**ARRAY IN JAVA**

**MODULE - 13**

**1.What is the default value of Array for different data types?**

In Java, when you declare an array, each element of the array is initialized with a default value based on its data type. Here are the default values for different data types:

Numeric Types:

* byte, short, int, long: 0
* float, double: 0.0

Boolean Type:

* boolean: false

Character Type:

* char: '\u0000' (null character)

Reference Types (Objects):

* Reference types such as objects, strings, and arrays are initialized to null.

**2. Can you pass the negative number in Array size?**

No, you cannot pass a negative number as the size of an array in Java. Attempting to create an array with a negative size will result in a runtime exception, specifically NegativeArraySizeException.

Array sizes in Java must be non-negative integers. This restriction ensures that arrays are well-defined and prevents memory-related errors such as buffer overflows or underflows.

For example, the following code would throw a NegativeArraySizeException:

int[] myArray = new int[-5]; // Trying to create an array with a negative size

**3. Where does Array stored in JVM memory?**

In Java, arrays are stored in the Java Virtual Machine (JVM) heap memory. The heap is a region of memory that is shared by all threads of a Java application and is used for the dynamic allocation of objects, including arrays.

When you create an array using the new keyword, memory for the array is allocated on the heap. Each element of the array is stored in contiguous memory locations, and the array itself is represented by a reference, which is also stored in the heap or on the stack (if it's a local variable or a member variable of an object).

**4. What are the disadvantages of Array?**

Arrays have several disadvantages compared to other data structures:

* Fixed Size: Arrays in Java have a fixed size, meaning that you must specify the size of the array when it's created. Once created, the size cannot be changed, which can lead to inefficiency if the size needs to be adjusted dynamically.
* Static Data Structure: Arrays are static data structures, meaning that they cannot grow or shrink in size during runtime. If you need to add or remove elements from an array, you must create a new array with the desired size and copy the elements over, which can be inefficient for large arrays.
* Memory Wastage: Arrays allocate memory for the maximum number of elements they can hold, even if they are not fully utilized. This can lead to memory wastage, especially if the array size is significantly larger than the actual number of elements it holds.
* No Built-in Methods for Manipulation: Arrays in Java do not provide built-in methods for common operations such as adding, removing, or searching for elements. You have to implement these operations manually or use helper methods from the Arrays class, which can be cumbersome and error-prone.
* Index-based Access: Accessing elements in an array is index-based, meaning that you need to know the index of the element you want to access. This can make it difficult to work with unordered or unsorted data.
* Lack of Flexibility: Arrays are not very flexible in terms of data manipulation and organization. Other data structures like lists, sets, and maps offer more flexibility and built-in methods for common operations.

**5. What is an Anonymous Array in Java? Give an example?**

In Java, an anonymous array is an array that is created without explicitly specifying its name. It's typically used in situations where you need to create and initialize an array, but you don't need to refer to it elsewhere in your code. Anonymous arrays are often used as arguments to methods or constructors.

public class AnonymousArrayExample {

public static void main(String[] args) {

// Creating and initializing an anonymous array of integers

printArray(new int[]{1, 2, 3, 4, 5});

}

// Method to print the elements of an array

public static void printArray(int[] arr) {

System.out.print("Array elements: ");

for (int i : arr) {

System.out.print(i + " ");

}

System.out.println();

}

}

**6. What are the different ways to traverse an Array in java?**

In Java, there are several ways to traverse or iterate over an array, depending on the specific requirements of your program. Here are the common methods:

* Using for loop: This is the most common and versatile method to traverse an array.

int[] array = {1, 2, 3, 4, 5};

for (int i = 0; i < array.length; i++) {

System.out.println(array[i]);

}

* Using enhanced for loop (for-each loop): This is a concise way to iterate over elements of an array or any iterable object.

int[] array = {1, 2, 3, 4, 5};

for (int num : array) {

System.out.println(num);

}

* Using Arrays.stream(): This method is available since Java 8 and provides various operations to work with arrays.

int[] array = {1, 2, 3, 4, 5};

Arrays.stream(array).forEach(System.out::println);

**7. What is the difference between length and length() method Give an Examples?**

In Java, length and length() are used to get the size or length of different types of data structures, but they belong to different contexts.

* **length:**

length is an attribute used with arrays to determine the number of elements in the array.

It's a final variable and doesn't have parentheses.

It's applicable only to arrays and not to other data structures.

It's used like this:

**int[] numbers = {1, 2, 3, 4, 5};**

**int arrayLength = numbers.length;**

**System.out.println("Array length: " + arrayLength); // Output: Array length: 5**

* **length():**

length() is a method used with strings, arrays, collections, and other objects to determine their size or length.

It's a method and has parentheses.

It's applicable to objects that implement the CharSequence interface, like strings.

It's used like this:

**String str = "Hello, World!";**

**int strLength = str.length();**

**System.out.println("String length: " + strLength); // Output: String length: 13**