

```

1  package Version3;
2
3  import java.util.ArrayList;
4
5  /**
6   * This version of the vector class has 66% of it "completed", which hopefully is
7   * tested and around 66% of the tests will pass
8   */
9  public class Vector {
10
11     private int N = 0;
12     private ArrayList<Double> data;
13
14     //empty
15     public Vector() {
16         N=0;
17         data = new ArrayList<Double>();
18     }
19
20     //a vector is created of size - size, with elements initalized to D
21     public Vector(int size, double D) {
22         N = size;
23         data = new ArrayList<Double>();
24         for(int i=0; i<size; i++) data.add(D);
25     }
26
27     //a vector is created to be initialized to the array D
28     public Vector(double [] D) {
29         int length = D.length;
30         N = length;
31         data = new ArrayList<Double>();
32         for(int i=0; i<length; i++) {
33             data.add(D[i]);
34         }
35     }
36
37     //the vector is initalized to Int I
38     public Vector(int [] I) {
39         int length = I.length;
40         N = length;
41         data = new ArrayList<Double>();
42         for(int i=0; i<length; i++) {
43             data.add( (double)I[i] );
44         }
45     }
46
47     public void append(double[] doubleArray) {
48         int len = doubleArray.length;
49         for(int i=0; i<len; i++) {
50             data.add(doubleArray[i]);
51         }
52     }
53
54     public void append(int[] intArray) {
55         int len = intArray.length;
56         for(int i=0; i<len; i++) {
57             data.add((double)intArray[i]);
58         }
59     }
60
61     public void append(Vector V) {
62         int len = V.getLength();
63         for(int i=0; i<len; i++) {
64             data.add(V.getValue(i));
65         }
66     }
67
68     public void append(double aDouble) {
69         data.add(aDouble);

```

```

70     }
71
72     //returns if the elements are the same
73     public boolean equal(Vector V) {
74         int len = V.getLength();
75         for(int i=0; i<len; i++) {
76             if(data.get(i) != V.getValue(i))return false;
77         }
78         return true;
79     }
80
81     //returns the # of elements
82     int getLength() {
83         return N;
84     }
85
86     //returns the value this[i]
87     double getValue(int i) {
88         return data.get(i);
89     }
90
91     Vector add(Vector V) {
92         throw new UnsupportedOperationException();
93     }
94
95     Vector add(double aDouble) {
96         throw new UnsupportedOperationException();
97     }
98
99     Vector sub(Vector V) {
100         throw new UnsupportedOperationException();
101     }
102
103     Vector subV(int l, int r) {
104         throw new UnsupportedOperationException();
105     }
106
107     Vector Mult(Vector V) {
108         throw new UnsupportedOperationException();
109     }
110
111     Vector Mult(double aDouble) {
112         throw new UnsupportedOperationException();
113     }
114
115     Vector Normalize() {
116         throw new UnsupportedOperationException();
117     }
118
119     double EuclidianDistance(Vector V){
120         throw new UnsupportedOperationException();
121     }
122 } //end v3
123

```