Interface – Vector

An interface (in Java) can be defined as a contract to guarantee the implementation of a set of methods by the implementation class. Such a contract ensures interchangeability among classes. In this assignment you will be given an interface file that defines a set of operations performed on a 3D vector. The operations are defined mathematically as follows. $< u, v, w > and < x, y, z >$ represent vectors where u, v, w, x, y, z are real numbers (scalars).

Vector addition

$$< u, v, w > + < x, y, z > = < u + x, v + y, w + z >$$

Vector subtraction

$$< u, v, w > - < x, y, z > = < u - x, v - y, w - z >$$

Scalar multiplication

$$\alpha \times < u, v, w > = < \alpha \times u, \alpha \times v, \alpha \times w >$$

Dot product

$$< u, v, w > \cdot < x, y, z > = \ u \times x + v \times y + \ w \times z$$

Cross product

$$< u, v, w > \times < x, y, z > = < vz - wy, uz - xw, uy - vx >$$

Vector norm

$$|< u, v, w >| \ = \sqrt{u^2 + v^2 + w^2}$$

Vector unit

$$Unit(< u, v, w >) = < \frac{u}{|< u, v, w >|}, \frac{v}{|< u, v, w >|}, \frac{w}{|< u, v, w >|} >$$

Angle between two vectors

$$AngleBetween(< u, v, w >, < x, y, z >) = \ \cos^{-1}\left(\frac{< u, v, w > \cdot < x, y, z >}{|< u, v, w >| \ \times |< x, y, z >|}\right)$$

Direction angles

$$DirectionAngle(XCOMPONENT, < u, v, w >) = \frac{u}{|< u, v, w >|}$$

$$DirectionAngle(YCOMPONENT, < u, v, w >) = \frac{v}{|< u, v, w >|}$$

$$DirectionAngle(ZCOMPONENT, < u, v, w >) = \frac{w}{|< u, v, w >|}$$

Parallel vectors

$$Parallel(< u, v, w >, < x, y, z >) = \begin{cases} true\ if\ |< u, v, w > \times < x, y, z >| = 0 \\ false\ otherwise \end{cases}$$

Anti-parallel vectors

$$AntiParallel(< u, v, w >, < x, y, z >) = \begin{cases} true\ if\ AngleBetween(< u, v, w >, < x, y, z >) = \pi \\ false\ otherwise \end{cases}$$

Projection of one vector onto another

$$Projection(< u, v, w >, < x, y, z >) = \big(< u, v, w > \cdot\ Unit(< x, y, z >)\big) \times Unit(< x, y, z >)$$

Vector equality

$$Equal(< u, v, w >, < x, y, z >) = \begin{cases} true\ if\ u = x, v = y, w = z \\ false\ otherwise \end{cases}$$

The Problem

Provide an implementation (in Java) for the vector interface file provided. The implementation should have three private member variables, data type double. The static member variables (constants) defined in the interface *XCOMPONENT*, *YCOMPONENT*, and *ZCOMPONENT* are used as selectors to specify the first (u and x above), second (v and y above), and third (w and z above) member variables in the accessor and mutator methods. That is, you are to provide a single Get method used to get any one of the components and a single Set method used to set any one of the components.

The toString method should return a string of the form **`<27.3, 14.7, 19.8>`**.

You will be given a main method. You must test your code with this main method without modification.

Deliverables

- Source code attached to assignment in Blackboard
- Screen shot of results from running your main function
- A reflective essay on your successes and difficulties.

Notes

- For C++ use **`#include <cmath>`** and **`std::sqrt(x)`** to access the square root [of a floating point number] function
- For the complex number sqrt function you may return only the positive result (you may ignore the negative result)

main method

```
public static void main (String argv[])
{
	Vector v0 = new Vector(1.0, 2.0, 3.0);
	Vector v1 = new Vector(4.0, 5.0, 6.0);
	VectorInterface vresult;
	vresult = v0.Add(v1);
	System.out.println(vresult);
	vresult = v0.Subtract(v1);
	System.out.println(vresult);
	System.out.println(v0.Multiply(2.0));
	System.out.println(v0.Dot(v1));
	System.out.println(v0.Cross(v1));
	System.out.println(v0.Norm());
	System.out.println(v0.Unit());
	v0 = new Vector(2, -3, 4);
	v1 = new Vector(5, 2, 1);
	System.out.println(v0.AngleBetween(v1) * 180.0 / Math.PI);
	System.out.println(v0.DirectionAngle(VectorInterface.XCOMPONENT));
	System.out.println(v0.DirectionAngle(VectorInterface.YCOMPONENT));
	System.out.println(v0.DirectionAngle(VectorInterface.ZCOMPONENT));
	System.out.println("parallel " + v0.Parallel(v0));
	System.out.println("parallel " + v0.Parallel(v1));
	v0 = new Vector(1, 0, 0);
	v1 = new Vector(-1, 0, 0);
	System.out.println("anti-parallel " + v0.AntiParallel(v1));
	System.out.println("anti-parallel " + v0.AntiParallel(v0));

	Vector vunit = (Vector)v0.Unit();
	System.out.println(vunit.Norm());
	System.out.println("equals " + v0.equals(v0));
	System.out.println("equals " + v0.equals(v1));

	v0 = new Vector(2, -3, 4);
	v1 = new Vector(5, 2, 1);
	System.out.println(v0.Projection(v1));
}
```