Arrays (review)

```java
int[] A = new int[] {1, 2, 3};
int[] B;
B = A;  // B and A reference the same array
B = A.clone(); // B and A reference two different arrays; the arrays contain the same values
B = Array.copyOf(A, 2); // B and A reference two different arrays; B has first 2 elements of A
```

Collection class – Bag of integers

- A bag can be put in its initial state, which is an empty bag.
- Numbers can be added into the bag.
- You may check how many occurrences of a certain number are in the bag.
- Numbers can be removed from the bag.
- You can check how many numbers are in the bag.

```java
package edu.colorado.collections;
public class IntArrayBag implements Cloneable {
    // Invariant of the IntArrayBag class:
    // 1. The number of elements in the bag is in the instance variable
    //    manyItems, which is no more than data.length.
    // 2. For an empty bag, we do not care what is stored in any of data;
    //    for a non-empty bag, the elements in the bag are stored in data[0]
    //    through data[manyItems-1], and we don’t care what’s in the
    //    rest of data.
    private int[] data;
    private int manyItems;

    public IntArrayBag( ) {
        final int INITIAL_CAPACITY = 10;
        manyItems = 0;
        data = new int[INITIAL_CAPACITY];
    }
    public IntArrayBag(int initialCapacity) {
        if (initialCapacity < 0)
            throw new IllegalArgumentException("The initialCapacity is negative: ", + initialCapacity);
        data = new int[initialCapacity];
        manyItems = 0;
    }
```
public void add(int element) {
    if (manyItems == data.length)
    {
        // Ensure twice as much space as we need.
        ensureCapacity((manyItems + 1)*2);
    }
    data[manyItems] = element;
    manyItems++;
}

public void addMany(int... elements) {
    if (manyItems + elements.length > data.length)
    {
        // Ensure twice as much space as we need.
        ensureCapacity(((manyItems + elements.length)*2));
    }
    System.arraycopy(elements, 0, data, manyItems, elements.length);
    manyItems += elements.length;
}

public void addAll(IntArrayBag addend) {
    // If addend is null, then a NullPointerException is thrown.
    // In the case that the total number of items is beyond
    // Integer.MAX_VALUE, there will be an arithmetic overflow and
    // the bag will fail.
    ensureCapacity(manyItems + addend.manyItems);
    System.arraycopy(addend.data, 0, data, manyItems, addend.manyItems);
    manyItems += addend.manyItems;
}

public IntArrayBag clone() { // Clone an IntArrayBag object.
    IntArrayBag answer;
    try {
        answer = (IntArrayBag) super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // This exception should not occur. But if it does, it would probably
        // indicate a programming error that made super.clone unavailable.
        // The most common error would be forgetting the "Implements Cloneable"
        // clause at the start of this class.
        throw new RuntimeException("This class does not implement Cloneable");
    }
    answer.data = data.clone();
    return answer;
}
public int countOccurrences(int target) {
    int answer;
    int index;
    answer = 0;
    for (index = 0; index < manyItems; index++)
        if (target == data[index])
            answer++;
    return answer;
}

public void ensureCapacity(int minimumCapacity) {
    int[ ] biggerArray;
    if (data.length < minimumCapacity) {
        biggerArray = new int[minimumCapacity];
        System.arraycopy(data, 0, biggerArray, 0, manyItems);
        data = biggerArray;
    }
}

public int getCapacity( ) { return data.length; }

public boolean remove(int target) {
    int index; // The location of target in the data array.
    // First, set index to the location of target in the data array,
    // which could be as small as 0 or as large as manyItems-1; If target
    // is not in the array, then index will be set equal to manyItems;
    for (index = 0; (index < manyItems) && (target != data[index]); index++)
        // No work is needed in the body of this for-loop.
    ;
    if (index == manyItems)
        // The target was not found, so nothing is removed.
        return false;
    else { // The target was found at data[index].
        // So reduce manyItems by 1 and copy the last element onto data[index].
        manyItems--;
        data[index] = data[manyItems];
        return true;
    }
}

public int size( ) { return manyItems; }
public void trimToSize( ) {
    int[ ] trimmedArray;
    if (data.length != manyItems) {
        trimmedArray = new int[manyItems];
        System.arraycopy(data, 0, trimmedArray, 0, manyItems);
        data = trimmedArray;
    }
}

public static IntArrayBag union(IntArrayBag b1, IntArrayBag b2) {
// If either b1 or b2 is null, then a NullPointerException is thrown.
// In the case that the total number of items is beyond
// Integer.MAX_VALUE, there will be an arithmetic overflow and
// the bag will fail.
    IntArrayBag answer = new IntArrayBag(b1.getCapacity( ) + b2.getCapacity( ));
    System.arraycopy(b1.data, 0, answer.data, 0, b1.manyItems);
    System.arraycopy(b2.data, 0, answer.data, b1.manyItems, b2.manyItems);
    answer.manyItems = b1.manyItems + b2.manyItems;
    return answer;
}
}