Recitation 6

Overview

- Mid Term Prep

Recitation 6

Mid term - Review

Question 1 – Mystery

```
public class Mystery {
	public static int num = 3;
	public static final int NUMBER = 3;
	public static final int[] NUMBERS_ARRAY = { 1, 2, 4 };
	public static void changeNum () {
		num = 5;
	}
	public static void changeNumber () {
		NUMBER = 5;
	}
	public static void changeNumbersI () {
		NUMBERS_ARRAY = { 4, 6, 7 };
	}
	public static void changeNumbersII () {
		NUMBERS_ARRAY = new int[5];
	}
	public static void changeNumbersIII () {
		NUMBERS_ARRAY[0] = 5;
	}
}
```

- Detail for each of the following functions, what will happen if it were to run individually, what the output will be?

Question 1 – Mystery (ANSWER)

```
public class Mystery {
	public static int num = 3;
	public static final int NUMBER = 3;
	public static final int[] NUMBERS_ARRAY = { 1, 2, 4 };
	public static void changeNum () {
		num = 5; // will be run successfully, num will be 5 at the end of the program
	}
	public static void changeNumber () {
		NUMBER = 5; // Error, since we defined 'NUMBER' as final it can't be redeclared
	}
	public static void changeNumbersI () {
		NUMBERS_ARRAY = { 4, 6, 7 }; // Since we can't redefine arrays explicitly
                                             // after declaration
	}
	public static void changeNumbersII () {
		NUMBERS_ARRAY = new int[5]; // Error, since we defined 'NUMBERS_ARRAY'
                                            // as final it can't be redeclared
	}
	public static void changeNumbersIII () {
		NUMBERS_ARRAY[0] = 5; // will be run successfully, NUMBERS_ARRAY
	}               // will be {5, 2, 4} at the end of the program
}
```

- Detail for each of the following functions, what will happen if it were to run individually, what the output will be?

Question 2 – Solution

Variables and Methods:

- **x**: Initialized as an int with value 3.
  It is passed to triple1 as a value (primitive type in Java).

- **y**: Initialized as an array {1, 2, 3}.
  It is passed to triple2 as a reference.

- Function **triple1(int a)**:
  Receives a copy of the value of x (3).
  Modifies the local variable a by setting it to 3 * a (i.e., 9).
  This change affects only a, not x, because x was passed by value.

- Function **triple2(int[] a)**:
  Receives a reference to the array y.
  Attempts to modify the array elements by tripling them, but it uses x (an undeclared variable in triple2) in the loop condition.
  This will cause a compilation error because x is not in the scope of triple2.

This will have a compilation error.

Question 2, Expansion 1 – Mystery (2021 Question 3) – Changed

```
public class Mystery {
      public static int x = 3; // changed
      public static void main (String [] args) {
            int[] y = {1, 2, 3};
            triple1();
            triple2(y);
            System.out.println(x);
            for (int i = 0; i < x; i++){
                  System.out.print(y[i] + " ");
            }
      }

      public static void triple1 () {// changed
            x = 3 * x;           // changed
      }

      public static void triple2 (int[] a) {
            for (int i = 0; i < x; i++){
                  a[i] = 3 * a[i];
            }
      }
}
```

- Write the program's output? Explain.

Question 2, Expansion 1 – Solution

Variables and Methods:

- **x**: Initialized as class variable int with value 3.
  It is passed to triple1 as a value (primitive type in Java).

- **y**: Initialized as an array {1, 2, 3}.
  It is passed to triple2 as a reference.

- Function **triple1(int a)**:
  Receives a copy of the value of x (3).
  Modifies the class variable a by setting it to 3 * x (i.e., 9).

- Function **triple2(int[] a)**:
  Receives a reference to the array y.
  Attempts to modify the array elements by tripling them, but it uses x (which has now value of 9) in the loop condition.
  This will cause a run time error, once it reaches i = 3.

Will and then have a run time error, during run of triple2.

Question 2, Expansion 2 – Mystery (2021 Question 3)

```
public class Mystery {
     public static void main (String [] args) {
          int x = 3;
          int[] y = { 1, 2, 3 };
          triple1(x);
          triple2(y);
          System.out.println(x);
          for (int i = 0; i < x; i++){
               System.out.print(y[i] + " ");
          }
     }

     public static void triple1 (int a) {
          a = 3 * a;
     }

     public static void triple2 (int[] a) {
          for (int i = 0; i < a.length; i++){// changed from original
               a[i] = 3 * a[i];
          }
     }
}
```

- Write the program's output? Explain.

Question 2, Expansion 2 – Solution

Output for Fixed Code:

- triple1(x):
  x remains 3 because triple1 modifies only the local copy of the variable.

- triple2(y, x):
  The array y is modified by tripling each element:
  y[0] = 3 * 1 = 3
  y[1] = 3 * 2 = 6
  y[2] = 3 * 3 = 9

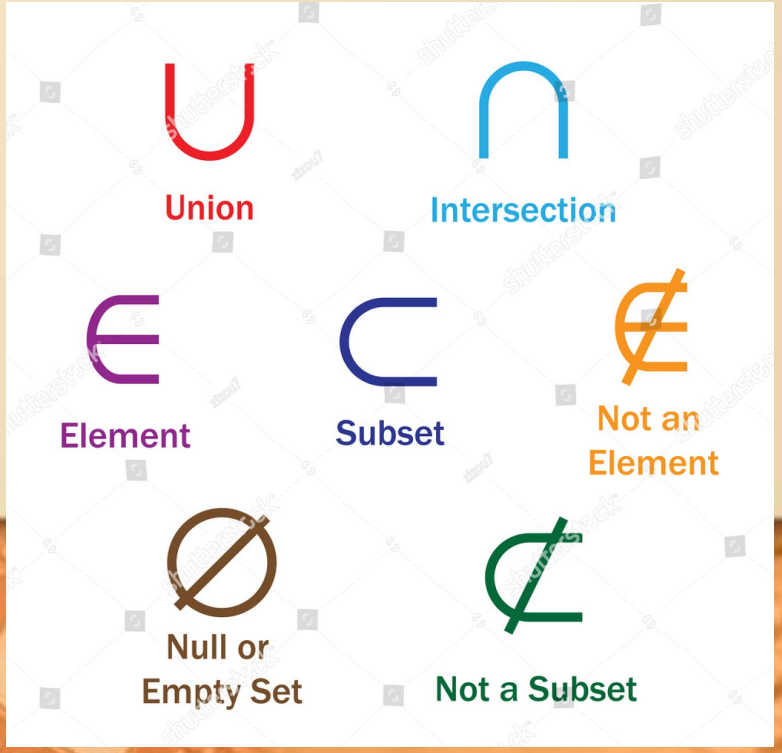
- Output:
  3
  3 6 9

- Explanation:
  - Primitive types (int x) are passed by value. Modifications to the parameter inside the method (a) do not affect the original variable (x).
  - Array references (int[] y) are passed by reference. Modifications to the array elements inside the method affect the original array.

Arrays (int[] y) are passed by reference. Hence modifications to the array elements inside the method affect the original array.

Recitation 6

Sets – Test 1



Todays Story – Sets

- Set is a sequence of numbers (or items) where each element is unique
- Here we will use arrays to hold sets
- However arrays don't have the limitations needed for set hence we must implement the logic.

- Sometimes we will want to use arrays to represent set (from Set Theory), that means we will have to have every element appear at most once.
  - {1,2,3,3} -> invalid as set, valid as array
  - {1,2,3} -> valid
- In the next, few exercises we will create code which
  - Turns array into a setArray (an array with unique values)
  - Implement some set theory methods

Question 3

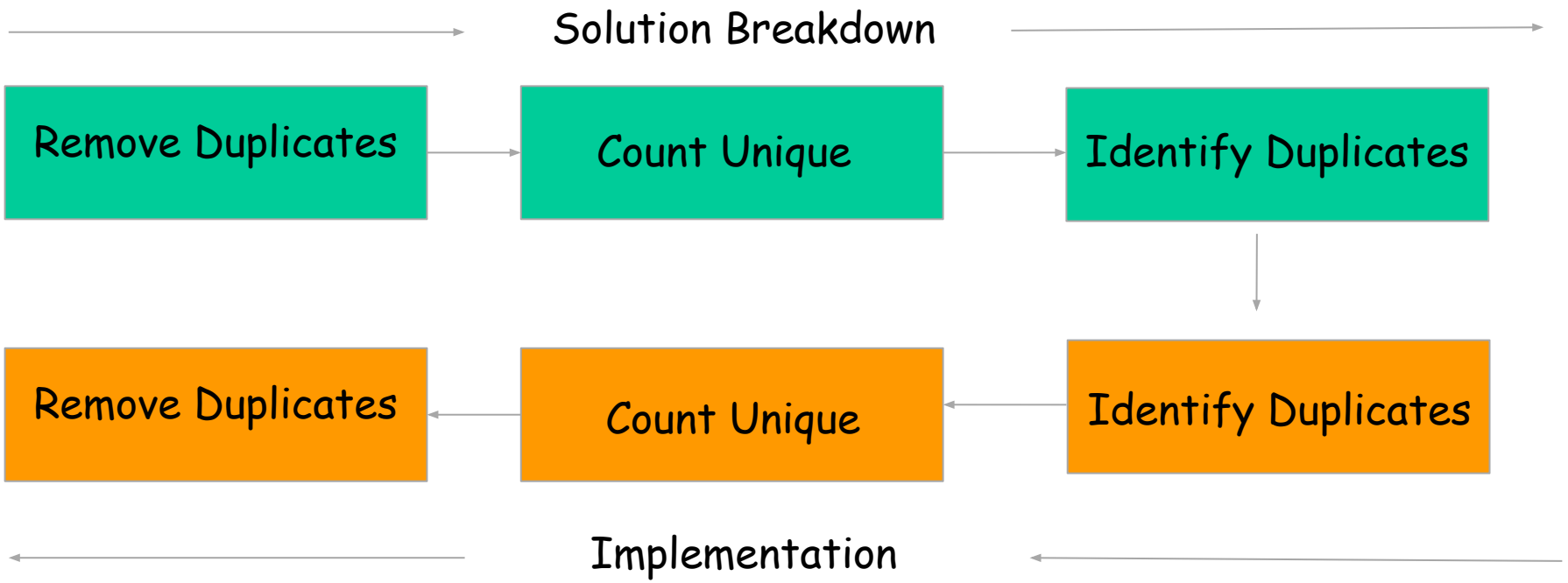
- The function set (int[] array) returns the set containing the unique elements of the array.
- Example:
  - set({1,2,30,30}); // {1,2,30}
  - set({1,2,30,30,2,2,1,3,2,1}); // {1,2,30,3}
  - set({1,2,3,1}); // {1,2,3}
  - set({1,2,2,1,1,4}); // {1,2,4}
- Implement the function

Question 3 - Solution

```
public static int[] set (int[] arr) {
    int [] res = new int[size_of_array_with_no_dups];
    int index = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i]_appears_the_first_time){
            res[index] = arr[i];
            index++;
        }
    }
    return res;
}
```

Solution Breakdown

- Implementing setArray



Implementation Steps:

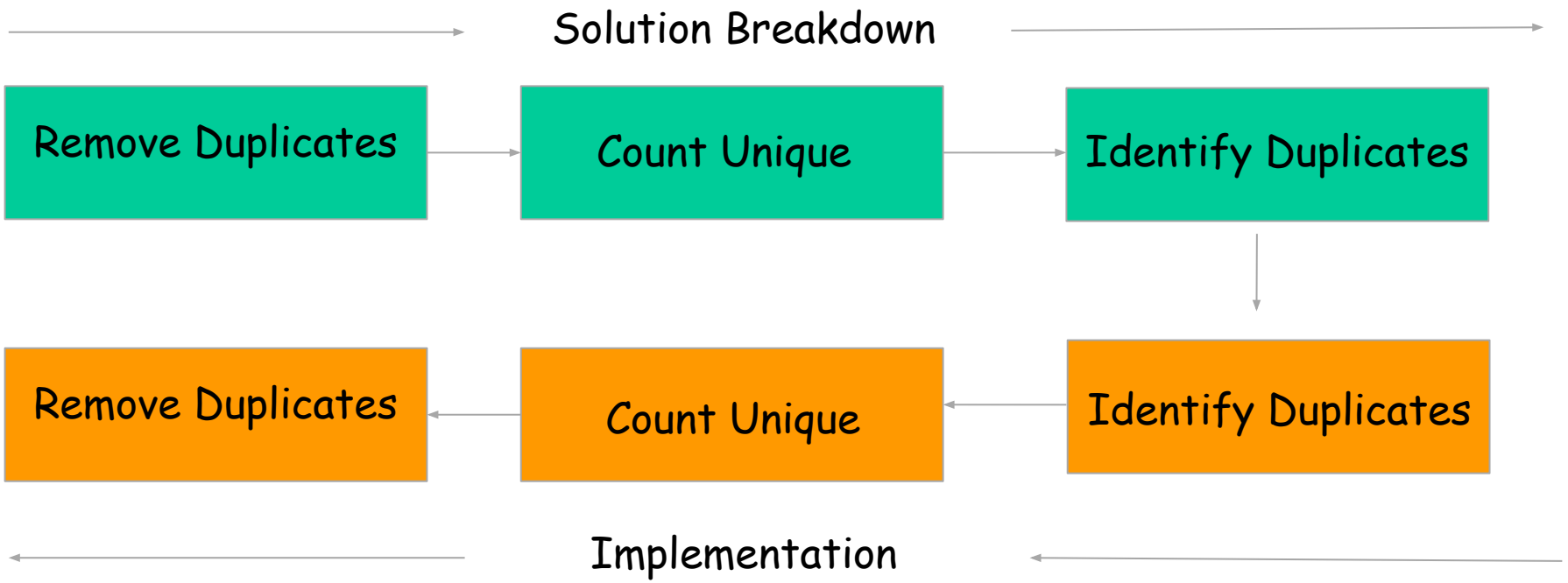
1. Implement function to identify if arr[i] is a duplicate
2. Implement function countUnique()
3. Use (1), (2) to convert an Array to a set

Question 3, Step 1 - SetOps

- The function countUnique (int [] array) returns the number of unique elements in the array.
- Example:
  - countUnique({1,2,30,30}); // 3
  - countUnique({1,2,30,30,2,2,1,3,2,1}); // 4
  - countUnique({1,2,3,1}); // 3
  - countUnique({1,1,1,1}); // 1
  - countUnique({1,2,3,4}); // 4
- Implement the function

Solution Breakdown

- Implementing setArray



Implementation Steps:

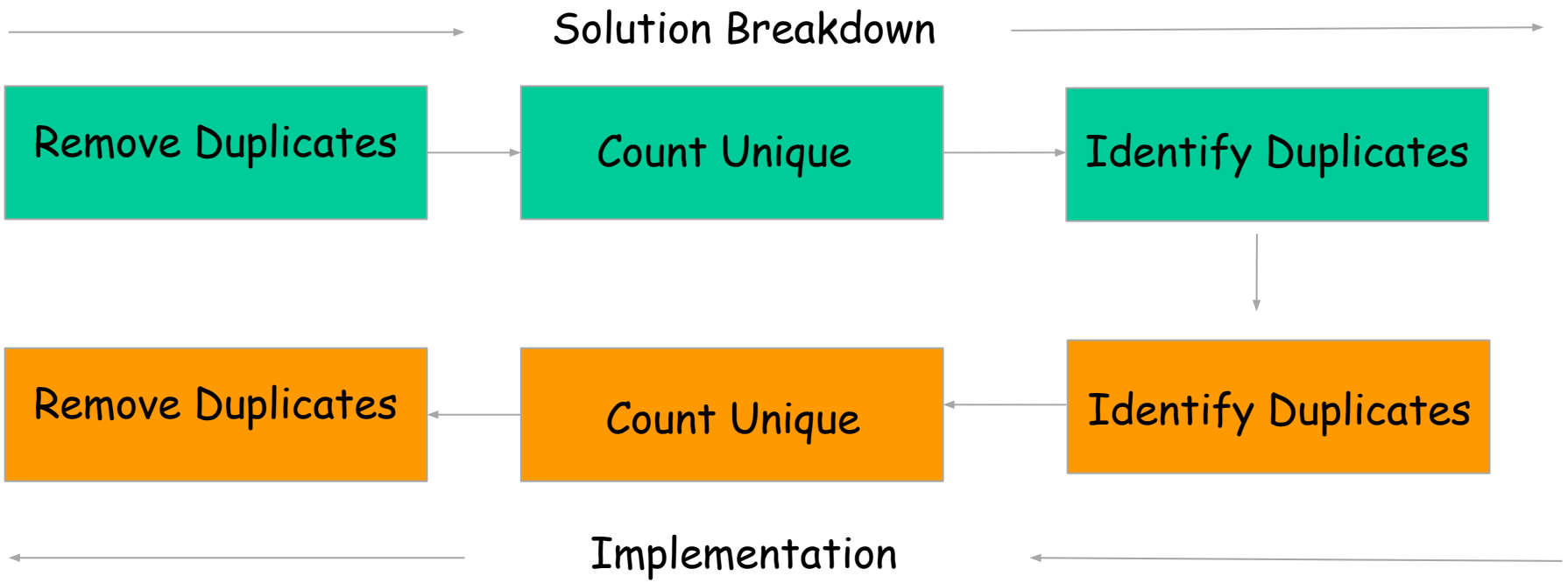
1. Implement function to identify if arr[i] is a duplicate
2. Implement function countUnique()
3. Use (1), (2) to convert an Array to a set

Question 3, Step 1 - Solution

```
public static int countUnique(int[] arr) {
    int count = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i]_appears_the_first_time){
            count++;
        }
    }
    return count;
}
```

Solution Breakdown

- Implementing setArray



Implementation Steps:

1. Implement function to identify if arr[i] is a duplicate
2. Implement function countUnique()
3. Use (1), (2) to convert an Array to a set

Question 3 - Step 2

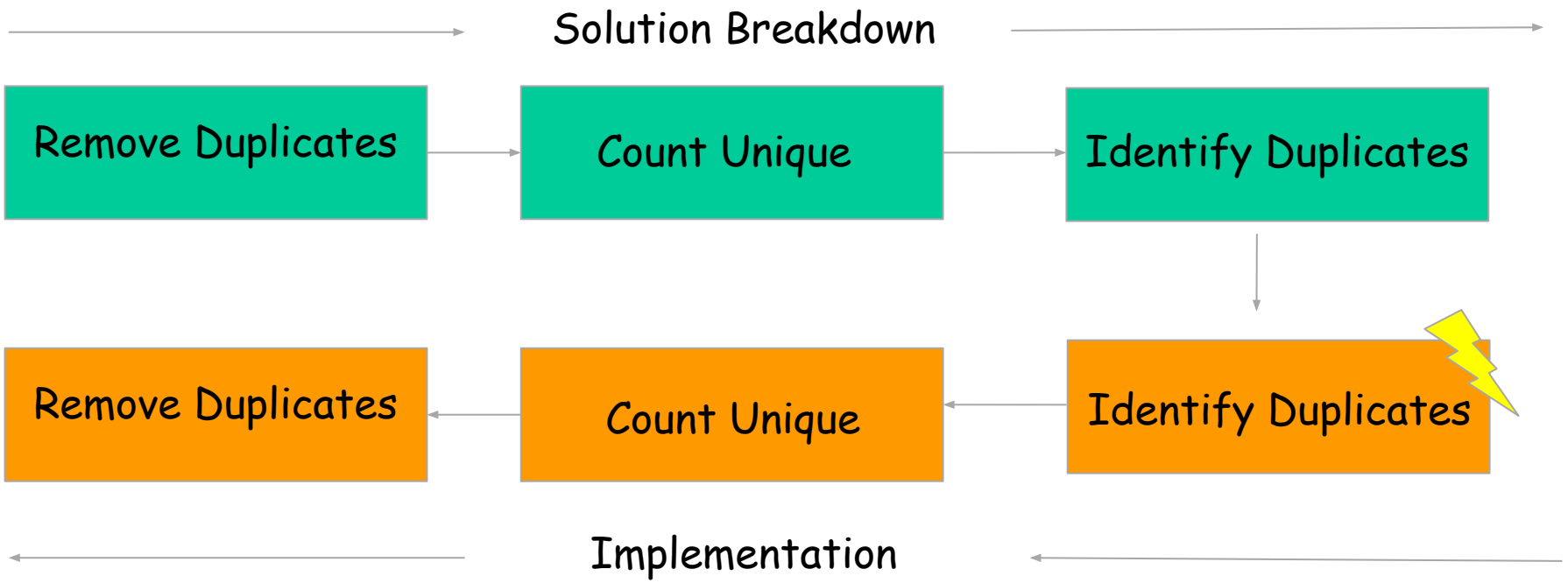
- The function contains (int [] array, int value, int index) return true iff the value appears in the the array before the index (not included).
- Example:
  - contains({1,2,30,30}, 30, 2); // false
  - contains({1,2,30,30}, 30, 3); // true
  - contains({1,2,3,1}, 1, 0); // false
  - contains({1,2,3,1}, 1, 4); // true
  - contains({1,2,3}, 4, 2); // false
- Implement the function

Question 3, Step 2 - Solution

```
public static boolean contains(int[] arr, int value, int index) {
    for (int i = 0; i < index; i++) {
        if (arr[i] == value) {
            return true;
        }
    }
    return false;
}
```

Solution Breakdown

- Implementing setArray



Implementation Steps:

1. Implement function to identify if arr[i] is a duplicate (Done!)
2. Implement function countUnique()
3. Use (1), (2) to convert an Array to a set

Question 3, Step 3 - Solution

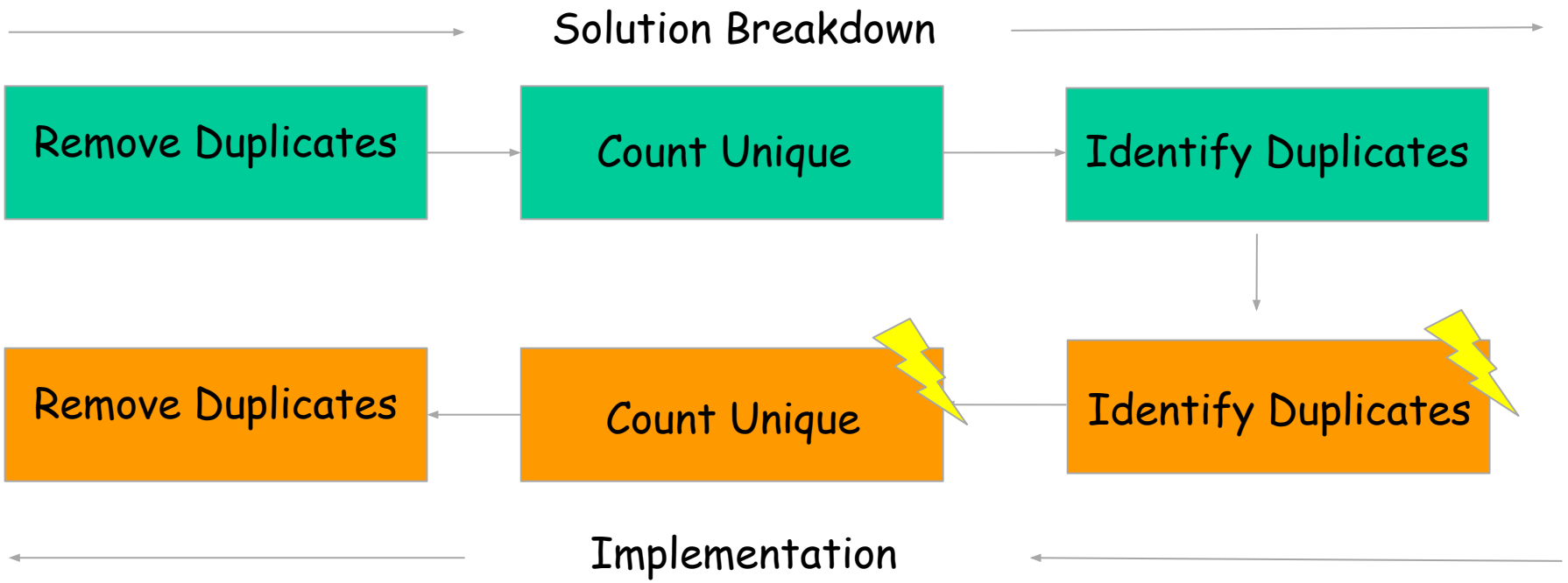
```
public static int countUnique(int[] arr) {
    int count = 0;
    for (int i = 0; i < arr.length; i++) {
        if (!contains(arr, arr[i], i)){
            count++;
        }
    }
    return count;
}
```

Previous Code:

```
public static int countUnique(int[] arr) {
    int count = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i]_appears_the_first_time){
            count++;
        }
    }
    return count;
}
```

Solution Breakdown

- Implementing setArray



Implementation Steps:

1. Implement function to identify if arr[i] is a duplicate (Done!)
2. Implement function countUnique() (Done!)
3. Use (1), (2) to convert an Array to a set

Question 3 - Solution (Before)

```
public static int [] set (int[] arr) {
    int [] res = new int[size_of_array_with_no_dups];
    int index = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i]_appears_the_first_time){
            res[index] = arr[i];
            index++;
        }
    }
    return res;
}
```

Question 3 - Solution (After)

```
public static int [] set (int[] arr) {
    int [] res = new int [countUnique(arr)];
    int index = 0;
    for (int i = 0; i < arr.length; i++) {
        if (!contains(arr, arr[i], i)){
            res[index] = arr[i];
            index++;
        }
    }
    return res;
}
```

Question 3, Expansion 1 - SetOps

- The function unionSets (int [] set1, int [] set2) emulates the union operation.
- The union of set1 and set2 contains elements that appear in either of in set1 or in set2. Note: each element should appear once.
- Example:
  - unionSets({1,7,3}, {4,3,2,7}); // {1,2,3,4,7}
  - unionSets({4,3,2,7}, {4,1}); // {4,3,2,7,1}
  - unionSets({1,2,3}, {3,2,1}); // {1,2,3}
- Implement the function

Question 3, Expansion 1 - Solution

```
public static int [] unionSets(int[] set1, int[] set2) {
    int [] res = new int [set1.length + set2.length];
    int index = 0;
    for (int i = 0; i < set1.length; i++) {
        res[index] = set1[i];
            index++;
    }
    for (int i = 0; i < set2.length; i++) {
        res[index] = set2[i];
            index++;
    }
    return set(res);
}
```

Question 3, Expansion 2 - SetOps

- The function intersectionSets (int [] set1, int [] set2) emulates the intersect operation.
- The intersection of set1 and set2 contains elements that appear in set1 and in set2. Note: each element should appear once. Assume the result will have at least 1 element.
- Example:
  - intersectionSets({1,2,3}, {2,4,8,10,3}); // {2,3}
  - intersectionSets({1,6,11,4,5,8}, {6,2,3,8,11}); // {6,11,8}
  - intersectionSets({6,2,3,8,11}, {1,8,4,5}); // {8}
- Implement the function

Question 3, Expansion 2 - Solution

```
public static boolean contains(int[] arr, int value) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == value) {
            return true;
        }
    }
    return false;
}
```

Question 3, Expansion 2 - Solution

```
public static int [] intersectionSets(int[] set1, int[] set2) {
    int [] arr = new int [Math.min(set1.length, set2.length)];
    int size = 0;

    for (int i = 0; i < set1.length; i++) {
        if (contains(set2, set1[i])){
                arr[size] = set1[i];
                size++;
            }
    }
    int [] res = new int[size];
    for (int i = 0; i < res.length; i++) {
        res[i] = arr[i];
    }
    return res;
}
```

Question 3, Expansion 3 - SetOps

- The function diffSets (int [] set1, int [] set2) emulates the difference operation.
- The difference of set1 - set2, contains elements that appear in set1 doesn't appear in set2. Note: each element should appear once. Assume the result will have at least 1 element
- Example:
  - diffSets({1,2,3}, {2,3}); // {1}
  - diffSets({1,2,3,5}, {2,4,3,6,7}); // {1,5}
  - diffSets({2,4,3,6,7}, {1,2,3,5}); // {4,6,7}
- Implement the function

Question 3, Expansion 3 - Solution

```
public static int [] diffSets(int[] set1, int[] set2) {
    int [] arr = new int [set1.length];
    int size = 0;

    for (int i = 0; i < set1.length; i++) {
        if (!contains(set2, set1[i])){
                arr[size] = set1[i];
                size++;
        }
    }
    int [] res = new int[size];
    for (int i = 0; i < res.length; i++) {
        res[i] = arr[i];
    }
    return res;
}
```

Question 3, Expansion 4 - SetOps

- The function symDiffSets (int [] set1, int [] set2) emulates the simetric difference operation.
- The symmetric difference of set1, set2, contains elements that appear in exactly 1 set. Note: each element should appear once. Assume the result will have at least 1 element
- Example:
  - symDiffSets({1,2,3,5},{2,4,3,6,7}); // {1,5,4,6,7}
  - synDiffSets({2,4,3,6,7},{1,2,3,5}); // {1,5,4,6,7}
- Implement the function

Question 3, Expansion 4 - Solution

```
public static int [] symDiffSets(int[] set1, int[] set2) {
    int [] diff12 = diffSets(set1, set2);
    int [] diff21 = diffSets(set2, set1);
    return unionSets(diff12, diff21);
}
```

Question 2 – Mystery (2021 Question 3)

```
public class Mystery {
     public static void main (String [] args) {
          int x = 3;
          int[] y = { 1, 2, 3 };
          triple1(x);
          triple2(y);
          System.out.println(x);
          for (int i = 0; i < x; i++){
               System.out.print(y[i] + " ");
          }
     }

     public static void triple1 (int a) {
          a = 3 * a;
     }

     public static void triple2 (int[] a) {
          for (int i = 0; i < x; i++){
               a[i] = 3 * a[i];
          }
     }
}
```

- Write the program's output? Explain.

Question 3 – All index of (2022 Question 5)

- In the following question, you need to write the function 'allIndexOf' which receives a String and a character. You need to return all the indices which the given character appears in the given string as an array.
- You may assume that the string is not an empty string, and the given character appears in the string at least once.

```
public static void main(String [] args){
      printArray(allIndexOf("Hello world",'l')); // output: {2,3,9}
      printArray(allIndexOf("Hello worLd",'l')); // output: {2,3}
      printArray(allIndexOf("Hello world",'o')); // output: {4,7}
      printArray(allIndexOf("Hello world",' ')); // output: {5}
      printArray(allIndexOf("Hello world",'d')); // output: {10}
      printArray(allIndexOf("MMMM",'M'));   // output: {0,1,2,3}
}
public static void printArray (int [] array){
      System.out.print("{");
      for (int i = 0; i < array.length ; i++){
            System.out.print(array[i]);
            char c = i != array.length - 1 ? ',' : '}' ;
            System.out.print(c);
      }
      System.out.println();
}
```

Question 4 – Decimal to Hexadecimal (2022 Question 3)

- Hexadecimal count (or hex' for short) is a representation of number with base 16. There are total of 16 digits.
  - The digits 0 to 9, which hold the same value as the decimal presentation.
  - The letters A to F, which hold the values from 10 to 15 (in the decimal representation), respectively. (A is 10, B is 11 etc.)
- Converting a number from a decimal base (base 10) to hex' base is being done in a series of division operations. The first thing you need to do is get the reminder of the number by dividing the number by 16, find the according hex' digit which represents the reminder and write it down, then we will divide the number by 16. The process will repeat itself until we reach 0. The letters should be written from right to left.
- For example, let's take the number 1071 and find its representation in hex':
  - We will divide 1071 by 16 and get 66 with reminder 15. The hex' digit 'F' represents 15, therefore we will write down the hex' digit F. (Current solution: F)
  - We will divide 66 by 16 and get 4 with reminder 2. The hex' digit '2' represents 2, therefore we will write down the hex' digit 2. (Current solution: 2F)
  - We will divide 4 by 16 and get 0 with reminder 4. The hex' digit '4' represents 4, therefore we will write down the hex' digit 4. (Current solution: 42F)
  - We reached 0. Therefore, we finished. 1071 in hex' is 42F.
- Write a function 'decToHex' which receives a non-negative decimal number and returns a String which represents the hex' value of the given number.

Question 4 – Solution

```
public class Hex {
	public static void main (String[] args) {
		System.out.println(decToHex(1071)); // output: “42F”
	}

	pubic static final int HEXA = 16;
	public static String decToHex(int num) {

			if (num == 0){

			return "0";

			}

		int cur = num;

		String hexDigits = "0123456789ABCDEF";

		String hex = "";

		while (cur > 0) {

			int remainder = cur % HEXA;

			hex = hexDigits.charAt(remainder) + hex;

			cur /= 16;

		}

		return hex;

	}

}
```

Question 5 – Run-Length Encoding (2019 Question 3)

- Runlength Encoding is a data compression technique in which runs of 0 and 1 values with repetition are stored by numbers that represent their length. For example, the data set {1,1,1,0,0,0,0,0,1,1,1,1} is encoded as "3,5,4". The method is implemented by the 'compressed' function. The function assumes that the data contains only 0's and 1's and starts with the value 1, there is no need to test it.

Question 5 – Solution

```
public class Compressed {
      public static String compressed(int[] arr) {
          String str = "";
          int currNum = 1;
          int count = 1;
          for (int i = 1; i < arr.length; i++) {
               if (arr[i] == currNum) {
                   count++;
               } else {
              str += count + ",";
              count = 1;
              currNum = 1 - currNum; // cases: currNum = 1 -> 1 - 1 = 0; currNum = 0 -> 1 - 0 = 1
                }
                if (i == arr.length - 1) {
                     str += "" + count;
                }
           }
           return str;
      }
}
```

Question 6 – Contains in string array

- The function containsInArray receives an array of lower case letter only strings, and a String 'val' (also only lowercase), the functions check whether the string 'val' appears in the array.

Question 6 – Solution

```
public class ContainsInStringArray {
	public static boolean containsInArray(String[] arr, String val) {
		for (int i = 0; i < arr.length; i++) {
			if (arr[i].equals(val)){
				return true;
			}
		}
		return false;

	}
}
```