MidtermReview/Railroadcar.java

```
// P3TrainDemo.java

/**
 * P3
 * Inheritance Problem
 *
 * Freight trains, like the one shown above, consist of 3 different types of railroad
    cars: rectangular box cars (which have a height, length and width), cylindrical tank
    cars (which have a diameter and a length), and engines. All railroad cars, regardless
    of their type, have two properties: a serial number and a year in which it was built.
    Shown below is a Java program that constructs a freight train by constructing
    individual BoxCars, TankCars and Engines and storing them in an array. The
    program then prints some basic information about these railroad cars.
 */

/**
class TrainDemo {
    public static void main (String [] args) {
    RailroadCar [] myFreightTrain =
        {
            new Engine ("OhBama", 1961),
            new BoxCar ("bar34", 1953, 10, 10.0, 8.9),
            new TankCar("TBrady", 1977, 10, 5.0),
            new BoxCar ("blah17", 1966, 10, 11, 5)
        };

        for (RailroadCar r : myFreightTrain){
            System.out.println( r + " and has a volume of " +
                r.volume() );
        }
    }
}

*/

/**

The output from this program appears below:
Engine #OhBama was built in 1961 and has a volume of 0.0
Box Car #bar34 was built in 1953 and has a volume of 890.0
Tank Car #TBrady was built in 1977 and has a volume of 785.3981
Box Car #blah17 was built in 1966 and has a volume of 550.0
The Engine class is pretty trivial to define:

class Engine extends RailroadCar {

    public Engine (String serialNumber, int yearBuilt) {
        super( serialNumber, yearBuilt );
    }

    public double volume () {
```

```
        return 0.0;
    }

    public String toString() {
        return "Engine " + super.toString();
    }
}

public abstract class RailroadCar {

    protected String serialNumber;
    protected int yearBuilt;

    public RailroadCar (String s, int y) {
        // complete this
    }

    public abstract double volume();

    public String toString () {
        // complete this
    }
}

*/

/**
 * Part A:
 * Fill in the two missing code fragments (one in the RailroadCar constructor, the
    other in the toString method) to complete the definition of the following
    RailroadCar class.
 */

/**
 * Part B:
 * What sort of error message would the javac compiler produce when trying to
    compile the main program if the statement
    public abstract double volume();
    were removed from the definition of RailroadCar?
 */

/**
 *
 * Part C:
 *  Now define the complete BoxCar class. Note that the volume of a BoxCar is
    easily computed as the product of its length, width and height.
 */

/**
 * Part D:
 * Suppose we want an easy way to keep track of the total number of RailroadCars
    that get constructed (whether they get stored in arrays or elsewhere). In one or
two
    simple, unambiguous English sentences, how can this be accomplished by making
    changes to the RailroadCar class only?
 */
```

```
107
108
109 class TrainDemo {
110
111     public static void main(String[] args) {
112
113         RailroadCar[] myFreightTrain = {
114             new Engine("OhBama", 1961),
115             new BoxCar("bar34", 1953, 10, 10.0, 8.9),
116             new TankCar("TBrady", 1977, 10, 5.0),
117             new BoxCar("blah17", 1966, 10, 11, 5)
118         };
119
120         for (RailroadCar r : myFreightTrain) {
121             System.out.println(r + " and has a volume of " + r.volume());
122         }
123         System.out.println();
124     }
125 }
126
127 public abstract class RailroadCar {
128
129     protected String serialNumber;
130     protected int yearBuilt;
131     protected static int count = 0;
132     protected double length;
133     protected double width;
134     protected double height;
135
136     public RailroadCar(String s, int y) {
137         this.serialNumber = s;
138         this.yearBuilt = y;
139         count++;
140     }
141
142     public abstract double volume();
143
144     public String toString() {
145         return "# " + this.serialNumber + " was built in " + this.yearBuilt;
146     }
147 }
148
149 class Engine extends RailroadCar {
150
151     public Engine(String serialNumber, int yearBuilt) {
152         super(serialNumber, yearBuilt);
153     }
154
155     public double volume() {
156         return 0.0;
157     }
158
159     public String toString() {
160         return "Engine " + super.toString();
161     }
162 }
```

```
163
164 // Part B
165 // error: RailroadCar is not abstract and does not override abstract method volume()
    in RailroadCar
166
167
168 // Part C
169 class BoxCar extends RailroadCar {
170
171     public BoxCar(String serialNumber, int yearBuilt, double height, double length,
    double width) {
172         super(serialNumber, yearBuilt);
173         this.height = height;
174         this.length = length;
175         this.width = width;
176     }
177
178     public double volume() {
179         return this.length * this.width * this.height;
180     }
181
182     public String toString() {
183         return "Box Car " + super.toString();
184     }
185 }
186
187 // Part D
188 // Add a static variable within the class and increment it to count the instances of
    RailroadCar objects
```