Assignment 8

Please submit the following java programs in a zipped folder on Canvas:

- Box Exercise:
  - Box Class (40 pts) – Box.java
  - Box Driver (20 pts)– BoxTest.java

Remember to include the pledge as a comment in every single file that you submit, followed by your name and U number.

"I pledge my Honor that I have not cheated, and will not cheat, on this assignment"

[insert name here], [insert U-number here]

Not including the pledge will result in a **50%** deduction of points for every file that does not have it. And no, putting one pledge as a comment on Canvas does not count.

Box Exercise

Write a class called Box that does contains the following:

1. Private members of the class to store the length, width, height and fullness of the box object
   - The fullness should be represented by a boolean variable called full.

2. A constructor that accepts no parameters, but sets the length, width and height to 11.25, 8.75 and 6 resepctively
   - All new box objects will be initially empty; i.e. the value of full should be false.
   - You will not pass the fullness of the box through the constructor!
3. A constructor to accept the length, width, height values of the box object
   - Again, all new box objects will be initially empty; i.e. the value of full should be false.
   - You will not pass the fullness of the box through the constructor!

4. Accessor (getter) methods for all of the instance data
   - You should have 4 of these methods

5. Mutator (setter) methods for all of the instance data
   - Again, you should have 4 of these methods

6. A toString Method to return a description of each box
   - If the Boolean value is false, output that the box is empty; otherwise the Boolean value is true and the box must be full.

To test the Box class, you will create a driver program called BoxTest that performs the following tasks:

1. Creates a box object using the constructor with no parameters
2. Creates two box objects using the constructor with parameters
    - You are expected (allowed) to pass values into the constructor via hardcoding
    - One of the box objects should be initialized to 0 (to test mutators)

3. Use the toString method to output the initial dimensions of the 3 boxes
4. Update the box object that was initialized to 0 using the mutators for that object
    - You are allowed to pass a hardcoded value to the mutators.

5. Prompts the user to enter a length, width and height of an item
6. If the length, width and height of the item is less than the length, width and height of a box, put the item in the box!
    - To do this, create an if else if structure that compares the dimensions of the items to the dimensions of each box
    - Use the accessors to get each dimension to compare them
    - If a box is big enough, use the mutator to change the status of the box from empty to full!
    - Tip: for this exercise, use values to ensure that your item can fit into one of the boxes

7. Use the toString method to output the final dimensions of the 3 boxes
    - Your output should show that the diminsions of one of your boxes was updated and that at least 1 box is full.

A sample of the output is shown below:

```
The initial dimensions of my boxes are:
Box size: 11.25 x 8.75 x 6.0 inches.
This box is empty.
Box size: 15.0 x 20.0 x 10.5 inches.
This box is empty.
Box size: 0.0 x 0.0 x 0.0 inches.
This box is empty.

Enter the dimensions of an item (separated by spaces): 9 10 7.5
The item can fit in box 2!
The final dimensions of my boxes are:
Box size: 11.25 x 8.75 x 6.0 inches.
This box is empty.
Box size: 15.0 x 20.0 x 10.5 inches.
```

This box is full.
Box size: 3.0 x 3.0 x 4.5 inches.
This box is empty.