```java
package Version3;

import java.util.ArrayList;

/**
* This version of the vector class has 66% of it "completed", which hopefully is
* tested and around 66% of the tests will pass
*/
public class Vector {

    private int N = 0;
    private ArrayList<Double> data;

    //empty
    public Vector() {
        N=0;
        data = new ArrayList<Double>();
    }

    //a vector is created of size - size, with elements initalized to D
    public Vector(int size, double D) {
        N = size;
        data = new ArrayList<Double>();
        for(int i=0; i<size; i++) data.add(D);
    }

    //a vector is created to be initialized to the array D
    public Vector(double [] D) {
        int length = D.length;
        N = length;
        data = new ArrayList<Double>();
        for(int i=0; i<length; i++) {
            data.add(D[i]);
        }
    }

    //the vector is initalized to Int I
    public Vector(int [] I) {
        int length = I.length;
        N = length;
        data = new ArrayList<Double>();
        for(int i=0; i<length; i++) {
            data.add( (double)I[i] );
        }
    }

    public void append(double[] doubleArray) {
        int len = doubleArray.length;
        for(int i=0; i<len; i++) {
            data.add(doubleArray[i]);
        }
    }

    public void append(int[] intArray) {
        int len = intArray.length;
        for(int i=0; i<len; i++) {
            data.add((double)intArray[i]);
        }
    }

    public void append(Vector V) {
        int len = V.getLength();
        for(int i=0; i<len; i++) {
            data.add(V.getValue(i));
        }
    }

    public void append(double aDouble) {
        data.add(aDouble);
    }

    //returns if the elements are the same
    public boolean equal(Vector V) {
        int len = V.getLength();
        for(int i=0; i<len; i++) {
            if(data.get(i) != V.getValue(i))return false;
        }
        return true;
    }

    //returns the # of elements
    int getLength() {
        return N;
    }

    //returns the value  this[i]
    double getValue(int i) {
        return data.get(i);
    }

    Vector add(Vector V) {
        throw new UnsupportedOperationException();
    }

    Vector add(double aDouble) {
        throw new UnsupportedOperationException();
    }

    Vector sub(Vector V) {
        throw new UnsupportedOperationException();
    }

    Vector subV(int l, int r) {
        throw new UnsupportedOperationException();
    }

    Vector Mult(Vector V) {
        throw new UnsupportedOperationException();
    }

    Vector Mult(double aDouble) {
        throw new UnsupportedOperationException();
    }

    Vector Normalize() {
        throw new UnsupportedOperationException();
    }

    double EuclidianDistance(Vector V){
        throw new UnsupportedOperationException();
    }
} //end v3
```