Lab 8: Vector Object

Your job is to finish the `Vector` object that has already been started. This object represents a vector in three-dimensional space. You might make an object like this if you were programming a game or a physics simulation.

The `Vector` object is filled with "stub functions" that do nothing, and return `null` or `NaN` (not a number) values. You need to fill in these functions to create a working `Vector` object.

In addition, `TestVectors.java` is the code for a program that uses the `Vector` object.

The functions that you need to implement are as follows:

- The constructor, which takes 3 `double`s and assigns them to fields `x`, `y`, and `z`.
- `add()`, which takes a second `Vector`, adds it to the first, and returns a new `Vector` that is the resulting sum.
- `subtract()`, which takes a second `Vector`, subtracts it from the first, and returns a new `Vector` that is the resulting difference.
- `multiply()`, which takes a `double` scalar value, multiplies it by the `Vector`, and returns the result.
- `calcCrossProduct()`, which takes a second `Vector`, calculates the cross product `Vector`, and returns it.
- `calcDotProduct()`, which takes a second `Vector`, calculates the dot product `double`, and returns it.
- `calcLength()`, which calculates the length of the `Vector`.
- `makeNormalizedVector()`, which returns a new `Vector` with the same proportions as the old one, but has a length of 1.0.
- `calcAngle()`, which uses the dot product and length methods to calculate the angle between two `Vector`s. Its argument is the second `Vector`, and it returns the angle in radians.
- `toString()`, which creates a String representing the`Vector` in human-readable terms.
- `equals()`, which takes another`Vector` and returns true if they're equal.
- Accessors for the components of the `Vector`, called `getX()`, `getY()`, and `getZ()`.
- Setters for the three components, called `setX()`, `setY()`, and `setZ()`.

You may have forgotten how to do some of these operations. Wikepedia is a wonderful resource for this sort of thing!

As you are programming, run `TestVectors` repeatedly to make sure that everything works. Read and understand the file `TestVectors.java`. When you finish, you should get an output similar to the following:

Est. 1888 UNIVERSITY of PUGET SOUND

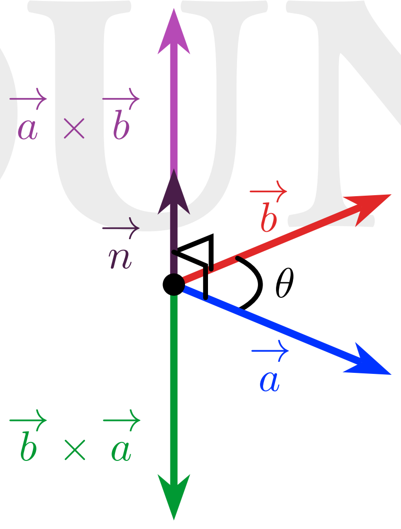
```
Vectors:         [1.0, 2.0, 3.0] [4.0, 5.0, 6.0]
Addition:        [5.0, 7.0, 9.0]
Subtraction:     [-3.0, -3.0, -3.0]
Multiplication:  [0.5, 1.0, 1.5]
Cross Product:   [-3.0, 6.0, -3.0]
Dot Product:     32.0
Angle:           0.2257261285527342
Length:          3.7416573867739413
Normalized:      [0.2672612419124244,0.5345224838248488,0.8017837257372732]
Norm Length:     1.0
Vectors equal?   false
1st is [1,2,3]?  true
Single Coords:   x:1.0 y:2.0 z:3.0
Adjusted:        [30.0, 20.0, 10.0]
```

Alternatively, you can test the object using BlueJ. When you right click on the object, choose `new Vector(double x, double y, double z)`. This lets you make a brand new `Vector` object to test. Make one `Vector` called `vector1` with inputs `1.0`, `2.0`, `3.0`, and another called `vector2` with inputs `4.0`, `5.0`, `6.0`. Then you'll be able to call all the methods as you write them, testing them one at a time. You should get the same answers as above.

You don't need to be as careful about stylistic concerns for this homework, since the commenting and method signatures are already written for you. However, take a good look at the Javadoc-style comments, since you'll have to write your own soon enough!

You only need to turn in `Vector.java`. We will test your code using our own `TestVectors` class that is different from the one provided.

Est. 1888 UNIVERSITY of PUGET SOUND


$\vec{a} \times \vec{b}$
$\vec{b}$
$\vec{n}$
θ
$\vec{a}$
$\vec{b} \times \vec{a}$