Dutch Flag Algorithm Dry Run

```
import java.util.Arrays;

public class SortColors {
    public static void sortArray(int[] arr) {
        int low = 0, mid = 0, high = arr.length - 1;

        while (mid <= high) {
            if (arr[mid] == 0) {
                // Swap arr[mid] and arr[low], move both pointers
                swap(arr, mid, low);
                low++;
                mid++;
            } else if (arr[mid] == 1) {
                // Just move mid pointer
                mid++;
            } else {
                // Swap arr[mid] and arr[high], move high pointer
                swap(arr, mid, high);
                high--;
            }
        }
    }

    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void main(String[] args) {
        int[] arr = {0, 2, 2, 1, 1, 0};
        sortArray(arr);
        System.out.println(Arrays.toString(arr)); // Output: [0, 0, 1, 1, 2, 2]
    }
}. Dry run
```

Let's dry run the **Dutch National Flag Algorithm** step by step with the input:

```
java

int[] arr = {0, 2, 2, 1, 1, 0};
```

We initialize:

- `low = 0`
- `mid = 0`
- `high = arr.length - 1 = 5`

Step-by-step Execution

Step	`low`	`mid`	`high`	Array State	Action
1	0	0	5	**[0**, 2, 2, 1, 1, 0]	Swap `arr[mid]` & `arr[low]` (same), `low++`, `mid++`
2	1	1	5	[0, **2**, 2, 1, 1, 0]	Swap `arr[mid]` & `arr[high]`, `high--`
3	1	1	4	[0, **0**, 2, 1, 1, 2]	Swap `arr[mid]` & `arr[low]`, `low++`, `mid++`
4	2	2	4	[0, 0, **2**, 1, 1, 2]	Swap `arr[mid]` & `arr[high]`, `high--`
5	2	2	3	[0, 0, **1**, 1, 2, 2]	`mid++` since `arr[mid] == 1`
6	2	3	3	[0, 0, 1, **1**, 2, 2]	`mid++` since `arr[mid] == 1`
7	2	4	3	[0, 0, 1, 1, 2, 2]	Loop exits as `mid > high`

Final Output

```java
[0, 0, 1, 1, 2, 2]
```

The array is now sorted correctly in a single pass. 🚀