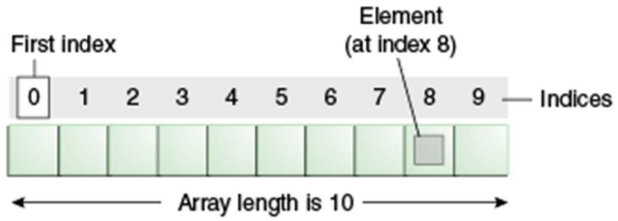
19. Arrays

Normally, an array is a collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on. Unlike C/C++, we can get the length of the array using the length member. In C/C++, we need to use the sizeof operator. In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimentional or multidimentional arrays in Java.

Moreover, Java provides the feature of anonymous arrays which is not available in C/C++.



Advantages:

- **Code Optimization**: It makes the code optimized, we can retrieve or sort the data efficiently.
- **Random access:** We can get any data located at an index position.

Disadvantages:

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

19.1 Types of Arrays

There are three types of arrays.

1. Single Dimensional arrays.
2. Multidimensional arrays.
3. Jagged arrays.

19.1.1 Single Dimensional arrays.

Syntax to Declare and initialize an array in Java

datatype[] variable = {value1, value2, value3,valueN};

(or)

datatype []variable = {value1, value2, value3,valueN};

(or)

datatype variable[] = {value1, value2, value3,valueN};

Declaration and Instantiation of an Array in Java

datatype variable[] = new datatype[size];

(or)

datatype []variable = new datatype[size];

(or)

datatype[] variable = new datatype[size];

Example1:

```
public class ArraysExample1
{
	public static void main(String[] args)
	{
		int a[]= {2,45,35,29};
		// accessing  array elements
		System.out.println(a[1]);
		// over riding array elements
		a[2]=235;
		System.out.println(a[2]);
	}
}
```

Output:

```
45
235
```

Example2:

```
public class ArraysExample2
{
	public static void main(String[] args)
	{
		int a[]= {29,35,235,229};
		// Traversing or printing elements
		for(int i=0; i<4; i++)
		{
			System.out.print(a[i]+" ");
		}
		System.out.println();
		// printing elements using length
		for(int i=0; i<a.length; i++)
		{
			System.out.print(a[i]+" ");
		}
```

```
	}
}
```

Output:

```
29 35 235 229
29 35 235 229
```

For-each loop for Java Array

We can also print the Java array using for-each loop. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop.

Syntax:

```
for(datatype variable : array)
{
//body of loop
}
```

Example3:

```
public class ArraysExample3
{
	public static void main(String[] args)
	{
		int a[]= {29,35,235,229};
		// printing elements using for-each
		for(int i:a)
		{
			System.out.print(i+" ");
		}
	}
}
```

Output:

```
29 35 235 229
```

Example4:

```
public class ArraysExample4
{
	public static void main(String[] args)
	{
		//declaration and instantiation
		int a[]=new int[5];
		a[0]=10;//initialization
		a[1]=20;
		a[2]=70;
		a[3]=40;
		a[4]=50;
```

```
            //traversing array
            //length is the property of array
            for(int i=0;i<a.length;i++)
            {
                  System.out.print(a[i]+" ");
            }
            System.out.println();
            //using for-each
            for(int j:a)
            {
                  System.out.print(j+" ");
            }
      }
}
```

Output:

```
10 20 70 40 50
10 20 70 40 50
```

Example5:

```
import java.util.Scanner;

public class ArraysExample5
{
      public static void main(String[] args)
      {
            //declaration and instantiation
            int a[]=new int[5];
            Scanner sc=new Scanner(System.in);
            for(int i=0;i<a.length;i++)
            {
            System.out.println("enter the value for
                  index "+i+" : ");
                  a[i]=sc.nextInt();
            }
            for(int i=0;i<a.length;i++)
            {
                  System.out.print(a[i]+" ");
            }
            System.out.println();
            //using for-each
            for(int j:a)
            {
                  System.out.print(j+" ");
            }
      }
}
```

Output:

```
enter the value for index 0 :
29
enter the value for index 1 :
35
enter the value for index 2 :
229
enter the value for index 3 :
235
enter the value for index 4 :
2935
29 35 229 235 2935
29 35 229 235 2935
```

19.1.2 Multidimensional Array

In such case, data is stored in row and column based index (also known as matrix form).

Syntax to declare and initialize multidimensional array:

datatype[][] variable = {{value1, value2, valueN}, {value1, value2, valueN},{…},{…}};

(or)

datatype [][]variable = {{value1, value2, valueN}, {value1, value2, valueN},{…},{…}};

(or)

datatype variable[][] = {{value1, value2, valueN}, {value1, value2, valueN},{…},{…}};

Declaration and Instantiation of Multidimensional Array in Java

datatype[][] variable = new int[size][size];

(or)

datatype [][]variable = new int[size][size];

(or)

datatype variable[][] = new int[size][size];

Example1:

```
public class ArraysExample1
{
      public static void main(String[] args)
      {
            int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
            //accessing array elements
            System.out.println(arr[1][2]);
            System.out.println(arr[2][1]);
            System.out.println(arr[0][0]);
            //over riding elements in array
            arr[1][2]=6;
            arr[2][1]=8;
            arr[0][0]=5;
```

```
            //again printing same to check whether
            //elements are changed or not
            System.out.println(arr[1][2]);
            System.out.println(arr[2][1]);
            System.out.println(arr[0][0]);
      }
}
```

Output:

```
5
4
1
6
8
5
```

Example2:

```
public class ArraysExample2
{
      public static void main(String[] args)
      {
            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();
            }
            System.out.println("using length");
            //using length
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                        System.out.print(arr[i][j]+" ");
                  }
                  System.out.println();
            }
            System.out.println("using for-each");
            //using for-each
            for(int j[]:arr)
            {
                  for(int k:j)
                  {
```

```
                    System.out.print(k+" ");
                }
                System.out.println();
            }
        }
}
```

Output:

```
1 2 3
2 4 5
4 4 5
using length
1 2 3
2 4 5
4 4 5
using for-each
1 2 3
2 4 5
4 4 5
```

Example3:

```
import java.util.Scanner;
public class ArraysExample3
{
      public static void main(String[] args)
      {
            int arr[][]=new int[2][3];
            Scanner sc=new Scanner(System.in);
            //for scanning elements
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                  System.out.print("enter value for "+i+
                              " row and "+j+" column : ");
                  arr[i][j]=sc.nextInt();
                  }
            }
            System.out.println("using length");

            //printing elements using length
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                        System.out.print(arr[i][j]+" ");
                }
                System.out.println();
            }
```

```
            System.out.println("using for-each");

            //printing elements using for-each
            for(int j[]:arr)
            {
                  for(int k:j)
                  {
                        System.out.print(k+" ");
                  }
                  System.out.println();
            }
      }
}
```

Output:

```
enter value for 0 row and 0 column : 2
enter value for 0 row and 1 column : 3
enter value for 0 row and 2 column : 5
enter value for 1 row and 0 column : 1
enter value for 1 row and 1 column : 4
enter value for 1 row and 2 column : 3
using length
2 3 5
1 4 3
using for-each
2 3 5
1 4 3
```

19.1.3 Jagged Array

If we are creating odd number of columns in a 2D array, it is known as a jagged array. In other words, it is an array of arrays with different number of columns.

Example1:

```
public class ArraysExample1
{
      public static void main(String[] args)
      {
            int arr[][]={{1,2,3},{9,7,8,4},
               {1,9},{2,3,5,7,6}};
            //printing elements using length
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                        System.out.print(arr[i][j]+" ");
                  }
                  System.out.println();
```

```
            }
            System.out.println("using for-each");

            //printing elements using for-each
            for(int j[]:arr)
            {
                  for(int k:j)
                  {
                        System.out.print(k+" ");
                  }
                  System.out.println();
            }
      }
}
```

Output:

```
1 2 3
9 7 8 4
1 9
2 3 5 7 6
using for-each
1 2 3
9 7 8 4
1 9
2 3 5 7 6
```

Example2:

```
import java.util.Scanner;
public class ArraysExample2
{
      public static void main(String[] args)
      {
            int arr[][]=new int[2][];
            Scanner sc=new Scanner(System.in);
            arr[0]=new int[3];
            arr[1]=new int[5];
            //for scanning elements
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                  System.out.print("enter value for "+i+
                              " row and "+j+" column : ");
                  arr[i][j]=sc.nextInt();
                  }
            }

            //printing elements using length
```

```
            for(int i=0;i<arr.length;i++)
            {
                  for(int j=0;j<arr[i].length;j++)
                  {
                        System.out.print(arr[i][j]+" ");
                  }
                  System.out.println();
            }
            System.out.println("\nusing for-each");

            //printing elements using for-each
            for(int j[]:arr)
            {
                  for(int k:j)
                  {
                        System.out.print(k+" ");
                  }
                  System.out.println();
            }
      }
}
```

Output:

```
enter value for 0 row and 0 column : 2
enter value for 0 row and 1 column : 3
enter value for 0 row and 2 column : 5
enter value for 1 row and 0 column : 1
enter value for 1 row and 1 column : 4
enter value for 1 row and 2 column : 3
enter value for 1 row and 3 column : 2
enter value for 1 row and 4 column : 9
2 3 5
1 4 3 2 9

using for-each
2 3 5
1 4 3 2 9
```

19.2 Array Index Out Of Bunds Exception

The Java Virtual Machine (JVM) throws an ArrayIndexOutOfBoundsException if length of the array in negative, equal to the array size or greater than the array size while traversing the array.

Example:

```
//Java Program to demonstrate the case of
//ArrayIndexOutOfBoundsException in a Java Array.
public class TestArrayException
```

```
{
	public static void main(String args[])
	{
		int arr[]={50,60,70,80};
		for(int i=0;i<=arr.length;i++)
		{
			System.out.println(arr[i]);
		}
	}
}
```

Output:

```
50
60
70
80
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
Index 4 out of bounds for length 4
	at TestArrayException.main(TestArrayException.java:10)
```

19.3 Passing Array to a Method in Java

We can pass the java array to method so that we can reuse the same logic on any array.
Let's see the simple example to get the minimum number of an array using a method.

Example:

```
//Java Program to demonstrate the way of
//passing an array to method.
public class Testarray2
{
	//creating a method which receives an
	//array as a parameter
	static void min(int arr[])
	{
		int min=arr[0];
		for(int i=1;i<arr.length;i++)
		{
			if(min>arr[i])
			{
				min=arr[i];
			}
		}

		System.out.println(min);
	}

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

```
            //declaring and initializing an array
            int a[]={33,3,4,5};
            min(a);//passing array to method
      }
}
```

Output:

```
3
```

19.4 Anonymous Array in Java

Java supports the feature of an anonymous array, so you don't need to declare the array while passing an array to the method.

Example:

```
//Java Program to demonstrate the way of
//passing an anonymous array to method.
public class TestAnonymousArray
{
      //creating a method which receives
      //an array as a parameter
      static void printArray(int arr[])
      {
            for(int i=0;i<arr.length;i++)
            {
            System.out.println(arr[i]);
            }
      }

      public static void main(String args[])
      {
            //passing anonymous array to method
            printArray(new int[]{10,22,44,66});
      }
}
```

Output:

```
10
22
44
66
```

19.5 Returning Array from the Method

We can also return an array from the method in Java.

Example:

```
//Java Program to return an array from the method
public class TestReturnArray
{
      //creating method which returns an array
      static int[] get()
      {
            return new int[]{10,30,50,90,60};
      }

      public static void main(String args[])
      {
      //calling method which returns an array
      int arr[]=get();
      //printing the values of an array
      for(int i=0;i<arr.length;i++)
      {
      System.out.println(arr[i]);
      }
      }
}
```

Output:

```
10
30
50
90
60
```

19.6 Addition of Matrix

Example:

```
import java.util.Scanner;

public class Array
{
      public static void main(String[] args)
      {
            Scanner sc= new Scanner(System.in);
            System.out.println("enter MatArow value");
            int MatArow=sc.nextInt();
            System.out.println("enter MatAcol value");
            int MatAcol=sc.nextInt();
            System.out.println("enter MatBrow value");
            int MatBrow=sc.nextInt();
            System.out.println("enter MatBcol value");
            int MatBcol=sc.nextInt();
```

```
            int MatA[][]=new int[MatArow][MatAcol];
            int MatB[][]=new int[MatBrow][MatBcol];

            if(MatArow==MatBrow && MatAcol==MatBcol)
            {
                  // to scan elements for MatA
                  for(int b=0; b<MatA.length;b++)
                  {
                        for(int c=0; c<MatA[b].length;c++)
                        {
                              System.out.print("enter the
                     value for MatA "+b+" row and "+c+" column : ");
                              MatA[b][c]=sc.nextInt();
                        }
                  }

                  //to scan elements for MatB
                  for(int b=0; b<MatB.length;b++)
                  {
                        for(int c=0; c<MatB[b].length;c++)
                        {
                              System.out.print("enter the value for
                         MatB "+b+" row and "+c+" column : ");
                              MatB[b][c]=sc.nextInt();
                        }
                  }

                  //to perform addition
                  for(int i=0; i<MatA.length;i++)
                  {
                        for(int j=0;j<MatA[i].length;j++)
                        {
                        System.out.print(MatA[i][j]+MatB[i][j]+" ");
                        }
                        System.out.println();
                  }
            }
            else
            {
            System.out.println("Invalid! cannot perform addition");
            }
      }
}
```

Output1:

```
enter MatArow value
2
enter MatAcol value
```

```
2
enter MatBrow value
2
enter MatBcol value
2
enter the value for MatA 0 row and 0 column : 1
enter the value for MatA 0 row and 1 column : 2
enter the value for MatA 1 row and 0 column : 3
enter the value for MatA 1 row and 1 column : 4
enter the value for MatB 0 row and 0 column : 1
enter the value for MatB 0 row and 1 column : 2
enter the value for MatB 1 row and 0 column : 3
enter the value for MatB 1 row and 1 column : 4
2 4
6 8
```

Output2:

```
enter MatArow value
2
enter MatAcol value
3
enter MatBrow value
3
enter MatBcol value
2
Invalid! cannot perform addition
```

19.7 Product of Matrix

Example:

```
import java.util.Scanner;
public class Array
{
	public static void main(String[] args)
	{
		Scanner sc= new Scanner(System.in);
		System.out.println("enter MatArow value");
		int MatArow=sc.nextInt();
		System.out.println("enter MatAcol value");
		int MatAcol=sc.nextInt();
		System.out.println("enter MatBrow value");
		int MatBrow=sc.nextInt();
		System.out.println("enter MatBcol value");
		int MatBcol=sc.nextInt();

		int MatA[][]=new int[MatArow][MatAcol];
		int MatB[][]=new int[MatBrow][MatBcol];

		if(MatArow==MatBcol && MatAcol==MatBrow)
```

```
            {
                  // to scan elements for MatA
                  for(int b=0; b<MatA.length;b++)
                  {
                        for(int c=0; c<MatA[b].length;c++)
                        {
                              System.out.print("enter the value for
                        MatA "+b+" row and "+c+" column : ");
                              MatA[b][c]=sc.nextInt();
                        }
                  }

                  //to scan elements for MatB
                  for(int b=0; b<MatB.length;b++)
                  {
                        for(int c=0; c<MatB[b].length;c++)
                        {
                              System.out.print("enter the value for
                        MatB "+b+" row and "+c+" column : ");
                              MatB[b][c]=sc.nextInt();
                        }
                  }

                  //to perform addition
                  for(int i=0; i<MatA.length;i++)
                  {
                        for(int k=0;k<MatA[i].length;k++)
                        {
                              int sum=0;
                        for(int j=0;j<MatA[i].length;j++)
                        {
                    sum=sum+(MatA[i][j]*MatB[j][k]);
                      }
                              System.out.print(sum+" ");
                        }
                        System.out.println();
                  }
            }
            else
            {
            System.out.println("Invalid! cannot perform addition");
            }
      }
}
```

Output:

```
enter MatArow value
2
enter MatAcol value
```

```
2
enter MatBrow value
2
enter MatBcol value
2
enter the value for MatA 0 row and 0 column : 1
enter the value for MatA 0 row and 1 column : 2
enter the value for MatA 1 row and 0 column : 3
enter the value for MatA 1 row and 1 column : 4
enter the value for MatB 0 row and 0 column : 1
enter the value for MatB 0 row and 1 column : 2
enter the value for MatB 1 row and 0 column : 3
enter the value for MatB 1 row and 1 column : 4
7 10
15 22
```