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.

```
1. //Java Program to demonstrate the way of passing an array
2. //to method.
3. class Testarray2{
4. //creating a method which receives an array as a parameter
5. static void min(int arr[]){
6. int min=arr[0];
7. for(int i=1;i<arr.length;i++)
8.  if(min>arr[i])
9.   min=arr[i];
10.
11. System.out.println(min);
12. }
13.
14. public static void main(String args[]){
15. int a[]={33,3,4,5};//declaring and initializing an array
16. min(a);//passing array to method
17. }}
```

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.

```
1. //Java Program to demonstrate the way of passing an anonymous array
2. //to method.
3. public class TestAnonymousArray{
4. //creating a method which receives an array as a parameter
5. static void printArray(int arr[]){
6. for(int i=0;i<arr.length;i++)
7. System.out.println(arr[i]);
8. }
9.
10. public static void main(String args[]){
11. printArray(new int[]{10,22,44,66});//passing anonymous array to method
```

```
12. }}
```

Returning Array from the Method

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

```
1.  //Java Program to return an array from the method
2.  class TestReturnArray{
3.  //creating method which returns an array
4.  static int[] get(){
5.  return new int[]{10,30,50,90,60};
6.  }
7.
8.  public static void main(String args[]){
9.  //calling method which returns an array
10. int arr[]=get();
11. //printing the values of an array
12. for(int i=0;i<arr.length;i++)
13. System.out.println(arr[i]);
14. }}
```

ArrayIndexOutOfBoundsException

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.

```
1.  //Java Program to demonstrate the case of
2.  //ArrayIndexOutOfBoundsException in a Java Array.
3.  public class TestArrayException{
4.  public static void main(String args[]){
5.  int arr[]={50,60,70,80};
6.  for(int i=0;i<=arr.length;i++){
7.  System.out.println(arr[i]);
8.  }
9.  }}
```

Output:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at TestArrayException.main(TestArrayException.java:5)
50
60
70
80
```

Multidimensional Array in Java

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

Syntax to Declare Multidimensional Array in Java

1. dataType[][] arrayRefVar; (or)
2. dataType [][]arrayRefVar; (or)
3. dataType arrayRefVar[][]; (or)
4. dataType []arrayRefVar[];

Example to instantiate Multidimensional Array in Java

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

Example of Multidimensional Java Array

```
1. /Java Program to illustrate the use of multidimensional array
2. class Testarray3{
3. public static void main(String args[]){
4. //declaring and initializing 2D array
5. int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
6. //printing 2D array
7. for(int i=0;i<3;i++){
8.  for(int j=0;j<3;j++){
9.    System.out.print(arr[i][j]+" ");
10. }
11. System.out.println();
12.}
13.}}
```

Cloning an Array in Java

Since, Java array implements the Cloneable interface, we can create the clone of the Java array. If we create the clone of a single-dimensional array, it creates the deep copy of the Java array. It means, it will copy the actual value. But, if we create the clone of a multidimensional array, it creates the shallow copy of the Java array which means it copies the references.

```
1. //Java Program to clone the array
2. class Testarray1{
3. public static void main(String args[]){
4. int arr[]={33,3,4,5};
5. System.out.println("Printing original array:");
6. for(int i:arr)
7. System.out.println(i);
8. 
9. System.out.println("Printing clone of the array:");
10. int carr[]=arr.clone();
11. for(int i:carr)
12. System.out.println(i);
13. 
14. System.out.println("Are both equal?");
15. System.out.println(arr==carr);
16. 
17. }}
```

Output:

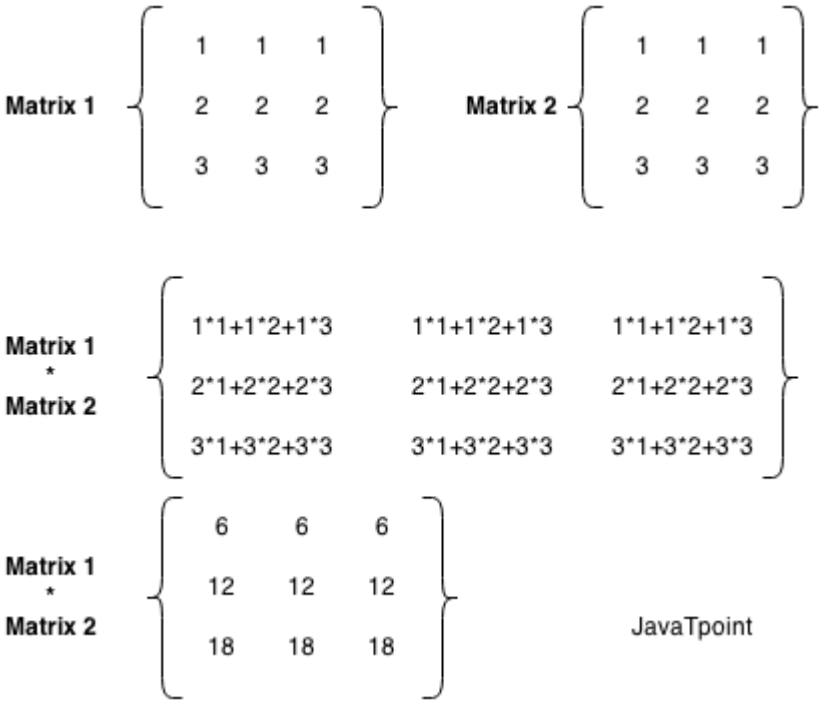
```
Printing original array:
33
3
4
5
Printing clone of the array:
33
3
4
5
Are both equal?
false
```

Addition of 2 Matrices in Java

```
1. /Java Program to demonstrate the addition of two matrices in Java
2. class Testarray5{
3. public static void main(String args[]){
4. //creating two matrices
5. int a[][]={{1,3,4},{3,4,5}};
6. int b[][]={{1,3,4},{3,4,5}};
7.
8. //creating another matrix to store the sum of two matrices
9. int c[][]=new int[2][3];
10.
11. //adding and printing addition of 2 matrices
12. for(int i=0;i<2;i++){
13. for(int j=0;j<3;j++){
14. c[i][j]=a[i][j]+b[i][j];
15. System.out.print(c[i][j]+" ");
16. }
17. System.out.println();//new line
18. }
19.
20. }}
```

Multiplication of 2 Matrices in Java

In the case of matrix multiplication, a one-row element of the first matrix is multiplied by all the columns of the second matrix which can be understood by the image given below.



Let's see a simple example to multiply two matrices of 3 rows and 3 columns.

```
1. //Java Program to multiply two matrices
2. public class MatrixMultiplicationExample{
3. public static void main(String args[]){
4. //creating two matrices
5. int a[][]={{1,1,1},{2,2,2},{3,3,3}};
6. int b[][]={{1,1,1},{2,2,2},{3,3,3}};
7.
8. //creating another matrix to store the multiplication of two matrices
9. int c[][]=new int[3][3];  //3 rows and 3 columns
10.
11. //multiplying and printing multiplication of 2 matrices
12. for(int i=0;i<3;i++){
13. for(int j=0;j<3;j++){
14. c[i][j]=0;
15. for(int k=0;k<3;k++)
16. {
17. c[i][j]+=a[i][k]*b[k][j];
18. }//end of k loop
19. System.out.print(c[i][j]+" ");  //printing matrix element
20. }//end of j loop
```

```
21. System.out.println();//new line
22. }
23. }}
```