Topic Covered:
One dimensional Arrays (1D Array), Two – dimensional Array (2D-Array), Introduction to JAVA String, Local variable type inference, Java User Input (Scanner class) and Arithmetic Operators.

ARRAYS

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.

DECLARING ARRAY VARIABLES

To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable:

```
dataType[] arrayRefVar;
or
dataType arrayRefVar[];
```

The style dataType[] arrayRefVar is preferred. The style dataType arrayRefVar[] comes from the C/C++ language and was adopted in Java to accommodate C/C++ programmers.
You can declare multiple arrays of the same data type in one statement by inserting a comma after each array name, using this syntax:

```
dataType[] arrayName1, arrayName2;
```

CREATING ARRAYS

You can create an array by using the new operator with the following syntax:

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

The above statement does two things:

- It creates an array using new dataType[arraySize].
- It assigns the reference of the newly created array to the variable arrayRefVar.

Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement, as shown below:

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

For Example:

int[] arr = new int[10];
```

Alternatively you can create arrays as follows:

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

The number of elements in the array is determined by the number of values in the initialization list. The values can be an expression, for example, nine and nine + 2.
For example, this statement declares and instantiates an array of odd numbers:

```
int nine = 9;
int[] oddNumbers = {1 ,3 ,5 ,7 ,nine ,nine+2, 13};
```

PROGRAM 1: Demonstrate a one-dimensional array.

```
class Array {
public static void main(String args[]) {
int month_days[];
month_days = new int[12];
month_days[0] = 31;
month_days[1] = 28;
month_days[2] = 31;
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
System.out.println("April has " + month_days[3] + " days.");
}
}
```

PROGRAM 2: Creates an initialized array of integers:

```
class AutoArray {
public static void main(String args[]) {
int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31 };
System.out.println("April has " + month_days[3] + " days.");
}
}
```

PROGRAM 3: Average an array of values.

```
class Average {
public static void main(String args[]) {
double nums[] = {10.1, 11.2, 12.3, 13.4, 14.5};
double result = 0;
int i;
for(i=0; i<5; i++)
result = result + nums[i];
System.out.println("Average is " + result / 5);
}
}
```

ACCESSING ARRAY ELEMENTS

Elements of an array are accessed using index, within the array. The index of the first element in the array is always 0 and the index of the last element is always 1 less than the number of elements. Arrays have a read-only, integer instance variable, length, which holds the number of elements in the array. To access the number of elements in an array named arrayName, use this syntax:

```
arr.length
```

Thus, to access the last element of an array, use this syntax:

```
arr [arr.length – 1 ]
```

PROGRAM 4: Simple Array

```
class TestArray {

   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // Print all the array elements
      for (int i = 0; i < myList.length; i++) {
         System.out.println(myList[i] + " ");
      }

      // Summing all elements
      double total = 0;
      for (int i = 0; i < myList.length; i++) {
         total += myList[i];
      }
      System.out.println("Total is " + total);
    }
}
```

EXERCISE 3-1:
Write down a program which has an array of 10 integer numbers then write the code to find the largest and smallest number from that array

MULTIDIMENSIONAL ARRAYS IN JAVA

Multidimensional Arrays can be defined in simple words as array of arrays.

TWO – DIMENSIONAL ARRAY (2D-ARRAY)

Two – dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding. Syntax to Declare Multidimensional Array in Java:

```
dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];
```

Example to instantiate Multidimensional Array in Java

```
int[][] arr=new int[3][3];//3 row and 3 column
```

When you allocate memory for a multidimensional array, you need only specify the memory for the first (leftmost) dimension. You can allocate the remaining dimensions separately. For example, this following code allocates memory for the first dimension of twoD when it is declared. It allocates the second dimension manually.

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

When you allocate dimensions manually, you do not need to allocate the same number of elements for each dimension. You can create a two-dimensional array in which the sizes of the second dimension are unequal.

PROGRAM 5: Demonstrate a two-dimensional array.

```
class TwoDArray {
public static void main(String args[]) {
int twoD[][]= new int[4][5];
int i, j, k = 0;
for(i=0; i<4; i++)
for(j=0; j<5; j++) {
twoD[i][j] = k;
k++;
}
for(i=0; i<4; i++) {
for(j=0; j<5; j++)
System.out.print(twoD[i][j] + " ");
System.out.println();
}
}
}
```

PROGRAM 6: Declaring and initializing 2D array

```
class TestArray{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//printing 2D array
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}
}
}
```

PROGRAM 7: Manually allocate differing size second dimensions.

```
class TwoDAgain {
public static void main(String args[]) {
int twoD[][] = new int[4][];
twoD[0] = new int[1];
twoD[1] = new int[2];
twoD[2] = new int[3];
twoD[3] = new int[4];
int i, j, k = 0;
for(i=0; i<4; i++)
for(j=0; j<i+1; j++) {
twoD[i][j] = k;
k++;
}
for(i=0; i<4; i++) {
for(j=0; j<i+1; j++)
System.out.print(twoD[i][j] + " ");
System.out.println();
}
}
}
```

JAVA STRING

String, is not a simple type. Nor is it simply an array of characters. Rather, String defines an object. The String type is used to declare string variables. You can also declare arrays of strings. String constant can be assigned to a String variable. A variable of type String can be assigned to another variable of type String. You can use an object of type String as an argument to println( ). For example, consider the following fragment:

```
String str = "this is a test";
System.out.println(str);
```

LOCAL VARIABLE TYPE INFERENCE

Type inference in Java refers to the automatic detection of the variable's datatype. This automatic detection usually happens at compile time. It is a feature of Java 10 and it allows the developer to skip declaring the type that is associated with the local variables. Local variables are those which are defined inside a method, initialization block, in for-loops, etc. The type would be usually identified by JDK.
For Example:

```
var message = "Hello, World!";      // String
var bullet = 'C';                   // Character
var dozen = 12;                     // int
var kb = 1_024L;                    // long
var tax = 0.19;                     // double
var reducedTax = 0.07d;             // double
var threeQuarter = 0.75f            // float
```

The literals byte and short don't have special indicators, so they always will be inferred to int instead. The var can be used to declare a variable only when that variable is initialized. For example:

```
var counter; // Wrong! Initializer required
```

However, when used in most other places, var is simply a user-defined identifier with no special meaning. For example, the following declaration is still valid:

```
int var = 1; // In this case, var is simply a user-defined variable.
```

You can also use var to declare an array. For example:

```
var myArray = new int[10]; // This is valid.
```

But this syntax is not allowed:

```
var[] myArray = new int[10]; // Wrong
var myArray[] = new int[10]; // Wrong
var myArray = { 1, 2, 3 }; // Wrong
```

JAVA USER INPUT

The Scanner class is used to get user input, and it is found in the java.util package. To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

PROGRAM 8: Taking input from user

```
import java.util.Scanner;  // Import the Scanner class

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
     // Create a Scanner object
    System.out.println("Enter username");

    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);
     // Output user input
  }
}
```

INPUT TYPES

In the example above, we used the nextLine() method, which is used to read Strings. To read other types, look at the table below:

Method	Description	Method	Description
nextBoolean()	Reads a boolean value from the user	nextInt()	Reads a int value from the user
nextByte()	Reads a byte value from the user	nextLine()	Reads a String value from the user
nextDouble()	Reads a double value from the user	nextLong()	Reads a long value from the user
nextFloat()	Reads a float value from the user	nextShort()	Reads a short value from the user

PROGRAM 9: Another example or taking input from user

```
import java.util.Scanner;

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);

    System.out.println("Enter name, age and salary");

    // String input
    String name = myObj.nextLine();

    // Numerical input
    int age = myObj.nextInt();
    double salary = myObj.nextDouble();

    // Output input by user
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Salary: " + salary);
  }
}
```

PROGRAM 10: Taking input from user in an array

```
import java.util.Scanner;
class Array {
public static void main(String args[]) {
Scanner myObj = new Scanner(System.in);
int marks[] = new int[10];
for (int x=0 ; x<marks.length ; x++) {
System.out.print("Enter Student's Marks:");
marks[x] = myObj.nextInt();
}
System.out.println();
System.out.println("Your Student Data:");
for (int x=0 ; x<marks.length ; x++)
System.out.println("Student No. " +  (x+1) + " Marks:" + marks[x]);
}
}
```

ARITHMETIC OPERATORS

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Addition: +	Subtraction: -	Multiplication: *	Division: /	Modulus: %

The + and - operator also has a unary form that negates its single operand

INCREMENT AND DECREMENT OPERATORS

The increment operator ++ adds 1 to its operand, and the decrement operator -- subtracts 1 from its operand.

`x = x+1;` is the same as `x++;`
And
`x = x-1;` is the same as `x--;`

Both the increment and decrement operators can either precede (prefix) or follow (postfix) the operand.

For example:

`x = x+1;` can be written as
`++x; // prefix form`

or as

`x++; // postfix form`

When an increment or decrement is used as part of an expression, there is an important difference in prefix and postfix forms. If you are using prefix form then increment or decrement will be done before rest of the expression, and if you are using postfix form, then increment or decrement will be done after the complete expression is evaluated.

ARITHMETIC ASSIGNMENT OPERATORS:

+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A
***=**	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A

PROGRAM 11: Demonstrating the basic arithmetic operators.

```
class BasicMath {
public static void main(String args[]) {

// arithmetic using integers
System.out.println("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);

// arithmetic using doubles
System.out.println("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de = -dd;
System.out.println("da = " + da);
System.out.println("db = " + db);
System.out.println("dc = " + dc);
System.out.println("dd = " + dd);
System.out.println("de = " + de);
}
}
```

PROGRAM 12: Demonstrate the % operator.

```
class Modulus {
public static void main(String args[]) {
int x = 42;
double y = 42.25;
System.out.println("x mod 10 = " + x % 10);
System.out.println("y mod 10 = " + y % 10);
}
}
```

EXERSICE 3-2:
Write down a program which takes the temperature in Celsius °C from the user and displays the output in Fahrenheit °F

$$°F = (°C \times 9/5) + 32$$

PROGRAM 13: Demonstrate several assignment operators.

```
class OpEquals {
public static void main(String args[]) {
int a = 1;
int b = 2;
int c = 3;
a += 5;
b *= 4;
c += a * b;
c %= 6;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
}
}
```

PROGRAM 14: Demonstrate ++ operator.

```
class IncDec {
public static void main(String args[]) {
int a = 1;
int b = 2;
int c;
int d;
c = ++b;
d = a++;
c++;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
}
}
```