PROTOCOLO:

/*

Instituto Tecnológico de León

Ingeniería en Sistemas Computacionales

Estructuras de datos

Pablo Vargas Bermúdez

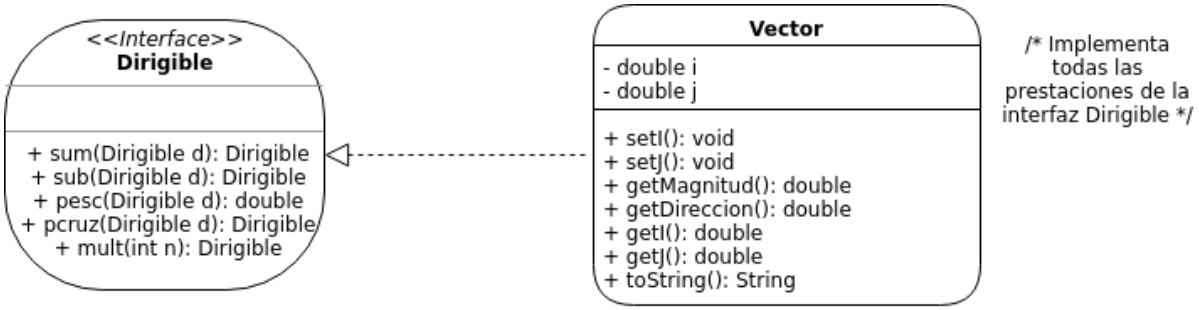
Tarea #: 5

Fecha pactada: 15 de noviembre de 2019

Fecha de entrega: 15 de noviembre de 2019

*/

DIAGRAMAS UML:



ALGORITMIA:

1.- Meta:

Probar la clase Vector que implementa Dirigible ADT
de forma persistente (leyendo y escribiendo a archivos)
Se pretende probar:
* Setters
* Getters
* Operaciones Aritméticas
A elección del usuario

2.- Datos(n):

// Usa n para saber cuántos Vectores leer
vectors[] = nuevo Vector[n]
abrir(in)
Desde i = 0 mientras i < n incrementar i en 1
x = in.leerDouble()
y = in.leerDouble()
vectors[i] = nuevo Vector(x, y)
terminar
cerrar(in)

3.- Cálculos:

3.1.- TestSetters:

vectors[] = llamar a datos(1)

```
        comp[] = {0, 1}
        msg  = "v(original) = " + vectors[0]
        vectors[0].setI(comp[0])
        vectors[0].setJ(comp[1])
        msg += "v(despues)  = " + vectors[0]
    3.2.- TestGetters:
        vectors[] = llamar a datos(1)
        msg  = "I   -> " + vectors[0].getI()
        msg += "J   -> " + vectors[0].getJ()
        msg += "|v| -> " + vectors[0].getMagnitud()
        msg += "<   -> " + vectors[0].getDireccion()
    3.3.- TestOperations:
        vectors[] = llamar a datos(2)
        escalar = 2
        msg  = "v1 + v2 = " + vectors[0].sum(vectors[1])
        msg += "v1 - v2 = " + vectors[0].sub(vectors[1])
        msg += "v1 . v2 = " + vectors[0].pesc(vectors[1])
        msg += "v2 x v2 = " + vectors[0].pcruz(vectors[1])
        msg += "2(v1)   = " + vectors[0].mult(escalar)
4.- Resultados:
    llamar a escribir()
    llamar a mostrar()
    4.1.- Escribir:
        abrir(out)
        out.escribir(msg)
        cerrar(out)
    4.2.- Mostrar:
        Escribir(msg)
5.- Navegabilidad:
    // Usa in  como arg[0]
    // Usa out como arg[1]
    Hacer
        llamar a MostrarMenu()
    Mientras(option < 0 o option >= 5)
    llamar a Direccionar()
    5.1.- MostrarMenu:
        Escribir("Menu: ")
        Escribir("1.- Setters")
        Escribir("2.- Getters")
        Escribir("3.- Operaciones")
        Escribir("4.- Salir")
        Escribir("opción: ") option = ?
    5.1.- Direccionar:
        Según(option) empezar
            caso 1: llamar a testSetters();    break;
            caso 2: llamar a testGetters();    break;
            caso 3: llamar a testOperations(); break;
            caso 4: Salir del programa
```

terminar

CLASES ENCAPSULADAS:

DIRIGIBLE

```
public interface Dirigible {
   public Dirigible sum(Dirigible d);
   public Dirigible sub(Dirigible d);
   public double pesc(Dirigible d);
   public Dirigible pcruz(Dirigible d);
   public Dirigible mult(int n);
}
```

VECTOR

```
public class Vector implements Dirigible {
   double i, j;
   public Vector(Vector v) {
      i = v.getI();
      j = v.getJ();
   }
   public Vector(double i, double j) {
      this.i = i;
      this.j = j;
   }
   @Override
   public Dirigible sum(Dirigible d) {
      Vector v = (Vector) d;
      double i2 = v.getI() + i;
      double j2 = v.getJ() + j;

      return new Vector(i2, j2);
   }
   @Override
   public Dirigible sub(Dirigible d) {
      Vector v = (Vector) d;
      double i2 = v.getI() - i;
      double j2 = v.getJ() - j;
      return new Vector(i2, j2);
   }
   @Override
   public double pesc(Dirigible d) {
      Vector v = (Vector) d;
      double i2 = v.getI() * i;
      double j2 = v.getJ() * j;
      return i2 + j2;
   }
   @Override
   public Dirigible pcruz(Dirigible d) {
      Vector v = (Vector) d;
      double i2 = i * v.getJ();
      double j2 = j * v.getI() * -1;
```

```
        return new Vector(i2, j2);
    }
    @Override
    public Dirigible mult(int n) {
        double i2 = n * i;
        double j2 = n * j;
        return new Vector(i2, j2);
    }
    public void setI(double d) { i = d; }
    public void setJ(double d) { j = d; }
    public double getMagnitud() { return Math.sqrt(i * i + j * j); }
    public double getDireccion() { return Math.tan(i / j); }
    public double getI() { return i; }
    public double getJ() { return j; }
    @Override
    public String toString() { return "<" + i + ", " + j + ">"; }
}
```

CLASES DE PRUEBA:

Persistencia

```
package Tarea5;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class Persistencia {
    String permission[] = {"r", "w"};
    Scanner in;
    FileWriter out;
    File f;
    public Persistencia(String f, String mode) {
        this.f = new File(f);
        init(mode);
    }
    private void init(String mode) {
        try {
            if(mode.equals(permission[0]))
                in = new Scanner(f);
            else if(mode.equals(permission[1]))
                out = new FileWriter(f);
        } catch(Exception e) {
            System.out.println("Ocurrió un error al abrir el archivo");
            System.exit(0);
        }
    }
    public double leerDouble() {
        double d;
        if(in == null) return -1;
        try {
```

```
            d = Double.parseDouble(in.nextLine());
        } catch(Exception e) {
            return 0;
        }
        return d;
    }
    public void escribir(String msg) {
        if(out == null) return;
        try {
            out.write(msg);
        } catch(IOException ex) {
            System.out.println("Ocurrió un error al escribir");
            System.exit(0);
        }
    }
    public void cerrar() {
        try {
            if(out != null) out.close();
            else if(in != null) in.close();
        } catch(Exception e) {
            System.out.println("Ocurrió un error al cerrar el archivo");
            System.exit(0);
        }
    }
}
```

Negocio

```
package Tarea5;
import Examen1.Vector;
public class TestVector {
    String msg, in, out;
    int option;
    public static void main(String[] args) {
        TestVector t = new TestVector();
        if(args.length != 2) {
            t.meta();
            System.exit(1);
        }
        t.in  = args[0];
        t.out = args[1];
        while (true) {
            do t.mostrarMenu();
            while(t.option <= 0 || t.option >= 5);
            t.direccionar();
            t.resultados();
        }
    }
    public void meta() {
        Vista v = new Vista();
```

```
   String msg = "Probar la clase Vector derivada de";
   msg += " Dirigible ADT";
   v.mostrar(msg);
}
public Vector[] datos(int n) {
   Vector vectors[] = new Vector[n];
   Persistencia p = new Persistencia(in, "r");
   for(int i = 0; i < n; ++i) {
      double x = p.leerDouble();
      double y = p.leerDouble();
      vectors[i] = new Vector(x, y);
   }
   p.cerrar();
   return vectors;
}
public void testSetters() {
   Vector vectors[] = datos(1);
   int comp[] = {0, 1};
   msg  = "v(original) = " + vectors[0] + "\n";
   vectors[0].setI(comp[0]);
   vectors[0].setJ(comp[1]);
   msg += "v(despues)  = " + vectors[0] + "\n";
}
public void testGetters() {
   Vector vectors[] = datos(1);
   msg  = "I   -> " + vectors[0].getI() + "\n";
   msg += "J   -> " + vectors[0].getJ() + "\n";
   msg += "|v| -> " + vectors[0].getMagnitud() + "\n";
   msg += "<   -> " + vectors[0].getDireccion() + "\n";
}
public void testOperations() {
   Vector vectors[] = datos(2);
   int escalar = 2;
   msg  = "v1 + v2 = " + vectors[0].sum(vectors[1]) + "\n";
   msg += "v1 - v2 = " + vectors[0].sub(vectors[1]) + "\n";
   msg += "v1 . v2 = " + vectors[0].pesc(vectors[1]) + "\n";
   msg += "v2 x v2 = " + vectors[0].pcruz(vectors[1]) + "\n";
   msg += "2(v1)   = " + vectors[0].mult(escalar) + "\n";
}
public void resultados() {
   escribir();
   mostrar();
}
public void escribir() {
   Persistencia p = new Persistencia(out, "w");
   p.escribir(msg);
   p.cerrar();
}
```

```
    public void mostrar() {
        Vista v = new Vista();
        v.mostrar(msg);
    }
    private void mostrarMenu() {
        Vista v = new Vista();
        v.mostrarln("Menu: ");
        v.mostrarln("1.- Setters");
        v.mostrarln("2.- Getters");
        v.mostrarln("3.- Operaciones");
        v.mostrarln("4.- Salir");
        v.mostrar("opción: ");
        option = v.getInt();
    }
    private void direccionar() {
        switch(option) {
            case 1: testSetters();    break;
            case 2: testGetters();    break;
            case 3: testOperations(); break;
            case 4: System.exit(0);
        }
    }
}
```

Vista

```
import java.util.Scanner;
public class Vista {
    public void mostrar(String msg) {
        System.out.print(msg);
    }
    public void mostrarln(String msg) {
        mostrar(msg);
        System.out.println();
    }
    public int getInt() {
        Vista v = new Vista();
        Scanner in = new Scanner(System.in);
        while(!in.hasNextInt())
            v.mostrarln(in.nextLine() + " no es válido");
        return in.nextInt();
    }
}
```

CORRIDAS:

In

```
1
2
3
4
In (END)
```

Ejecución

```
λ ~/Universidad/EstructuraDatos/Estructuras/src/ java Tarea5.TestVector In Out
Menu:
1.- Setters
2.- Getters
3.- Operaciones
4.- Salir
opción: 3
v1 + v2 = <4.0, 6.0>
v1 - v2 = <2.0, 2.0>
v1 . v2 = 11.0
v2 x v2 = <4.0, -6.0>
2(v1)   = <2.0, 4.0>
Menu:
1.- Setters
2.- Getters
3.- Operaciones
4.- Salir
opción: 4
```

Out

```
v1 + v2 = <4.0, 6.0>
v1 - v2 = <2.0, 2.0>
v1 . v2 = 11.0
v2 x v2 = <4.0, -6.0>
2(v1)   = <2.0, 4.0>
```