```java
/**
 * This class populates an array with 1000 unique random numbers between
 * 1 and 2000. It also checks uniqueness of the final array.
 *
 * @name    PopulateUnique (Extra Credit Assignment)
 * @author  Ravi S. Ramphal
 * @class   CCSF CS111B
 * @date    2017.07.12
 * @version 1.0
 */

import java.util.*;

public class PopulateUnique
{
    /**
     * This method returns a random integer between the provided lower limit
     * and upper limit.
     *
     * @param  a   An int representing the lower limit
     * @param  b   An int representing the upper limit
     * @return int A random number between the two limits
     */
    private static int rand(int a, int b)
    {
        return ((int)((b - a + 1) * Math.random() + a));
    }

    /**
     * This method sorts the array using quicksort and then compares each
     * element to its next neighbor to return a boolean representing uniqueness.
     *
     * @param  set     An array of integers to check against
     * @return boolean A boolean for if all the elements of the array are unique
     */
    private static boolean isUniq(int[] set)
    {
        Arrays.sort(set); // quicksort

        for (int i = 0; i < set.length - 1; i++)
        {
            if (set[i] == set[i + 1]) return false;
        }
        return true;
    }

    /**
     * This method iterates through an array searching for a target number and
     * returns a boolean if it is/is not found. It performs sequential search,
     * but it only searches over elements that have already been populated
     * (represented by the ‘limit‘ param).
     *
     * @param  set     An array of integers to check against
     * @param  target  The integer that is being searched for
     * @param  limit   The last index that has already been populated
     * @return boolean A boolean for if the number already exists in the set
     */
    private static boolean isRepeated(int[] set, int target, int limit)
    {
        for (int i = 0; i < limit + 1; i++)
        {
            if (target == set[i]) return true;
        }

        return false;
    }

    /**
```

```java
     * This method generates an array of given number of random integers between
     * the lower and upper limits provided and returns the array.
     *
     * @param  count The number of numbers that should be generated
     * @param  lower An int representing the lower limit
     * @param  upper An int representing the upper limit
     * @return int[] An array of integers that were generated
     */
    private static int[] generateUniqueNumbers(int count, int lower, int upper)
    {
        int[] numbers = new int[count];

        for (int i = 0; i < numbers.length; i++)
        {
            int randomNumber = rand(lower, upper);
            while (isRepeated(numbers, randomNumber, i))
            {
                randomNumber = rand(lower, upper);
            }
            numbers[i] = randomNumber;
        }

        return numbers;
    }

    /**
     * This method evaluates a provided array of integers for uniqueness and
     * displays the result to the user.
     *
     * @param numbers An array of integers that should be evaluated
     */
    private static void printEvaluation(int[] numbers)
    {
        System.out.println();
        System.out.print(numbers.length + " ELEMENTS ");
        System.out.println("ARE ALL " +
                           (isUniq(numbers) ? "" : "NOT ") + "UNIQUE:");
        System.out.println();
        System.out.println(Arrays.toString(numbers));
    }

    /**
     * This is the ‘main‘ method of this class. It generates 1000 random numbers
     * between 1 and 2000 and inputs them into an array. Finally, the array is
     * checked for uniqueness and the result is output.
     *
     * @param args An array of arguments provided to the program (unused)
     */
    public static void main(String ... args)
    {
        printEvaluation(generateUniqueNumbers(1000, 1, 2000));
    }
}
```