Se generará un programa que reciba un archivo de texto separado por comas, o un arreglo de 2 dimensiones (lista enlazada con 2 valores relacionados): columna 1 será la etiqueta, columna 2 será el valor.

El programa recibirá los valores, calculará la desviación estándar, y hará la asignación de VS,S,M,L,VL de acuerdo a los cálculos hechos, indicando las etiquetas que más se ajusten a dichos valores.

Los valores se presentarán en pantalla por dos medios: 1. consola (para aplicación standalone), 2. Interfaz HTML simple (como una tabla), para la aplicación de Heroku.

Los pasos a seguir serán:

1. Revisar las utilidades de programas pasados, para ver qué se puede reutilizar o modificar para que funcione en este programa, además de determinar qué clases nuevas se necesitarán. 3 horas (planeación y diseño)

1.1. Determinar las clases modificadas y el código a reutilizar/modificar. 30 min (planeación)

- Value.java se puede reutilizar casi igual, sólo cambiando nombres de variables para que sean apropiadas al contexto del programa.

- ValuesArray.java sólo el método del read from file, pues los otros son específicos para la funcionalidad anterior.

-ValuesArray.java de tarea1 se puede utilizar los métodos calculateStdDev y calculateAvg

- Calculator.java se puede reutilizar, probablemente será mejor generar un archivo config para la ruta del archivo, de manera que se pueda dinamizar dicha entrada, y generar una versión que lea directamente desde valores fijos (un constructor por defecto?) para heroku, y otra que lea desde archivo, para la standalone.

- RegressionTest.java, se podría renombrar y reutilizar; tal vez dar nombre más genérico para otras ocasiones no tener que contarlo como archivo nuevo?

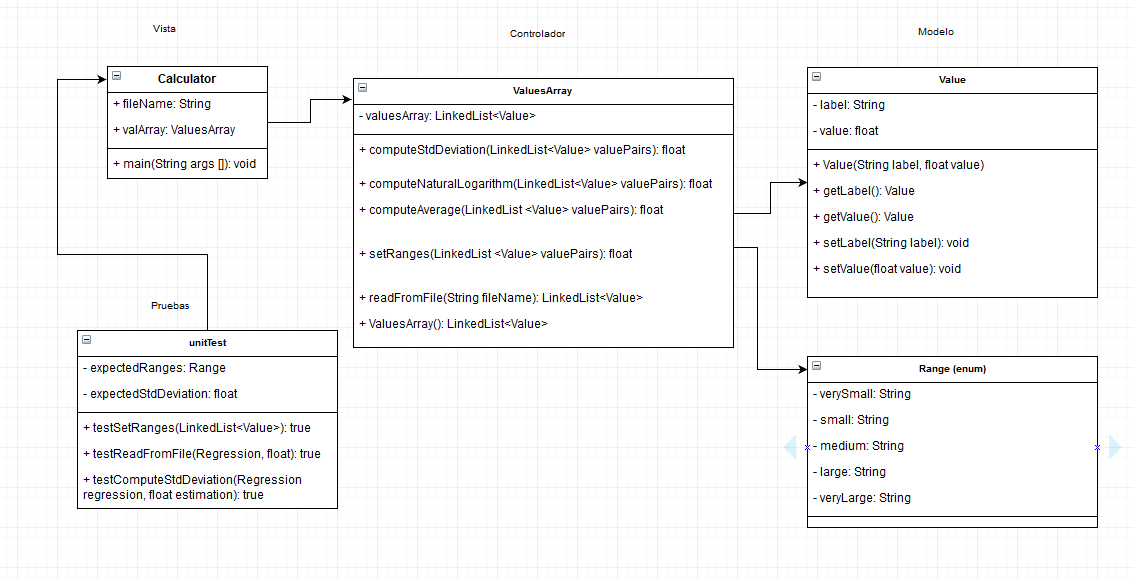
La parte que se puede reutilizar de esto es la creación de los elementos, cambiando valores.

1.2. Determinar y diagramar las clases nuevas, así como los métodos y utilidades para la interfaz de Heroku. 30 min

Las clases nuevas serían: Range (enum)

HTML Interface (?) - averiguar para esta parte.

1.3. Generar diagrama de clases detallado y métodos que se van a utilizar. 1 hora



1.4. Buscar código para utilizar en la parte de Heroku, añadir a diagrama. 1 hora

código para inicializar app en heroku (está en el procfile):

web: java -jar target/helloworld.jar

para dejar de correr dynos:

heroku ps:scale web=0

Se debe agregar la configuración al POM.xml:

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<version>2.3</version>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<finalName>helloworld</finalName>

</configuration>

</plugin>

se hace para instalar:

mvn clean install

ya teniéndolo local, se puede probar ahí, con:

heroku local web

para mandarlo a heroku, se utilizan:

git add .

git commit -m "NombreEntrega"

git push heroku master

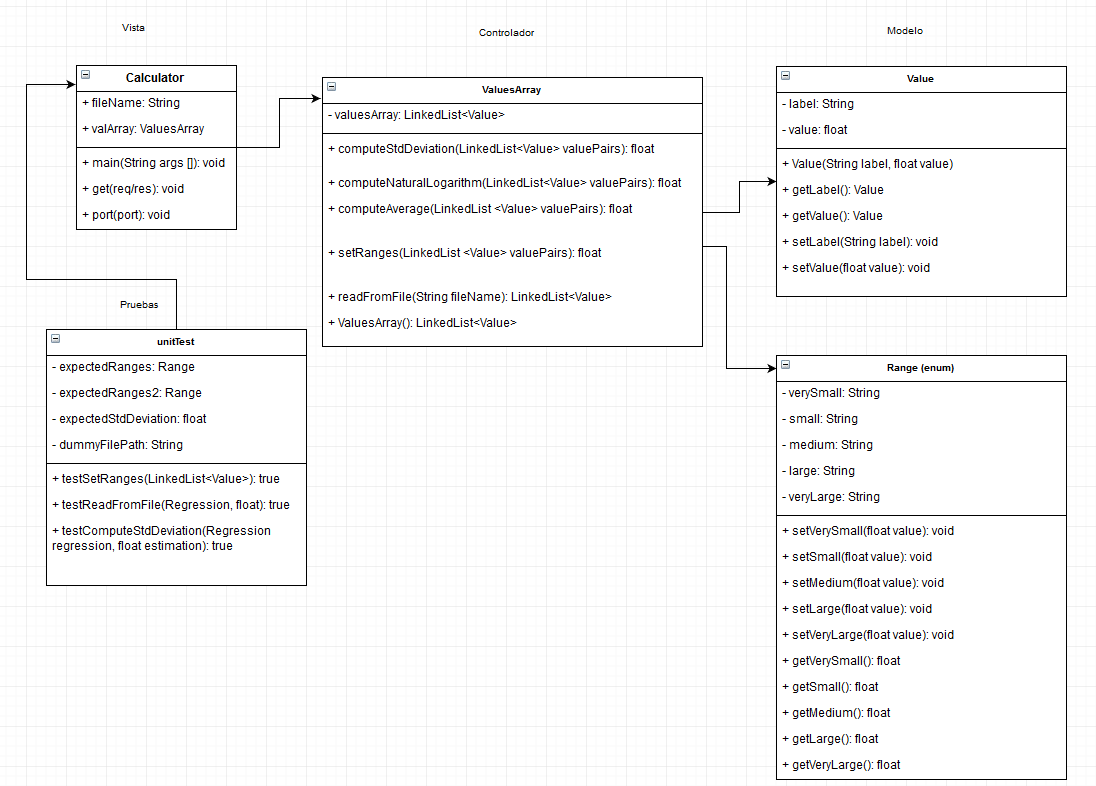
Para probar que todo funcione, se usa:

heroku open urlMetodoWeb

1.5 Se proyecta, basado en los programas anteriores, que se va a inyectar aproximadamente 10 defectos en total; probablemente 1 desde planeación, 5 en diseño y 4 en código.

El porcentaje de errores quedará así: 10% en planeación, 50% en diseño, y 40% en código.

Después de revisión de diseño, las clases quedan:



2. Generar las clases principales (Modelo, Vista, Controlador) para que se manejen los valores, los métodos de cálculo de desviación estándar, y la asignación de tamaño a cada par de valores (etiqueta, valor). 3 horas (codificación)

2.1. Los modelos serán las clases de los pares de valores, el enumerador de etiquetas (VS,S,M,L,VL), y tal vez la lista de pares de etiquetas y valores, de ser necesario. (30 min)

2.2. La vista será el main, donde probablemente se haga una interfaz adicional para poder mostrar los valores por página web, de manera que se puedan imprimir en un dyno de heroku. (2 horas)

2.3. El controlador será la clase donde se calculará la desviación estándar, y se asignarán los pares de valores para imprimirlos en la vista. 1

3. Una vez se tenga la funcionalidad lista, trabajar en la visualización de los datos, y la impresión hacia una página web en heroku. 1.5 horas

4. Hacer las pruebas de todo el proyecto; subir a heroku. 2 horas

Se ha subido el app a Heroku; queda en las URL: https://young-caverns-71471.herokuapp.com/test1 y

https://young-caverns-71471.herokuapp.com/test2

5. Post mortem. 30 min a 1 hora.