1. Implement a class to represent a 2D box. A box has two attributes that define it: width and height. Accordingly, the constructor for the Box class should take the width and the height as parameters. In addition, the following operations are specified for a box. You should implement each of these operations using public (but not static) methods.

(a) Addition of boxes: The sum of two boxes \( b_1 \) and \( b_2 \) is a box whose width is equal to \( b_1\.width + b_2\.width \) and whose height is equal to \( b_1\.height + b_2\.height \).

(b) Scalar Multiplication (a box · an integer): The product of an integer \( c \) and a box \( b \) is a box whose width is equal to \( c \cdot b\.width \) and whose height is equal to \( c \cdot b\.height \).

(c) Equality Test: Two boxes are equal if and only if their widths are equal and their heights are equal. The method should be of the form public boolean equals(Box b).

HINT: The first two methods should return a new Box object.

You should also define a public String toString() method that returns a string representation of the Box. This method will automatically be called for any Box object that is used in a print statement.

Now write an additional .java file that includes the main method. Use it to test your Box class. For example,

Box b1, b2, b3;
b1 = new Box(1, 2);
b2 = new Box(10, 20);
b3 = b1.add(b2);
System.out.println(b1); // should print "h: 1, w: 2"
System.out.println(b2); // should print "h: 10, w: 20"
System.out.println(b3); // should print "h: 11, w: 22"

2. If you have not already done so, declare the width and height attributes of your Box class as private. This means that these variables cannot be directly accessed from outside of the Box class. That is, we could not write
System.out.println(b1.width). Nor could we write b1.width = 4. This is good programming practice, because width and height are internal properties of the Box class.

Suppose, however, that we want the user of the Box class to be able to determine the width and the height of a box b without using the toString() method. Then, you should define a getter method for width and height. For example, the getter method for width should be of the form public int getWidth() which returns the width of this Box object.

Now suppose that for some reason you want the user of the Box class to be able to modify the width and height of Box objects. To provide this functionality, define setter methods for width and height. HINT: The setter methods are similar to the constructor.

The use of private member variables along with getters and setters is a Java convention. In fact, Eclipse will create these methods for you via Source→Generate Getters and Setters.

3. Define a class to represent a complex number a + bi. Recall that a complex number is defined by the value of a and the value of b, so your constructor should take these the two values as parameters. Include the following methods.

(a) public Complex add(Complex c)
    which returns the sum of c and this complex number. For example, c1.add(c2) returns c1 + c2. Here this complex number is c1.

(b) public Complex add(double n)
    which returns the sum of this complex number and the scalar n.

(c) public Complex multiply(Complex c)
    which returns the product of c and this complex number.

Now write a driver program in a separate .java file to test your class.