```java
package Version1; // version obviously depends of what version

import org.junit.Test;

import junit.framework.Assert;

import static org.junit.Assert.assertEquals;
import Version1.Vector; //import the version to test

/**
* This is the test class, where tests for each part of the vector class will be tested
* Our goal is for 100% of these test to pass. The tests attempt to use the
* method, if it fails, we get a message, which takes fractions of a second, when more
* tests are done, it takes longer, but when all tests pass, the test is quick!
*/
public class TestJUnitVector {

    // Attempts creation of a vector
    @Test
    public void testConstructor() {
        try {
            Vector newVector = new Vector();
        }
        catch(Exception e) {
            Assert.fail("Test Failed : " + e.getMessage());
        }
    }

    // attempts to create a  vector of a size "size"
    @Test
    public void testConstructor2() {
        try {
            Vector newVector = new Vector(5, 3);
        }
        catch(Exception e) {
            Assert.fail("Test Failed : " + e.getMessage());
        }
    }

    // tests doubles
    @Test
    public void testConstructor3() {
        try {
            double []arr = {32.5,21.5,35.5};
            Vector newVector = new Vector(arr);
        }
        catch(Exception e) {
            Assert.fail("Test Failed : " + e.getMessage());
        }
    }

    // tests ints
    @Test
    public void testConstructor4() {
        try {
            int [] arr = {32,21,35};
            Vector newVector = new Vector(arr);
        }
        catch(Exception e) {
            Assert.fail("Test Failed : " + e.getMessage());
        }
    }

    //attempts to append a double
    @Test
    public void testAppend1() {
        try {
            Vector newVector = new Vector(4, 2);
            newVector.append(3.4);
```

```
70          }
71          catch(Exception e) {
72              Assert.fail("Test Failed" + e.getMessage());
73          }
74      }
75
76      // attempts to append a an array of ints
77      @Test
78      public void testAppend2() {
79          try {
80              Vector newVector = new Vector();
81              double [] arr = {3.4,1.2,7.8};
82              newVector.append(arr);
83          }
84          catch(Exception e) {
85              Assert.fail("Test Failed : " + e.getMessage());
86          }
87
88      }
89
90      //attempts to append a vector
91      @Test
92      public void testAppend3() {
93          try {
94              Vector newVector = new Vector();
95              int [] arr = {3,1,7};
96              newVector.append(arr);
97          }
98          catch(Exception e) {
99              Assert.fail("Test Failed : " + e.getMessage());
100         }
101     }
102
103     //attempts append a double
104     @Test
105     public void testAppend4() {
106         try {
107             Vector newVector = new Vector();
108             Vector appendVector = new Vector(4, 3.4);
109             newVector.append(appendVector);
110         }
111         catch(Exception e) {
112             Assert.fail("Test Failed : " + e.getMessage());
113         }
114     }
115
116     //tests if the vectors are the same
117     @Test
118     public void testEqual() {
119         try {
120             Vector newVector = new Vector(4, 3.4);
121             Vector secondVector = new Vector(4, 3.4);
122             newVector.equal(secondVector);
123         }
124         catch(Exception e) {
125             Assert.fail("Test Failed : " + e.getMessage());
126         }
127     }
128
129     //checks the length of a vector
130     @Test
131     public void testGetLength() {
132         try {
133             Vector newVector = new Vector(4, 3.4);
134             assertEquals(newVector.getLength(), 4);
135         }
136         catch(Exception e) {
137             Assert.fail("Test Failed : " + e.getMessage());
138         }
```

```
139         }
140
141         // checks if the value you obtain is the same at the position
142         @Test
143         public void testGetValue() {
144             try {
145                 Vector newVector = new Vector(4, 2.4);
146                 assertEquals(newVector.getValue(2), 2.4, 0.01);
147             }
148             catch(Exception e) {
149                 Assert.fail("Test Failed : " + e.getMessage());
150             }
151         }
152
153         // tries to add this to the vector, and it returns the vector of same size
154         @Test
155         public void testAdd1() {
156             try {
157                 Vector newVector = new Vector(4, 2);
158                 Vector secondVector = new Vector(4, 3);
159                 newVector.add(secondVector);
160                 assertEquals(newVector.getValue(2), 5, 0.01);
161             }
162             catch(Exception e) {
163                 Assert.fail("Test Failed : " + e.getMessage());
164             }
165         }
166
167         // attempts to add a doubl to the vector
168         @Test
169         public void testAdd2() {
170             try {
171                 Vector newVector = new Vector(4, 2);
172                 newVector.add(3);
173                 assertEquals(newVector.getValue(2), 5, 0.01);
174             }
175             catch(Exception e) {
176                 Assert.fail("Test Failed : " + e.getMessage());
177             }
178         }
179
180         // tries to subtract an element from the vector
181         @Test
182         public void testSub() {
183             try {
184                 Vector newVector = new Vector(4, 3);
185                 Vector secondVector = new Vector(4, 2);
186                 newVector.sub(secondVector);
187                 assertEquals(newVector.getValue(2), 1, 0.01);
188             }
189             catch(Exception e) {
190                 Assert.fail("Test Failed : " + e.getMessage());
191             }
192         }
193
194         //tries to returned a subtracted vector between these two points
195         @Test
196         public void testSubV() {
197             try {
198                 Vector newVector = new Vector(6, 2.4);
199                 Vector secondVector = new Vector(3, 2.4);
200                 assertEquals(secondVector.equal(newVector.subV(0, 3)), true);
201             }
202             catch(Exception e) {
203                 Assert.fail("Test Failed : " + e.getMessage());
204             }
205         }
206
207         // tests if every element is multiplied by its corresponding element in first vector
```

```
208         @Test
209         public void testMult() {
210             try {
211                 Vector newVector = new Vector(4, 2.4);
212                 Vector secondVector = new Vector(4, 2);
213                 Vector finalVec = newVector.Mult(secondVector);
214                 assertEquals(finalVec.getValue(2), 4.8, 0.01);
215             }
216             catch(Exception e) {
217                 Assert.fail("Test Failed : " + e.getMessage());
218             }
219         }
220 
221         // tests if every element is multiplied by a double
222         @Test
223         public void testMult2() {
224             try {
225                 Vector newVector = new Vector(4, 2.4);
226                 newVector.Mult(2);
227                 assertEquals(newVector.getValue(2), 4.8, 0.01);
228             }
229             catch(Exception e) {
230                 Assert.fail("Test Failed : " + e.getMessage());
231             }
232         }
233 
234         // tests if a normalized vector is returned
235         @Test
236         public void testNormalization() {
237             try {
238                 Vector newVector = new Vector(3, 2);
239                 Vector normalizedVector = newVector.Normalize();
240                 assertEquals(normalizedVector.getValue(2), 0.57, 0.1);
241             }
242             catch(Exception e) {
243                 Assert.fail("Test Failed : " + e.getMessage());
244             }
245         }
246 
247         // test to see distance between two vectors
248         @Test
249         public void testEudDistance() {
250             try {
251                 Vector newVector = new Vector(2, 2);
252                 Vector secondVector = new Vector(2, 3);
253                 assertEquals(newVector.EuclidianDistance(secondVector), 1.41, 0.1);
254             }
255             catch(Exception e) {
256                 Assert.fail("Test Failed : " + e.getMessage());
257             }
258         }
259 
260     } //end test
261 
```