Forum:Hack Forums

Thread: Why does this produce duplicates?

Tostig - *November 28, 2016, 9:01 pm*

Code import java.util.ArrayList;
import java.util.Random;

```
public class Permutations {

    public static void main(String[] args) {
        ArrayList<Integer> intList = new ArrayList<Integer>();
        intList.add(1);
        intList.add(2);
        intList.add(3);
        intList.add(4);
        intList.add(5);
        intList.add(6);
        intList.add(7);
        intList.add(8);
        intList.add(9);
        intList.add(10);
        permutate();

    }

    public static void permutate() {
        ArrayList<Integer> printList = new ArrayList<Integer>();
        int nextint;
        int pastint = 0;
        Random random = new Random();
        for (int i = 0; i <= 10; i++) {
            nextint = random.nextInt(10 - 1 + 1) + 1;
            if (nextint != pastint)
            {
                printList.add(nextint);
            }
            pastint = nextint;
        }
        System.out.print(printList);

    }

}
```

My output was:
Code [8, 6, 7, 1, 10, 1, 2, 1, 5]
Why does this not produce a result with no duplicates?

Acteda - *November 28, 2016, 9:06 pm*

I think its because you are setting the pastint as the nextint everytime the loop goes through

Meaning the nextint isnt checking for all pastints just the previous one

Right?

Try using a for loop / nested for loop to and keep track of it with an array instead of just one variable (hope this is clear, sorry)

Tostig - *November 28, 2016, 9:11 pm*

(Nov 28, 2016 10:06 PM)Acteda Wrote: [CITE] I think its because you are setting the pastint as the nextint everytime the loop goes through

Meaning the nextint isnt checking for all pastints just the previous one

Right?

Try using a for loop / nested for loop to and keep track of it with an array instead of just one variable (hope this is clear, sorry)
I realize the logic error AFTER posting but fuck me, for some reason using a second arraylist and checking if that arraylist contains any "used" values just does not work. No errors, it just outputs nothing.

clrlslwi - *November 28, 2016, 9:14 pm*

(Nov 28, 2016 10:11 PM)Viktor Wrote: [CITE] I realize the logic error AFTER posting but fuck me, for some reason using a second arraylist and checking if that arraylist contains any "used" values just does not work. No errors, it just outputs nothing.
bruh.

Code
```
if (!printList.contains(nextint))
    printList.add(nextint);
```

Tostig - *November 28, 2016, 9:32 pm*

(Nov 28, 2016 10:14 PM)Demmonic Wrote: [CITE] bruh.

Code
```
if (!printList.contains(nextint))
    printList.add(nextint);
```
sat taking a shit and realized how retarded this thread was

Code
```
public static void permutate() { //void method returns nothing

        int nextint; //new int value to store our random int
        Random random = new Random(); //new random value to generate random ints
        for (int i = 1; i <= 10; i++) { //for each integer 1 to 10 (inclusive)
            ArrayList<Integer> printList = new ArrayList<Integer>(); //make our arraylist to store permutations
            while (printList.size() < 10) { //to ensure the size of the arraylist is 10, make this while loop
                nextint = random.nextInt(10) + 1; //generate our random number
                if (!printList.contains(nextint)) { //if it is not found in printlist then add it
                    printList.add(nextint); //add the int to printlist
                }
```

```
            }
            System.out.println("List " + i + " " +  printList + "Sum of first and last: " + ((int)printList.get(0) + (int)printList.get(9))); //print instructed output
        }
    }
```

ka$h - *November 29, 2016, 2:27 am*

(Nov 28, 2016 10:32 PM)Viktor Wrote: [CITE] sat taking a shit and realized how retarded this thread was

Don't be silly, taking a shit gives you super human thinking powers, along with taking showers. You can't expect yourself to comprehend this when not in super human thinking state unless you already knew it.

Goodies - *November 29, 2016, 10:50 am*

You're just overriding the prevint with the current one. All you're doing is making sure that adjacent numbers cannot be identical.

What you SHOULD be using is a hash table. Something that will have a very fast lookup table instead of O(N) it's O(1).

This will have at least one number and at most 10, but does not allow duplicates.

PHP Code

```
<?php
import java.security.SecureRandom;
import java.util.HashSet;
import java.util.Set;

public class Stuff {
    public static void main(String args[]){
        Set<Integer> set = new HashSet<Integer>();
        SecureRandom random = new SecureRandom();
        for(int i = 0; i < 10; ++i) {
            set.add(random.nextInt(10));
        }
        for(Integer i : set){
            System.out.println(i);
        }
    }
}
```

Stealth™ - *November 29, 2016, 2:09 pm*

(Nov 29, 2016 11:50 AM)Goodies Wrote: [CITE] You're just overriding the prevint with the current one. All you're doing is making sure that adjacent numbers cannot be identical.

What you SHOULD be using is a hash table. Something that will have a very fast lookup table instead of O(N) it's O(1).

This will have at least one number and at most 10, but does not allow duplicates.

PHP Code

```
<?php
import java.security.SecureRandom;
import java.util.HashSet;
```

```
import java.util.Set;

public class Stuff {
   public static void main(String args[]){
      Set<Integer> set = new HashSet<Integer>();
      SecureRandom random = new SecureRandom();
      for(int i = 0; i < 10; ++i) {
         set.add(random.nextInt(10));
      }
      for(Integer i : set){
         System.out.println(i);
      }
   }
}
```

If you're compiling with Java 7 or above, there's no need to define the generic type twice, the right side can be omitted. Also, SecureRandom isn't really needed in this case. If you're looking for efficiency, use ThreadLocalRandom as it's about three times faster than Random.

+Haze - *November 29, 2016, 2:10 pm*

(Nov 28, 2016 10:32 PM)Viktor Wrote: [CITE] sat taking a shit and realized how retarded this thread was

Code public static void permutate() { //void method returns nothing

```
        int nextint; //new int value to store our random int
        Random random = new Random(); //new random value to generate random ints
        for (int i = 1; i <= 10; i++) { //for each integer 1 to 10 (inclusive)
            ArrayList<Integer> printList = new ArrayList<Integer>(); //make our arraylist to store
permutations
            while (printList.size() < 10) { //to ensure the size of the arraylist is 10, make this while loop
                nextint = random.nextInt(10) + 1; //generate our random number
                if (!printList.contains(nextint)) { //if it is not found in printlist then add it
                    printList.add(nextint); //add the int to printlist
                }

            }
            System.out.println("List " + i + " " +  printList + "Sum of first and last: " + ((int)printList.get(0)
+ (int)printList.get(9))); //print instructed output
        }
    }
```

Talk about useless commenting.

@Op
You're doing nothing to prevent duplicates in the first place anyway.

Goodies - *November 29, 2016, 2:41 pm*

(Nov 29, 2016 03:09 PM)Stealth™ Wrote: [CITE] If you're compiling with Java 7 or above, there's no need to define the generic type twice, the right side can be omitted. Also, SecureRandom isn't really needed in this case. If you're looking for efficiency, use ThreadLocalRandom as it's about three times faster than Random.

I actually didn't. My IDE filled it in for me which I thought was strange, but I left it because who cares... (Click to View)

As for the random thing, I'm just sticking with the devil I know. Of course if we were looking for any serious efficiency, I would be using TLR, but there's no need here. I'd be happy with either. Generating 10 numbers is nowhere near enough to actually gain a sufficient improvement.

Tostig - *November 29, 2016, 4:06 pm*

(Nov 29, 2016 03:10 PM)+Haze Wrote: *[CITE]* Talk about useless commenting.

@Op
You're doing nothing to prevent duplicates in the first place anyway.
have to comment for the class even if its redundant

read the thread i fixed it lmao