**(Ping) Nalongsne Daddank**

**private** **int** numberOfItems;

**private** **int**[] items;

The problem is the items array because when we make a clone method for the class like super.clone() in the method. The super.clone() is the Object class only just copy the nuberOfItems’s and the array of items reference in this case. So that means there just make a copy only numberOfItems, there does not make a copy of the array items.

@Override

**protected** BagOfIntegers clone() {

BagOfIntegers copy = **null**;

**try** {

copy = (BagOfIntegers) **super**.clone();

} **catch** (CloneNotSupportedException e) {

e.printStackTrace();

}

**return** copy;

}

We can fix by making a deep copy for array items, using the array’s clone method like items.clone() to the copy of items array. So when we make a clone method of the instance class, the method will copy each element of array to the new array. Then, when we modify the other instance of class, they will not affect each other because they already had own reference of their each available.

@Override

**protected** BagOfIntegers clone() {

BagOfIntegers copy = **null**;

**try** {

copy = (BagOfIntegers) **super**.clone();

} **catch** (CloneNotSupportedException e) {

e.printStackTrace();

}

copy.items = items.clone();

**return** copy;

}