U2T3: Method Overriding: Warm Up!

Name:	

Below is an `Adder` class with multiple constructors and methods. Some of the constructors and methods are *valid* **overloaded** constructors and methods. Others have method signatures that are *duplicates* of another method signature, and are thus *not* valid overloaded methods and will **not** compile (red squigglies in IntelliJ).

For each constructor and method:

1. Write the method signature in the middle column.
2. Determine if that method's signature is a *duplicate* of another signature; if it **is**, indicate in the right column that it will not compile because it is not properly overloaded. If it is *not* a duplicate, indicate that it will compile and is properly overloaded!

```
public class Adder {
    // instance variable
    private String name;
    private int counter;
    private int increment;

    // ---- constructors ----
    public Adder(String name) {
        this.name = name;
        counter = 0; // default value
        increment = 0; // default value
    }

    public Adder(String name, int counter) {
        this.name = name;
        this.counter = counter;
        increment = 0; // default value
    }

    public Adder(String name, int counter, int increment) {
        this.name = name;
        this.counter = counter;
        this.increment = increment;
    }
```

TYPE THE METHOD SIGNATURE HERE FOR EACH CONSTRUCTOR AND METHOD IN THE ADDER CLASS The first two constructors and the first two methods are done for you as examples	**Will this method compile?** YES if the method signature is **unique** (*no* duplicates), NO if the method signature is *not* unique!
`Adder(String)`	
`Adder(String, int)`	

```
public Adder(String name, int increment, int counter) {
    this.name = name;
    this.counter = counter;
    this.increment = increment;
}

public Adder(int counter, int increment, String name) {
    this.name = name;
    this.counter = counter;
    this.increment = increment;
}

// ---- methods ----
public void incrementCounter() {
    counter++;
}

public void incrementCounter(int amt) {
    counter += amt;
}

public void incrementCounter(double amt) {
    counter += (int) amt;
}

public int add(int num1, int num2) {
    return num1 + num2;
}

public int add(int num2, int num1) {
    return num1 + num2;
}

public int add(double num1, int num2) {
    int casted = (int) (num1 + num2);
    return casted;
}

public int add(int num1, double num2) {
    int casted = (int) (num1 + num2);
    return casted;
}
```

incrementCounter

incrementCounter(int)

```
    public int add(double num1, double num2) {
        int casted = (int) (num1 + num2);
        return casted;
    }

    public double add(double num1, double num2) {
        return num1 + num2;
    }

    public double add(int num1, int num2, double num3) {
        return num1 + num2 + num3;
    }

    public double add(int x, int y, double z) {
        return x + y + z;
    }

    public double addMore(int num1, int num2, double num3) {
        return num1 + num2 + num3;
    }
}
```


3. **Compare** your answers with your Partner!

Did you get the same answers? If not, discuss and come to a consensus!	

4. **TEST!** Open IntelliJ, create a new project (you do **not** need to upload this to GitHub unless you want to). Add a new Adder class and copy/paste the class above in. **Look for the red squigglies (compiler errors!).** Compare these with your **YES/NO** answers above.

5. **FIX!** Get the Adder class to **fully compile** (no red squigglies) by **REMOVING ONE** method of each pair with *duplicate* **method signatures** (Java only lets you have *unique method signatures* so you must get rid of one!)

How many constructors/methods did you have to **remove**, in total, to get your code to fully compile?	
Now count what remains: **A.** How many **overloaded constructors** does the Adder class have? **B.** How many **overloaded methods** does the Adder class have? **C.** How many *non-overloaded* methods does the Adder class have (different method name)?	A. B. C.
Let's say you had two methods with the same method signatures, but those methods did very different things, and you didn't want to just delete one. How might you go about changing your code so that you could keep both methods?	