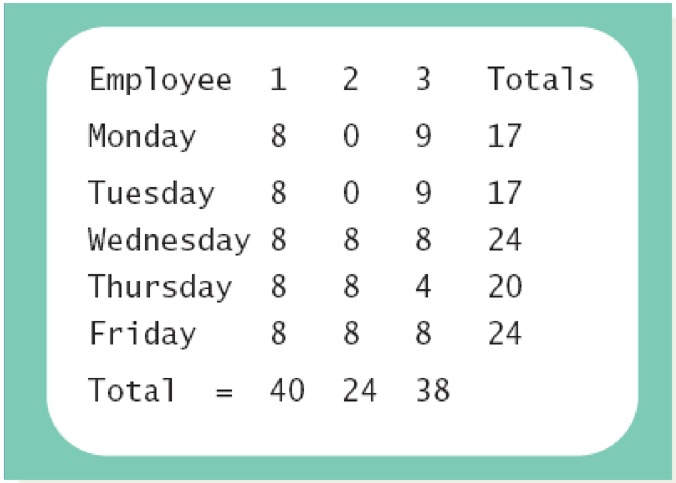
```
private void computeWeekHours( )    {//Process one employee:
[redacted]
        }
    }
    private void computeDayHours( )    {//Process one day (for all employees):
[redacted]
        }
    }
    public void showTable( )    {
      // heading
        System.out.print("Employee  ");
        for (int employeeNumber = 1; employeeNumber <= numberOfEmployees; employeeNumber++)
            System.out.print(employeeNumber + "   ");
        System.out.println("Totals");
        System.out.println( );

      // row entries
        for (int day = MON; day <= FRI; day++) {
            System.out.print(getDayName(day) + " ");
            for (int column = 0; column < hours[day].length; column++)
                System.out.print(hours[day][column] + "   ");
            System.out.println(dayHours[day]);
        }
        System.out.println( );

        System.out.print("Total  =  ");
        for (int column = 0; column < numberOfEmployees; column++)
            System.out.print(weekHours[column] + "  ");
        System.out.println( );
    }
```

// 인당 총 근무시간 계산
// 세로로 더하기

// 주당 총 근무시간 계산
// 가로로 더하기



```
Employee   1   2   3   Totals
Monday     8   0   9   17
Tuesday    8   0   9   17
Wednesday  8   8   8   24
Thursday   8   8   4   20
Friday     8   8   8   24
Total   =  40  24  38
```

… 이어서

```
private String getDayName(int day)   {
        String dayName = null;
        switch (day)  {
            case MON:
                dayName = "Monday   ";                    // 기타 코드 : 출력시 요일 이름 출력
                break;
            case TUE:
                dayName = "Tuesday  ";
                break;
            case WED:
                dayName = "Wednesday";
                break;
            case THU:
                dayName = "Thursday ";
                break;
            case FRI:
                dayName = "Friday   ";
                break;
            default:
                System.out.println("Fatal Error.");
                System.exit(0);
                break;
        }
        return dayName;
    }
}
```