Program for Intersection of 2 Arrays

```java
import java.util.HashSet;
import java.util.Set;

public class IntersectionOfArrays {

    public static int[] intersectionOfArray(int[] num1, int[] num2) {
        Set<Integer> hs1 = new HashSet<>();
        Set<Integer> hs2 = new HashSet<>();

        // Add elements of nums1 to set1
        for(int i=0;i<num1.length;i++){
            hs1.add(num1[i]);
        }

        // Check for intersection with elements of nums2
        for(int i=0;i<num2.length;i++){
            if (hs1.contains(num2[i])) {
                hs2.add(num2[i]);
            }
        }

        // Convert the intersection set to an array
        int[] arr = new int[hs2.size()];
        int index = 0;
        for (int num : hs2) {
            arr[index++] = num;
        }
        return arr;
    }

    public static void main(String[] args) {

        int[] nums1 = {1, 2, 2, 1, 7, 9, 3, 4};
        int[] nums2 = {2,3,2 ,5 ,9};
        int[] result = intersectionOfArray(nums1, nums2);

        System.out.print("Intersection: ");
        for (int num : result) {
            System.out.print(num + " ");

        }
    }
}
```

OUTPUT

```
Intersection: 2 3 9
```

Program for Serialize and Deserialize BST

```java
import java.util.*;

class TreeNodeBST {
    int val;
    TreeNodeBST left;
    TreeNodeBST right;

    TreeNodeBST(int val) {
        this.val = val;
    }
}

public class SerializeDeserializeBST {

    // Serialize a BST to a string using preorder traversal
    public String serialize(TreeNodeBST root)
    {
        StringBuilder sb = new StringBuilder();
        serializeHelper(root, sb);
        return sb.toString();
    }

    private void serializeHelper(TreeNodeBST node, StringBuilder sb) {
        if (node == null) {
            sb.append("null").append(",");
            return;
        }

        sb.append(node.val).append(",");
        serializeHelper(node.left, sb);
        serializeHelper(node.right, sb);
    }

    // Deserialize a string to a BST
    public TreeNodeBST deserialize(String data) {
        String[] values = data.split(",");
        Queue<String> queue = new LinkedList<>(Arrays.asList(values));
        return deserializeHelper(queue);
    }

    private TreeNodeBST deserializeHelper(Queue<String> queue) {
        String val = queue.poll();
        if (val.equals("null")) {
            return null;
        }

        TreeNodeBST node = new TreeNodeBST(Integer.parseInt(val));
        node.left = deserializeHelper(queue);
        node.right = deserializeHelper(queue);
        return node;
    }
    public static void main(String[] args) {
        SerializeDeserializeBST sd = new SerializeDeserializeBST();

        // Construct a BST
        TreeNodeBST root = new TreeNodeBST(5);
        root.left = new TreeNodeBST(3);
        root.right = new TreeNodeBST(8);
        root.left.left = new TreeNodeBST(2);
        root.left.right = new TreeNodeBST(4);
        root.right.left = new TreeNodeBST(6);
        root.right.right = new TreeNodeBST(9);

        // Serialize the BST
        String serialized = sd.serialize(root);
        System.out.println("Serialized BST: " + serialized);

        // Deserialize the BST
        TreeNodeBST deserialized = sd.deserialize(serialized);
        System.out.println("Deserialized BST: " + sd.serialize(deserialized));
    }
}
```

OUTPUT

```
Serialized BST: 5,3,2,null,null,4,null,null,8,6,null,null,9,null,null,

Deserialized BST: 5,3,2,null,null,4,null,null,8,6,null,null,9,null,null,
```