Array Assignment Solution

Question 1: What is the default value of Array for different data types?

Answer 1: If we don't assign values to array elements and try to access them, complier doesn't produce an error as in the case of simple variables. Instead, it assigns values that aren't garbage.

- Boolean → false
- Int → 0
- Double → 0.0
- String → null
- User-defined Type → null.

Question 2: Can you Pass the negative number in Array size?

Answer 2: No, as per the convention of array declaration, it is mandatory that each time an array is evaluated it shall have a size greater than 0. Declaring array size negative breaks this "shall" condition. That's why this action gives an error.

Question 3: Where does Array store in JVM memory?

Answer 3: Memory is allocated in Heap are for the Array in Java. In Java reference types are stored in the Heap area. As arrays are also reference types, (they can be created using the "new" keyword) they are also stored in the Heap area. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. In Java, an array stores primitive values (int, char, etc) or references (i.e. pointers) to objects.

Question 4: What are the disadvantages of Array?

Answer 4: There are some disadvantages of Array: -

- **Fixed size**: Arrays have a fixed size that is determined at the time of creation. This means that if the size of the array needs to be increased, a new array must be created and the data must be copied from the old array to the new array, which can be time-consuming and memory-intensive.
- **Memory allocation issues**: Allocating a large array can be problematic, particularly in systems with limited memory. If the size of the array is too large, the system may run out of memory, which can cause the program to crash.
- **Limited data type support**: Arrays have limited support for complex data types such as objects and structures, as the elements of an array must all be of the same data type.
- **Lack of flexibility**: The fixed size and limited support for complex data types can make arrays inflexible compared to other data structures such as linked lists and trees.

Question 5: What is an Anonymous Array in Java? Give an example?

Answer 5: An array in Java without any name is known as an anonymous array. It is an array just for creating and using instantly. Using an anonymous array, we can pass an array with user values without the referenced variable.

Properties of Anonymous Arrays:

- We can create an array without a name. Such types of nameless arrays are called anonymous arrays.
- The main purpose of an anonymous array is just for instant use (just for one-time usage).
- An anonymous array is passed as an argument of a method.

Note: For Anonymous array creation, do not mention size in []. The number of values passing inside {} will become the size.

- Syntax:

```
new <data type>[]{<list of values with comma separator>};
```

- Examples:

```
// anonymous int array
    new int[] { 1, 2, 3, 4};
// anonymous char array
    new char[] {'x', 'y', 'z'};
// anonymous String array
    new String[] {"Geeks", "for", "Geeks"};
// anonymous multidimensional array
    new int[][] { {10, 20}, {30, 40, 50} };
```

Question 6: What are the different ways to traverse an Array in java?

Answer 6: Using a For Loop: This is the most common method. You iterate over the array using a for loop and access each element by its index.

```
int[] arr = {1, 2, 3, 4, 5};
for (int i = 0; i < arr.length; i++) {
   System.out.println(arr[i]);
```

```
}
```

- **Using an Enhanced For Loop (For-each loop)**: This is a simpler and cleaner way to iterate through arrays and collections introduced in Java 5.

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

for (int num : arr) {

   System.out.println(num);

}
```

- **Using Arrays.stream() with forEach()**: You can convert the array to a stream and then use the forEach() method to perform an action on each element.

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

Arrays.stream(arr).forEach(System.out::println);
```

- **Using Arrays.asList() with forEach()**: If you have an array of objects (not primitives), you can convert it to a List and then use forEach().

```
Integer[] arr = {1, 2, 3, 4, 5};

Arrays.asList(arr).forEach(System.out::println);
```

- **Using Iterator**: You can also use an Iterator to traverse through the array.

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

Iterator<Integer> iterator = Arrays.stream(arr).iterator();

while (iterator.hasNext()) {

   System.out.println(iterator.next());

}.
```

Question 7: What is the difference between length and length() method? Give an Examples?

Answer 7: **array.length**: length is a final variable applicable for arrays. With the help of the length variable, we can obtain the size of the array.

string.length() : length() method is a final method which is applicable for string objects. The length() method returns the number of characters present in the string.

length vs length()

- The length variable is applicable to an array but not for string objects whereas the length() method is applicable for string objects but not for arrays.
- Examples:
  - length can be used for int[], double[], String[]
  - to know the length of the arrays.
  - length() can be used for String, StringBuilder, etc
  - String class related Objects to know the length of the String
- To directly access a field member of an array we can use .length; whereas .length() invokes a method to access a field member.