English Name: ____________________

Worksheet: Simple `Vector` Class

Follow the steps to design a final two-dimensional `Vector` class.

a) Write a class named `Vector` that contains only two fields, named `x` and `y`, both of type `double`.

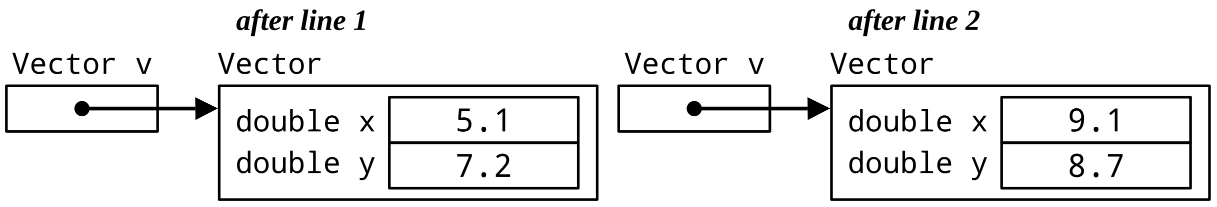
```
public class Vector {
   public double x;
   public double y;
}
```

b) Write a **constructor** for the above `Vector` class that takes two parameters of type `double` that are used to initialize the fields of the class. Do not include class information.

```
public Vector(double x, double y) {
    this.x = x;
    this.y = y;
}
```

c) Examine the code segment and diagrammatic representation of memory structures, below:

```
Vector v = new Vector(5.1, 7.5);
v.add(4.0, 1.5);
```



In the space below, write the **method** named `add` for the class `Vector`. It is to operate on a `Vector` object, not return any value, and take two parameters, `x` and `y`, both of type `double`. The method is to add the parameter values to the respective fields of the current object, with result as shown in the diagrams above.

```
public void add(double x, double y) {
    this.x += x;
    this.y += y;
}
```

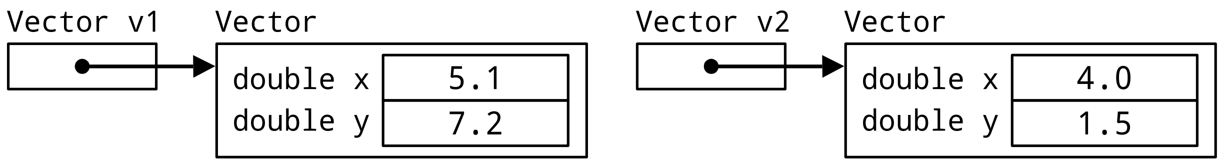
English Name: ____________________

Worksheet: Simple `Vector` Class

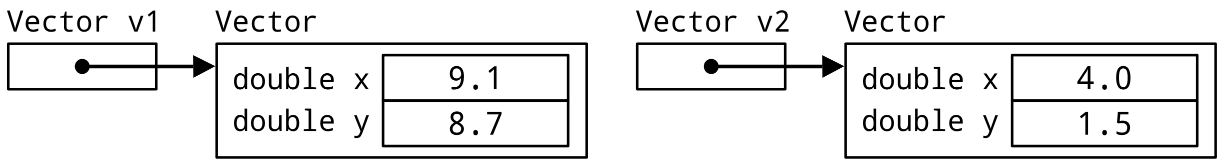
d) Examine the code segment and diagrammatic representation of memory structures, below:

```
1 Vector v1 = new Vector(5.1, 7.5);
2 Vector v2 = new Vector(4.0, 1.5);
3 v1.add(v2);
```

after line 2



after line 3



Write another **method** named `add` for the above `Vector` class (overload the `add` method). This method is to return no value, and take a single parameter, `v`, of type `Vector`. It is to add the fields of the `v` object to the respective fields of the current object, with results as shown in the diagrams above.

```
public void add(Vector v) {
    this.x += v.x;
    this.y += v.y;
}
```

e) Write a **method** named `equals` for the above `Vector` class. This method is to take a single parameter, `v`, of type `Vector`. It is to return `true` if both the `x` and `y` fields of the parameter object, `v`, are equal to the `x` and `y` fields, respectively, of the current object.

```
public boolean equals(Vector v) {
    return this.x == v.x && this.y == v.y;
}
```