This document explains 3 different solutions to remove duplicates in an integer array.

1. **DeDup by simple looping**

**Positives:**

* Easy to understand as plain java is used without any references of internal/external libraries.
* Output unique array preserves the original element order.

**Negatives:**

* This solution is not a choice if you are looking at performance as it is time consuming (comparison with 2 for loops), when the original array size is high.

Production level solutions can be implemented using array sorting method or java streams.

**Use case:**

* This solution should be implemented to demonstrate the logic of removing duplicates for academic purpose only.

1. **DeDup with original order**

**Positives:**

* Simple solution and easy to understand. Uses java collections (HashSet/LinkedHashSet). Duplicate filtering logic is handled by these classes.
* Use LinkedHashSet if you need to preserve the order else use HashSet.
* This solution has better time complexity compared to solution 1.

**Negatives:**

* If the application is memory intensive, this can consume more memory as this solution internally uses HashMap/LinkedHashMap.
* This solution is not a choice if you are looking at performance as it is time consuming (comparison with 2 for loops), when the original array size is high.

**Use case:**

* Generate list of client ids who have placed the orders in a given day based on the creation time. Since, each client can place multiple orders, this solution can be implemented by getting the client ids from the complete order list for the day and filtering the duplicates.

1. **DeDup by using java streams**

**Positives:**

* This solution is very simple and can be built with very less lines of code.
* Parallel streams will work very fast.

**Negatives:**

* Since streams are relatively new in Java, their behavior in the production environment cannot be predicted.