```java
package Version4;

import java.lang.*;

import java.util.ArrayList;

/**
* This version of the vector class has 100% of it "completed", which hopefully is
* tested and all 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]);
        }
    }

    //this will be equivalent to the vector V
    public void append(Vector V) {
        int len = V.getLength();
        for(int i=0; i<len; i++) {
            data.add(V.getValue(i));
        }
    }
```

```
70
71      public void append(double aDouble) {
72          data.add(aDouble);
73      }
74
75      public boolean equal(Vector V) {
76          int len = V.getLength();
77          for(int i=0; i<len; i++) {
78              if(Math.abs(data.get(i) - V.getValue(i)) >0.1)return false;
79          }
80          return true;
81
82      }
83
84      //returns the # of elements
85      int getLength() {
86          return N;
87      }
88
89      //returns the value  this[i]
90      double getValue(int i) {
91          return data.get(i);
92      }
93
94      //add this to V, returning a Vector the same size as this
95      public Vector add(Vector V) {
96          Vector result = new Vector();
97          for(int i=0; i<N; i++) {
98              result.append(data.get(i) + V.getValue(i));
99          }
100         return result;
101     }
102
103     //add aDouble to every element of this
104     public Vector add(double aDouble) {
105         Vector result = new Vector(N, aDouble);
106         for(int i=0; i<N; i++) {
107             result.set(i, result.getValue(i) + this.getValue(i));
108         }
109         return result;
110
111     }
112
113     private void set(int i, double d) {
114         data.set(i, d);
115     }
116
117     //sub this - V
118     public Vector sub(Vector V) {
119         Vector res = new Vector(N, 0);
120         for(int i=0; i<N; i++) {
121             res.set(i, this.getValue(i) - V.getValue(i));
122         }
123         return res;
124     }
125
126     //will return a sub vector between the
127     //indices l and r inclusive
128     public Vector subV(int l, int r) {
129         Vector fin = new Vector();
130         for(int i=l; i<=r; i++) {
131             fin.append(this.getValue(i));
132         }
133         return fin;
134     }
135
136     //Multiple every element of this by corresponding element in V
137     public Vector Mult(Vector V) {
138         Vector fin = new Vector();
```

```
139             for(int i=0; i<N; i++) {
140                 fin.append(this.getValue(i) * V.getValue(i));
141             }
142             return fin;
143         }
144 
145         //Multiply every element of this by aDouble
146         public Vector Mult(double aDouble) {
147             Vector fin = new Vector();
148             for(int i=0; i<N; i++) {
149                 fin.append(this.getValue(i) * aDouble);
150             }
151             return fin;
152         }
153 
154         //returns this as a normalized vector
155         public Vector Normalize() {
156             double div = 0;
157             Vector fin = new Vector();
158             for(int i=0; i<N; i++) {
159                 div += data.get(i) * data.get(i);
160             }
161             div = Math.sqrt(div);
162             for(int i=0; i<N; i++) {
163                 fin.append( data.get(i) / div);
164             }
165             return fin;
166         }
167 
168         //Returns the euclidean distance between this and V
169         double EuclidianDistance(Vector V){
170             double div = 0;
171             for(int i=0; i<N; i++) {
172                 div += Math.pow(V.getValue(i) - this.getValue(i), 2);
173             }
174             div = Math.sqrt(div);
175             return div;
176         }
177 
178     }
179 
```