TELUSKO

9.6-Array of Objects

Length Property of an Array

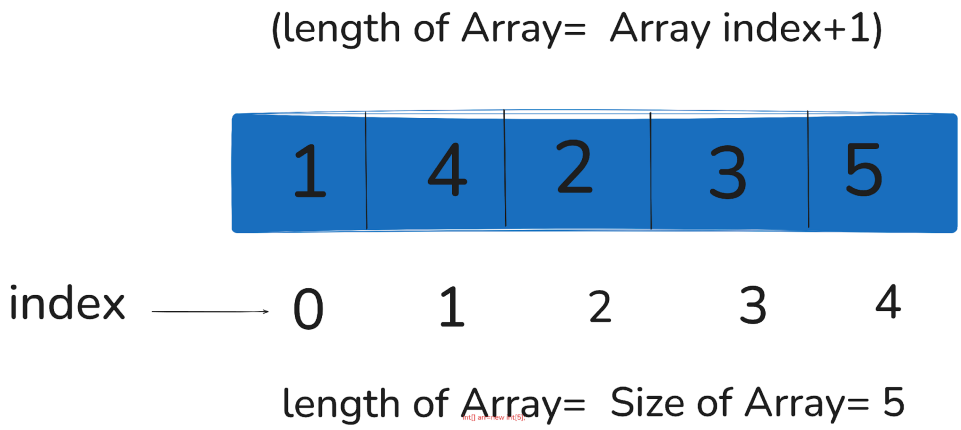
In Java, the length of an array refers to the number of elements it can hold. Unlike other data structures, Java does not provide a predefined method to obtain the length of an array. Instead, the array's length can be accessed using the length property, an attribute built into every array.

- **Understanding length:**
  - The length property gives the total number of elements that an array can contain.
  - This property is accessed using the dot . operator, followed by the array's name.

Example:

```
int[] arr = new int[5];

int arrayLength = arr.length;  // arrayLength will be 5
```

- **Logical Size vs. Array Index:**
  - The logical size usually refers to the number of elements that actually hold meaningful data, not necessarily to the highest index of the array, which is `(length - 1)`.
  - Therefore, if you want to refer to the highest index, it would be **(arrayLength – 1)**.



TELUSKO

Example:

```
int logicalSize = arr.length - 1;  // logicalSize will be 4
```

Code Example:

```
public class MyClass {
   public static void main(String args[]) {
                int[] num = new int[6];
         num[0] = 3;
         num[1] = 4;
         num[2] = 5;
         num[3] = 8;

         for (int i = 0; i < num.length; i++) {
             System.out.print(" " + num[i]);
         }
    }
}
```

Output:

```
3 4 5 8 0 0
```

- **Explanation:**
  - Only the first four elements of the array are assigned values.
  - The remaining two elements are automatically assigned the default value for int, which is 0.
- **Caution:**
  - When iterating through an array, it's crucial to loop only within the bounds of the array's length. Exceeding the array's length will result in an

ArrayIndexOutOfBoundsException, meaning you are trying to access an element beyond the array's limits.

- Using the length property helps ensure that your loop iterates safely within the array's bounds.

Array of Objects

Java is an object-oriented programming language, meaning that classes and objects are fundamental concepts. Typically, when we need to store a single object, we use a variable of the object's type. However, when dealing with multiple objects, an array of objects becomes more practical.

- **Definition:**
  - An array of objects stores objects as its elements, as opposed to traditional arrays, which store primitive data types like int, String, or boolean.
  - The array is created using the class name, making use of Java's Object class, the root class of all classes.
- **Syntax:**

```
Class_Name[] objectArrayReference;
```

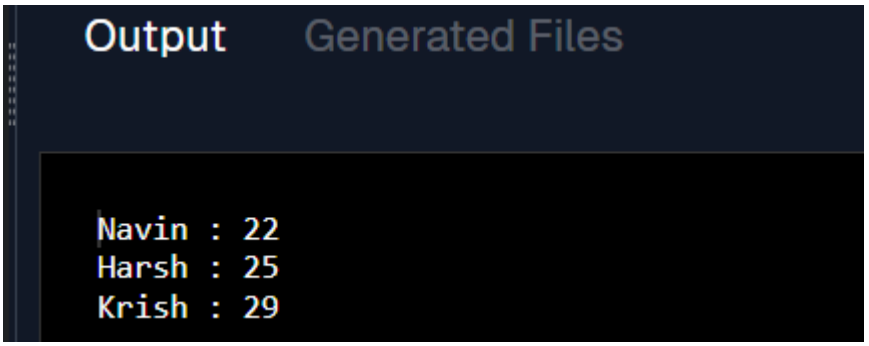
Code Example:

```java
class Student {
    String name;
    int rollNo;
}


public class MyClass {
  public static void main(String args[]) {
                      Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();

        s1.name="Navin";
        s2.name="Harsh";
        s3.name="Krish";

         s1.rollNo=22;
        s2.rollNo=25;
        s3.rollNo=29;

        // Array of Student objects
        Student[] students = new Student[3];
        students[0] = s1;
        students[1] = s2;
        students[2] = s3;

        for (int i = 0; i<students.length; i++) {
            System.out.println(students[i].name + " : " + students[i].rollNo);
        }
    }
}
```

Explanation:

- In this example, students is an array that can hold references to Student objects.
- We then assign references of three Student objects to the array.
- The loop accesses each student's data, such as name and rollNo, via the array.

Output:



Important Note:

The array students holds **references** to Student objects, not the actual objects themselves.

This means the array elements point to where the objects are stored in memory, rather than containing the objects directly.