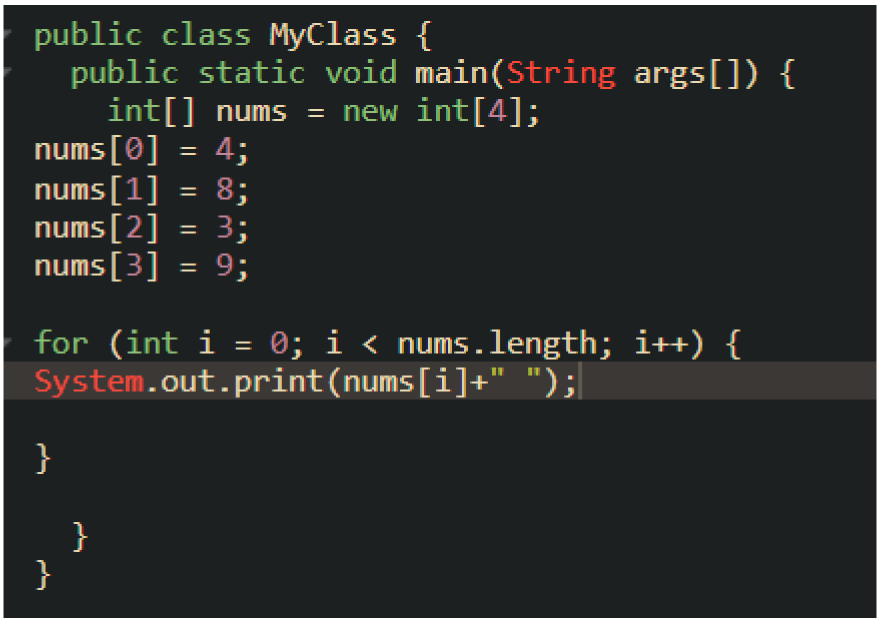
9.7-Enhanced For Loop

- **Introduction:**

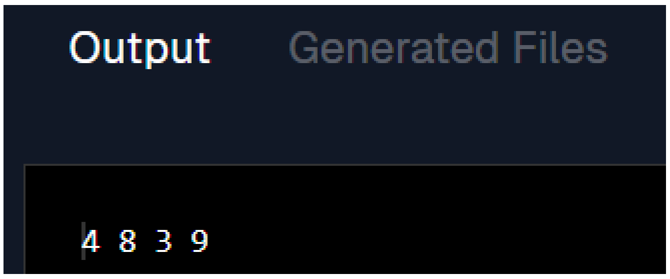
The enhanced for loop, also known as the **for-each loop**, was introduced in Java 1.5. It offers a more concise and readable way to iterate over arrays and collections compared to the traditional for loop. This type of loop is especially useful for traversing elements without needing to manage an index or counter.

- **Traditional For Loop Example**



```
public class MyClass {
  public static void main(String args[]) {
    int[] nums = new int[4];
nums[0] = 4;
nums[1] = 8;
nums[2] = 3;
nums[3] = 9;

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

}

  }
}
```

- **Output:**



```
4 8 3 9
```

- **Explanation:**

In the traditional for loop example above, the loop uses a counter variable (i) to iterate over the array elements.

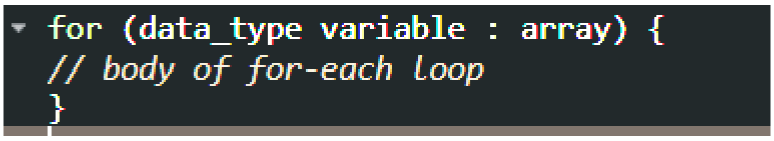
The loop runs from i = 0 to i < nums.length and prints each element.

While this method works fine, it requires managing the index (i) and the array length, which can lead to errors like **ArrayIndexOutOfBoundsException** if the loop exceeds the array's length.

- **Enhanced For Loop**

The enhanced for loop simplifies this process. It is easier to use and reduces the risk of common programming errors.

- **Syntax:**



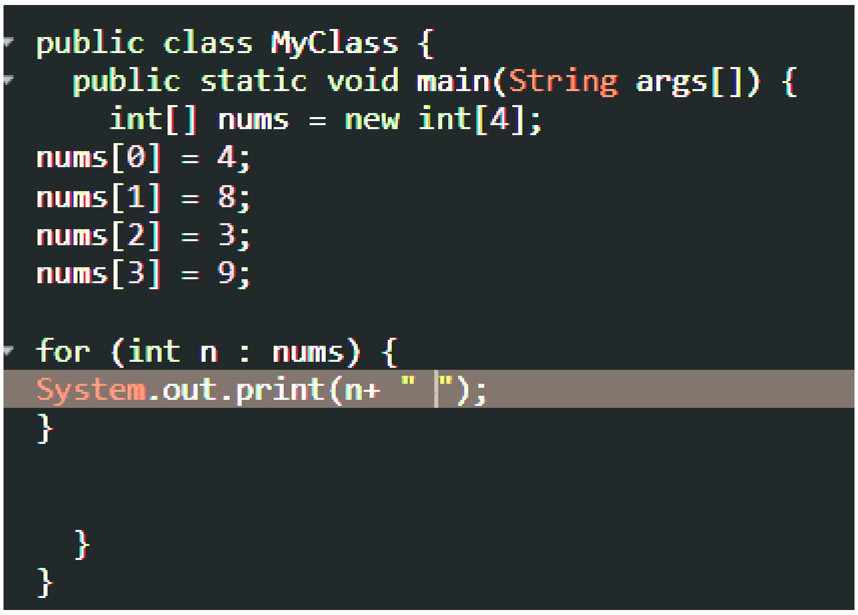
```
for (data_type variable : array) {
// body of for-each loop
}
```

data_type is the type of the elements in the array or collection.

variable is the loop variable that holds the current element.

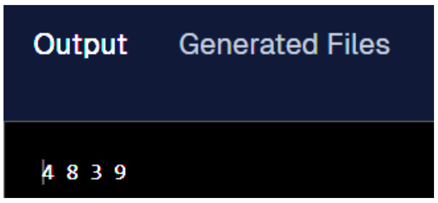
array is the array or collection you want to iterate over.

- **Enhanced For Loop Example**



```
public class MyClass {
  public static void main(String args[]) {
    int[] nums = new int[4];
nums[0] = 4;
nums[1] = 8;
nums[2] = 3;
nums[3] = 9;

for (int n : nums) {
System.out.print(n+ " ");
}


  }
}
```

- **Output:**



```
4 8 3 9
```

- **Explanation:**

In this example, the loop automatically iterates over each element in the nums array.

No need for an index variable or manually managing the loop's range.

The code is cleaner, easier to read, and less prone to errors.

- **Advantages of the Enhanced For Loop**

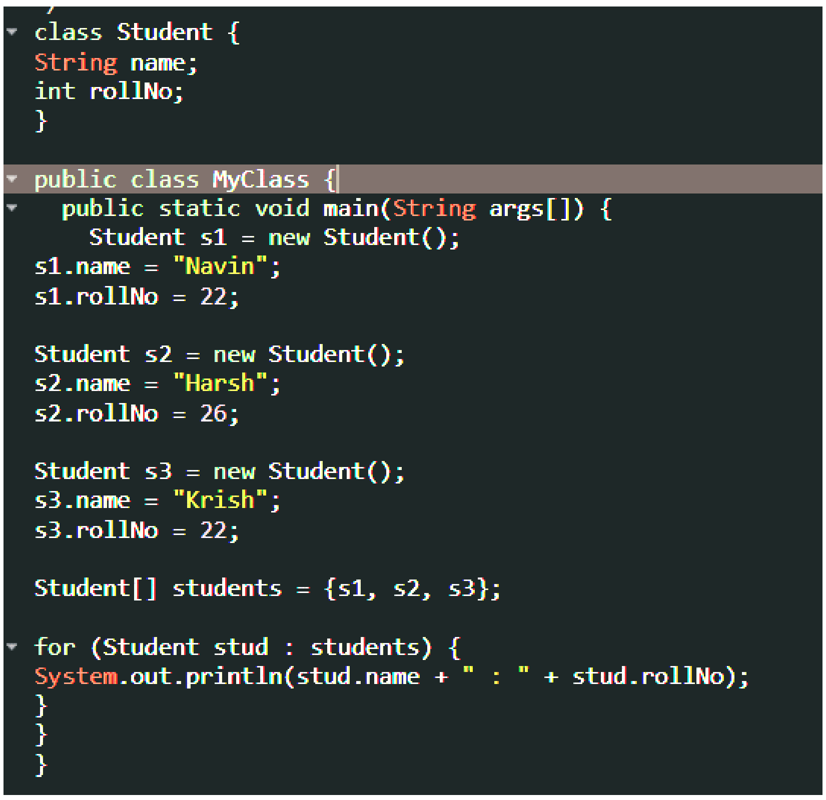
1. **Readability:** The enhanced for loop is more readable and concise.
2. **Error Reduction:** It reduces the chances of errors, such as going out of bounds, because there's no need to manage an index.

- **Limitations of the Enhanced For Loop**

1. **No Reverse Traversal:** The enhanced for loop cannot traverse elements in reverse order.
2. **No Index-Based Access:** You cannot skip elements or access specific elements based on an index within the loop.
3. **Fixed Order:** The loop always iterates from the first element to the last.

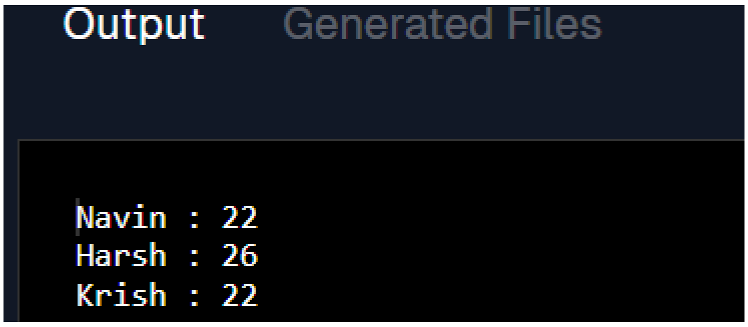
- **Example: Enhanced For Loop with Objects**

The enhanced for loop can also be used to iterate over an array of objects.



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

public class MyClass {
  public static void main(String args[]) {
    Student s1 = new Student();
s1.name = "Navin";
s1.rollNo = 22;

Student s2 = new Student();
s2.name = "Harsh";
s2.rollNo = 26;

Student s3 = new Student();
s3.name = "Krish";
s3.rollNo = 22;

Student[] students = {s1, s2, s3};

for (Student stud : students) {
System.out.println(stud.name + " : " + stud.rollNo);
}
}
}
```

- **Output:**



```
Navin : 22
Harsh : 26
Krish : 22
```

- **Explanation:**

Here, we have an array of Student objects.

The enhanced for loop iterates over each Student object in the students array and prints their names and roll numbers.

This loop is efficient and requires minimal code to achieve the desired result.