```java
// EXAM #1 Sample Solution for CylindricalTank
// Instructions
// ****************************************************************
/*Complete the definition of the following MUTABLE class named CylindricalTank.  Each
instance of the class (i.e. each object) will represent a cylindrically shaped liquid
tank.

Your job consist of completing the following tasks:

Complete the definition of the CylindricalTank copy constructor
Complete the definition of the getLiquidVolumeInGallons() instance method
Complete the definition of compareTo instance method
Complete the definitions of the add instance method
Complete the definitions of the getPercentFilled instance method
*/
// Sample Solution
// ****************************************************************

public class CylindricalTank {

    private String idTag;           // Unique String identifying the tank
    private double radius;          // The non-negative radius of the base of the tank
    in meters
    private double height;          // The non-negative height of the tank in meters
    private double liquidLevel;     // The current height of the liquid in the tank in
    meters

    /**
     * CylindricalTank General Constructor
     */
    public CylindricalTank(String tag, double radius, double height, double liquidLevel)
     {
        super();
        this.idTag = tag;
        this.radius = radius;
        this.height = height;
        this.liquidLevel = liquidLevel;
    }

    /**
     * Exercise #1
     * Copy Constructor creates a tank with the same properties as the parameter tank.
     */
    public CylindricalTank(CylindricalTank t) {
        // YOUR CODE HERE
        this.idTag = t.idTag;
        this.radius = t.radius;
        this.height = t.height;
        this.liquidLevel = t.liquidLevel;
    }

    // Getters
    public String getIdTag() { return idTag; }
    public double getRadius() { return radius; }
    public double getHeight() { return height; }
    public double getLiquidLevel() { return liquidLevel; }

    // Setters
    public void setRadius(double radius) { this.radius = radius; }
    public void setHeight(double height) { this.height = height; }
    public void setLiquidLevel(double liquidLevel) { this.liquidLevel = liquidLevel; }

    // Instance Methods

    /**
     * Returns true if both the target and parameter tanks have the same id
```

```
62          */
63         public boolean equals(Object t2) {
64             if (t2 instanceof CylindricalTank) {
65                 CylindricalTank ct = (CylindricalTank) t2;
66                 return this.getIdTag().equals(ct.getIdTag());
67             }
68             return false;
69         }
70 
71         /**
72          * Returns the Cylindrical Tank as a string.
73          */
74         public String toString() {
75             return "CylindricalTank[id=" + this.getIdTag() + "]";
76         }
77 
78         /**
79          * Returns the maximum volume of liquid that the tank can hold in cubic meters.
80          */
81         public double getCapacity() {
82             return Math.PI * this.radius * this.radius * this.height;
83         }
84 
85         /**
86          * Returns the current volume of liquid that the tank holds in cubic meters.
87          */
88         public double getLiquidVolume() {
89             return Math.PI * this.radius * this.radius * this.liquidLevel;
90         }
91 
92         /**
93          * Exercise #2
94          * Returns the current volume of liquid that the tank can hold in gallons.
95          *
96          * Hint: 1 Cubic Meter is equivalent to 264 US Gallons.
97          */
98         public double getLiquidVolumeInGallons() {
99             return getLiquidVolume()*264;
100        }
101 
102        /**
103         * Exercise #3
104         * Compares the capacity (volume) of the target tank and the parameter tank.
105         * Returns 0 if they have the same capacity, 1 if the target tank has larger capacity
106         * and -1 otherwise.
107         */
108        public int compareTo(CylindricalTank t) {
109            // YOUR CODE HERE
110            if (this.getCapacity() == t.getCapacity()) {
111                return 0;
112            }
113            else if (this.getCapacity() > t.getCapacity()) {
114                return 1;
115            }
116            return -1; // Dummy return
117        }
118 
119        /**
120         * Exercise #4
121         * Modifies the target tank and add to it as much of the volume of liquid specified
122         * by the cubicMeters parameters as possible.  If the tank cannot hold all the
            liquid then it
123         * should become full and the remainder of the liquid should be simply ignored.
124         *
125         * Note: This method must return the instance object (this).
126         */
```

```
127         public CylindricalTank add(double cubicMeters) {
128             // YOUR CODE HERE
129             double spaceOnTarget = this.getCapacity() - this.getLiquidVolume();
130             double volumeToTransfer = Math.min(spaceOnTarget,cubicMeters);
131             this.liquidLevel += volumeToTransfer / (Math.PI * this.radius * this.radius);
132             return this; // Leave return as is
133         }
134 
135 
136         /**
137          * Exercise #5
138          * Returns a string indicating the percent of its volume that the tank is full
             ignoring
139          * any fractional part. For instance if the tank is exactly half way full the
             method should return the
140          * String "50% Full".  If the tank is exactly one third full the method should
             return the
141          * String "33% Full".  If the tank is empty the method should return the String
             "Empty".
142          * HINT: To convert a double to an int you can cast it as follows (int)5.3 yields 5.
143          */
144         public String getPercentFilled() {
145             // YOUR CODE HERE
146             if (this.liquidLevel == 0) {
147                 return "Empty";
148             }
149             else {
150                 return (int)(100 * liquidLevel / height) + "% Full";
151             }
152             //return ""; // Dummy return
153         }
154 
155     }
156 
157 
158     // Tests
159     // ****************************************************************
160 
161     import static org.junit.Assert.*;
162 
163     import org.junit.Before;
164     import org.junit.Test;
165 
166     public class CylindricalTankTest {
167 
168         private static double epsilon = 1; // Tolerance allowed when comparing doubles
169 
170         private static CylindricalTank emptyTank;
171         private static CylindricalTank bigFullTank;
172         private static CylindricalTank smallFullTank;
173         private static CylindricalTank halfFullTank;
174         private static CylindricalTank lessThanTenLittersLeftTank;
175         private static CylindricalTank containsTwentyLittersTank;
176 
177         //CylindricalTank(String tag, double radius, double height, double liquidLevel)
178 
179         @Before
180         public void setUp() {
181             emptyTank = new CylindricalTank("1", 10.0, 10.0, 0.0);
182             bigFullTank = new CylindricalTank("2", 50.0, 100.0, 100.0);
183             smallFullTank = new CylindricalTank("3", 5.0, 5.0, 5.0);
184             halfFullTank = new CylindricalTank("4", 50.0, 100.0, 50.0);
185             lessThanTenLittersLeftTank = new CylindricalTank("5", 50.0, 100.0, 99.9);
186             containsTwentyLittersTank = new CylindricalTank("6", 50.0, 100.0, 99.9);
187         }
188 
```

```
189         @Test
190         public void testCopyConstructor() {
191             CylindricalTank ct1 = new CylindricalTank(emptyTank);
192             assertEquals(emptyTank, ct1);
193             assertEquals(emptyTank.getHeight(), ct1.getHeight(), epsilon);
194             assertEquals(emptyTank.getRadius(), ct1.getRadius(), epsilon);
195             assertEquals(emptyTank.getLiquidLevel(), ct1.getLiquidLevel(), epsilon);
196 
197             CylindricalTank ct2 = new CylindricalTank(lessThanTenLittersLeftTank);
198             assertEquals(lessThanTenLittersLeftTank, ct2);
199             assertEquals(lessThanTenLittersLeftTank.getHeight(), ct2.getHeight(), epsilon);
200             assertEquals(lessThanTenLittersLeftTank.getRadius(), ct2.getRadius(), epsilon);
201             assertEquals(lessThanTenLittersLeftTank.getLiquidLevel(), ct2.getLiquidLevel(),
                epsilon);
202         }
203 
204         @Test
205         public void testGetLiquidVolumeInGallons() {
206             assertEquals(0, emptyTank.getLiquidVolumeInGallons(), epsilon);
207             assertEquals(103672, smallFullTank.getLiquidVolumeInGallons(), epsilon);
208             assertEquals(103672557, halfFullTank.getLiquidVolumeInGallons(), epsilon);
209             assertEquals(207137770, lessThanTenLittersLeftTank.getLiquidVolumeInGallons(),
                epsilon);
210 
211         }
212 
213         @Test
214         public void testCompareTo() {
215             assertEquals(0, emptyTank.compareTo(emptyTank));
216             assertEquals(0, halfFullTank.compareTo(halfFullTank));
217             assertEquals(1, lessThanTenLittersLeftTank.compareTo(emptyTank));
218             assertEquals(1, lessThanTenLittersLeftTank.compareTo(smallFullTank));
219             assertEquals(-1, emptyTank.compareTo(bigFullTank));
220             assertEquals(-1, smallFullTank.compareTo(containsTwentyLittersTank));
221         }
222 
223         @Test
224         public void testGetPercentFilled1() {
225             assertEquals("Empty", emptyTank.getPercentFilled());
226             assertEquals("100% Full", smallFullTank.getPercentFilled());
227             assertEquals("100% Full", bigFullTank.getPercentFilled());
228             assertEquals("50% Full", halfFullTank.getPercentFilled());
229             assertEquals("99% Full", lessThanTenLittersLeftTank.getPercentFilled());
230         }
231 
232         @Test
233         public void testAdd() {
234             assertEquals(0,emptyTank.add(0).getLiquidVolume(), epsilon);
235             assertEquals(10,emptyTank.add(10).getLiquidVolume(), epsilon);
236             assertEquals(30,emptyTank.add(10).add(10).getLiquidVolume(), epsilon);
237             assertEquals(emptyTank.getCapacity(),emptyTank.add(5000).add(10).getLiquidVolume
                (), epsilon);
238         }
239 
240     }
241 
242 
243     // ************************************************************
```