EXPERIMENT NO. – 05

AIM: Edit, compile, execute and test a Java program that uses single and multidimensional arrays.

THEORY:

What is an Array?

An array is a very common type of data structure wherein all elements must be of the same data type. Once defined, the size of an array is fixed and cannot increase to accommodate more elements. The first element of an array starts with index zero.

In simple words, it's a programming construct which helps to replace this

```
x0=0;
x1=1;
x2=2;
x3=3;
x4=4;
x5=5;
```

with this …

```
x[0]=0;
x[1]=1;
x[2]=2;
x[3]=3;
x[4]=4;
x[5]=5;
```

In this experiment, you will learn-

how this helps is that a variable can reference the index (the number in the bracket[]) for easy looping.

```
for(count=0; count<5; count++) {
System.out.println(x[count]);
  }
```

Array Variables

Using an array in your program is a **3 step process** -

1) Declaring your Array

2) Constructing your Array**3)** Initialize your Array

1) Declaring your Array

Syntax

```
<elementType>[] <arrayName>;
```

or

```
<elementType><arrayName>[];
```

Example:

```
intintArray[];
 // Defines that intArray is an ARRAY variable which will store integer values
int []intArray;
```

2) Constructing an Array

```
arrayname = new dataType[]
```

Example:

```
intArray = new int[10]; // Defines that intArray will store 10 integer values
```

Declaration and Construction combined

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

3) Initialize an Array

```
intArray[0]=1; // Assigns an integer value 1 to the first element 0 of the array

intArray[1]=2; // Assigns an integer value 2 to the second element 1 of the array
```

Declaring and initialize an Array

```
[]  = {};
```

Example:

```
intintArray[] = {1, 2, 3, 4};
// Initilializes an integer array of length 4 where the first element is 1 , second element is 2 and so on.
```

First Array Program

Step 1) Copy the following code into an editor.

Step 2) Save , Compile & Run the code. Observe the Output

Step 3) If x is a reference to an array, *x.length* will give you the length of the array.

Uncomment line #10. Save, Compile & Run the code.Observe the Output

```
Length of Array  =  7
```

Step 4) Unlike C, Java checks the boundary of an array while accessing an element in it. Java will not allow the programmer to exceed its boundary.

Uncomment line #11. Save, Compile & Run the code.Observe the Output

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 8
atArrayDemo.main(ArrayDemo.java:11)
Command exited with non-zero status 1
```

Step 5) ArrayIndexOutOfBoundsException is thrown. In case of C, the same code would have shown some garbage value.

Java Array: Pass by reference

Arrays are passed to functions by reference, or as a pointer to the original. This means anything you do to the Array inside the function affects the original.

Example: To understand Array are passed by reference

Step 1) Copy the following code into an editor

Step 2) Save, Compile & Run the code. Observe the Output

Multidimensional arrays

Multidimensional arrays are actually arrays of arrays.

To declare a multidimensional array variable, specify each additional index using another set of square brackets.

```
Ex: inttwoD[ ][ ] = new int[4][5] ;
```

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.

In Java, array length of each array in a multidimensional array is under your control.

CONCLUSION: Thus we have successfully studied about editing, compiling, executing and testing a Java program that uses single and multidimensional arrays.

EXPERIMENT NO. – 05

AIM: Edit, compile, execute and test a Java program that uses single and multidimensional arrays.

PROGRAM:

```
class Array Demo{
public static void main(String args[]){
int array[] = new int[7];
for (int count=0;count<7;count++){
array[count]=count+1;
       }
for (int count=0;count<7;count++){
System.out.println("array["+count+"] = "+array[count]);
       }
      //System.out.println("Length of Array  =  "+array.length);
      // array[8] =10;
      }
}
```

Output:

```
array[0] = 1
array[1] = 2
array[2] = 3
array[3] = 4
array[4] = 5
array[5] = 6
array[6] = 7
```

PROGRAM:

```
class Array Demo {
public static void passByReference(String a[]){
a[0] = "Changed";
   }

public static void main(String args[]){
      String []b={"Apple","Mango","Orange"};
System.out.println("Before Function Call    "+b[0]);
ArrayDemo.passByReference(b);
System.out.println("After Function Call    "+b[0]);
   }
}
```

Output:

```
Before Function Call    Apple
After Function Call    Changed
```

PROGRAM:

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

// Create 2-dimensional array.
int[][] twoD = new int[4][4];

  // Assign three elements in it.
twoD[0][0] = 1;
twoD[1][1] = 2;
twoD[3][2] = 3;
System.out.print(twoD[0][0] + " ");
}

}
```

Output:

```
1
```