Programa 1

Pruebas de Software

Juan Carlos Marín

UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

Daniel Felipe Ospina Pérez - 1036668605
Pedro Pablo Gallego Pinzón - 1037649475
Mario Andrés García Toro - 1017244629

Universidad de Antioquia
Medellín
2017 – 2

Casos de prueba

Método	Especificación
Clase: ListaLigada void insertarDato(float x, float y)	Se ingresa a la lista los datos para cada una de las columnas, se crea un nuevo nodo en el cual se agregan los datos pasados como argumento, este nodo se agrega al final de la lista, en el nodo cabeza se suma en 1 la cantidad de datos.
Clase: Calculo float calcularMedia(Lista y)	Lee los datos ingresados en una lista y devuelve un flotante con el resultado del promedio de estos; aplicando la siguiente fórmula: $x_{avg} = \frac{\sum_{i=1}^{n} x_i}{n}$
Clase: Calculos float calcularDesvEst(Lista z)	Lee uno a uno los datos de la lista, y con ellos aplica la fórmula estipulada para la desviación estándar, la cual es: $\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_{avg})^2}{n-1}}$ Donde: ● Σ: símbolo sumatoria ● *i*: posición del nodo ● *x*: dato del nodo (flotante) ● *n*: número de datos Retorna el resultado de la solución a dicha fórmula.
Clase: Programa1 Lista leerDatos(String archivo)	Lee los datos del archivo .txt según el formato de la tabla xx y los ingresa en un objeto ListaLigada para retornarlo.

Estimate Proxy Size	Development Hours	Formato Archivo.txt
160	15.0	160 15.0 591 69.9 114 6.5 229 22.4 230 28.4 270 65.9 128 19.4 1657 198.7 624 38.8 1503 138.2
591	69.9	
114	6.5	
229	22.4	
230	28.4	
270	65.9	
128	19.4	
1657	198.7	
624	38.8	
1503	138.2	

Pruebas unitarias realizadas

ID	Método a probar	Entrada 1	Salida esperada
1	insertarNodo	Nodo con datos 160, 15.0	dato agregado al nodo y añadido a la lista en posición final, número de datos sumado en 1.
2	insertarNodo	Nodo con datos 1657, 198.7	dato agregado al nodo y añadido a la lista en posición final, número de datos sumado en 1.
3	insertarNodo	Nodo con datos -1657, 198.7	“Error por número negativo.”
4	insertarNodo	Nodo con datos 1657, -198.7	“Error por número negativo.”
5	calcularMedia	listaVacía	“Lista vacía, ingrese datos”
6	calcularMedia	null	“Parámetro inválido”
7	calcularMedia	listaDatos	550.6, 60.32
8	calcularDesvEst	listaVacía	“Lista vacía, ingrese datos”

9	calcularDesvEst	null	“Parámetro inválido”
10	calcularDesvEst	listaDatos	572.03, 62.26
11	calcularDesvEst	listaUnDato	“La lista debe tener más de un dato para calcular la desviación estándar”
12	leerDatos	archivoCaracteresInvalidos.txt	“El archivo no tiene el formato correcto.”
13	leerDatos	archivoNumColumDiferenteDe2.txt	“El archivo no tiene el formato correcto.”
14	leerDatos	archivoInexistente.txt	“El archivo no existe en la misma carpeta del ejecutable.”
15	leerDatos	archivoVacio.txt	“El archivo está vacío.”
16	leerDatos	archivoDatos.txt	listaDatos

Resultados pruebas

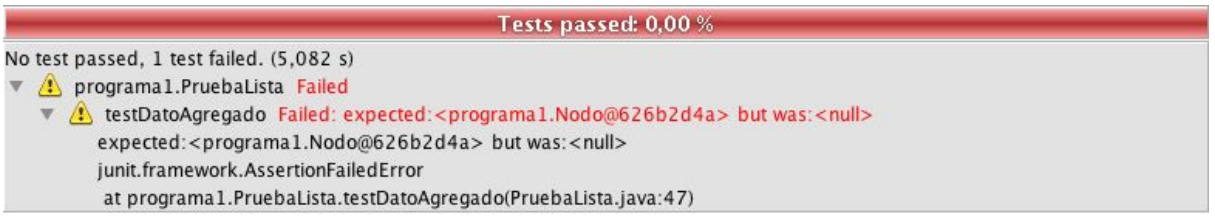
Siguiendo con la metodología planteada en el documento sobre PSP, y referencias encontradas sobre TDD se procedió a, con base en las pruebas diseñadas anteriormente realizar una a una su implementación así:

1. InsertarNodo (testDatoAgregado):
   1.1. Implementación prueba y comprobación de que esta falla.

   Se puede ver que al no estar implementado el método insertar Nodo la prueba falla.

   1.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.

Se puede ver que hay un error en la implementacion del metodo insertar nodo, este se debe a que la lista en su nodo cabeza al inicializarse las ligas apuntan al valor null, se corrige la implementación de este método y se ejecutan nuevamente las pruebas.



Nuevamente la prueba falla, esta vez porque en la implementacion del metodo Insertar nodo no se le asigna la liga del nodo cabeza al primer nodo de la lista, se corrige este error y se ejecutan nuevamente las pruebas.

Ahora se puede ver que el resultado de esta es exitosa, se continúa con el siguiente caso de prueba.

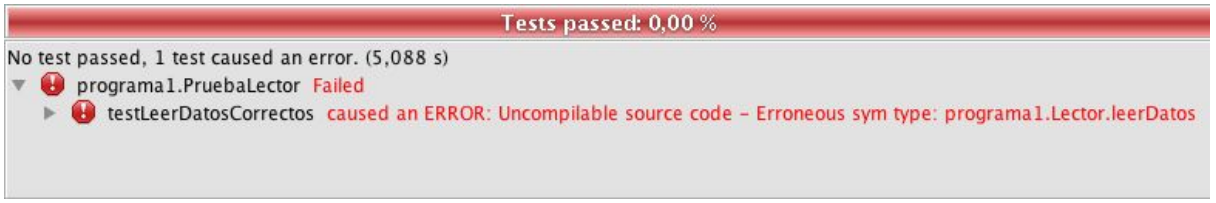
2. insertarNodo (testDatoNegativoAgregado):
   2.1. Implementación prueba y comprobación de que esta falla.

Se puede ver que la prueba falla, ya que el método insertar nodo no valida que los datos ingresados sean positivos y por esto agrega el nodo con los datos negativos a la lista.

   2.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.

Al ejecutar nuevamente todas las pruebas, luego de hacer las correcciones necesarias se obtiene un resultado exitoso en todas las pruebas, se continúa con el siguiente caso de prueba.

3. leerDatos (testLeerDatosCorrectos)
   3.1. Implementación prueba y comprobación de que esta falla.



Se puede ver que al no estar implementado el método leerDatos la prueba falla.

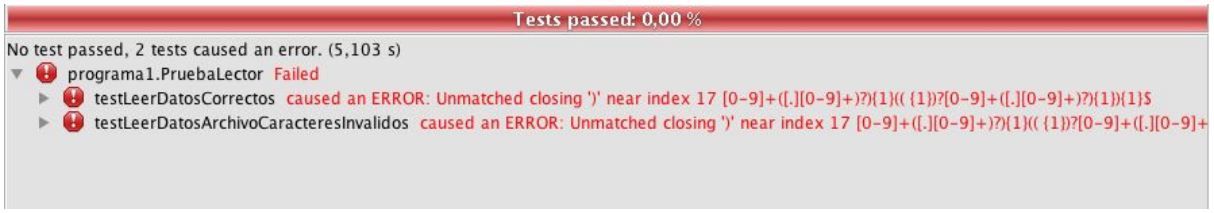
   3.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.

La prueba es exitosa, el método agrega de forma correcta los datos del archivo en el formato definido a la lista.

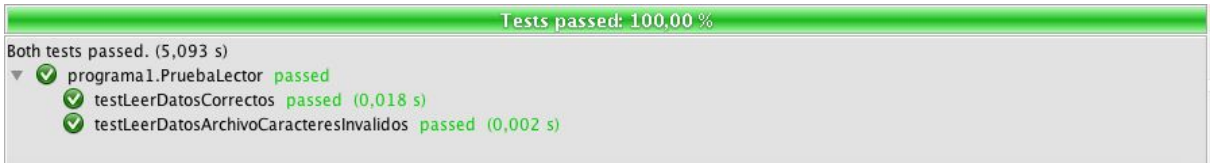
4. leerDatos (testLeerDatosArchivoCaracteresInvalidos)
   4.1. Implementación prueba y comprobación de que esta falla.

La prueba falla ya que al entrar un archivo en el cual el formato de este no es el correcto no lo valida y genera un error.

4.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.



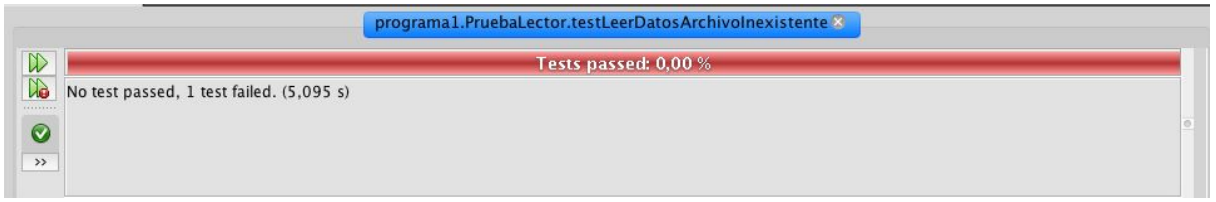
Una vez implementado el método se ejecuta la prueba en la cual el resultado es fallido nuevamente, se encuentra que la gramática usada para validar que el formato de cada una de las líneas tiene un error en su construcción, se corrige esto y se vuelven a correr todas las pruebas.



Ahora se puede ver que el resultado de esta es exitosa, se continúa con el siguiente caso de prueba.

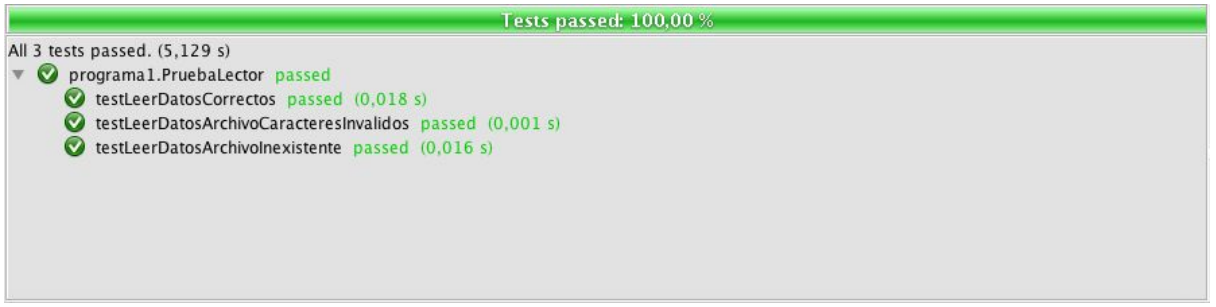
5. leerDatos (testLeerDatosArchivoInexistente)

5.1. Implementación prueba y comprobación de que esta falla.



Al ejecutar la prueba con un archivo que no existe la prueba falla ya que no se valida antes que este exista.

5.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.



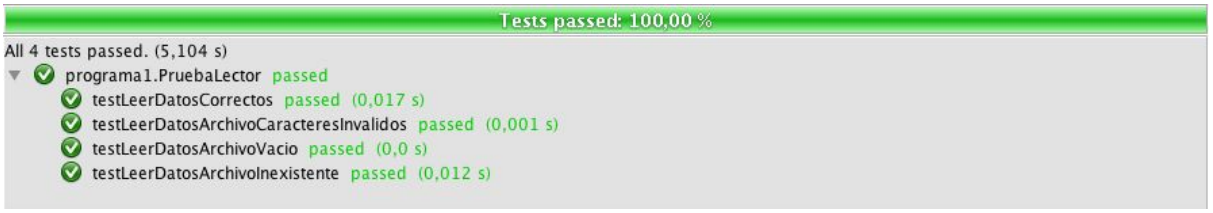
Al ejecutar todas las pruebas se comprueba que todas estas son exitosas, se continúa con el siguiente caso de prueba.

6. leerDatos(testLeerDatosArchivoVacio)

6.1. Implementación prueba y comprobación de que esta falla.

Al ejecutar la prueba con un archivo que vació la prueba falla ya que no se valida antes este este vacío, independientemente de si está vacío o no retorna una lista.

6.2. Implementacion metodo para que la prueba sea exitosa y nueva ejecución de las pruebas.



Al ejecutar todas las pruebas se comprueba que todas estas son exitosas, se continúa con el siguiente caso de prueba.

7. leerDatos(testLeerDatosArchivoUnaColumna)
   7.1. Implementación prueba y comprobación de que esta falla.

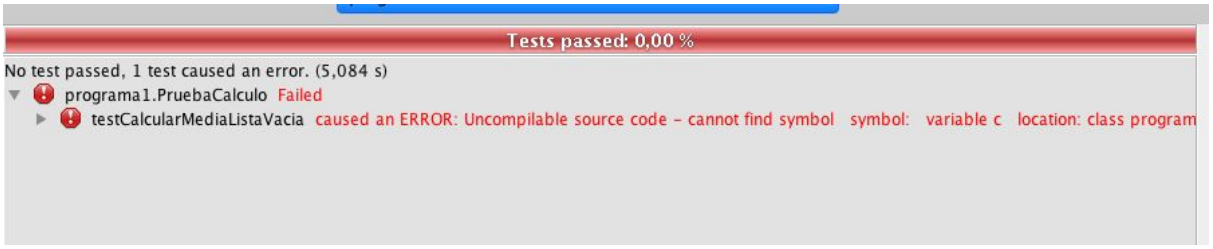
Al ejecutar la prueba se encuentra que esta es exitosa ya que el método valida el formato correcto.

8. leerDatos(testLeerDatosArchivoTresColumnas)
   8.1. Implementación prueba y comprobación de que esta falla.

Al ejecutar la prueba se encuentra que esta es exitosa ya que el método valida el formato correcto.

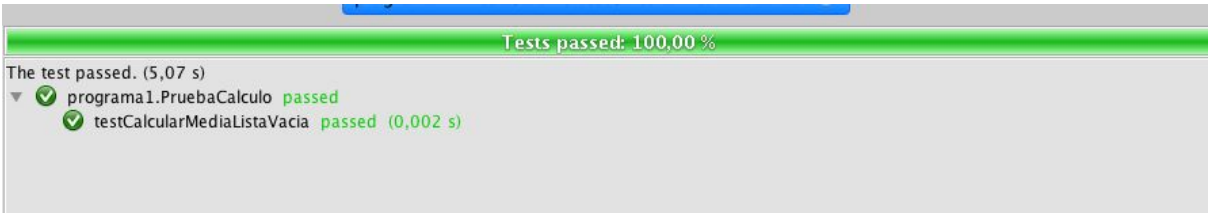
9. calcularMedia(listaVacía)

9.1 Implementación prueba y comprobación de que esta falla.



Al ejecutar la prueba con una lista vacía, la prueba falla ya que no se valida antes que este exista.

9.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.

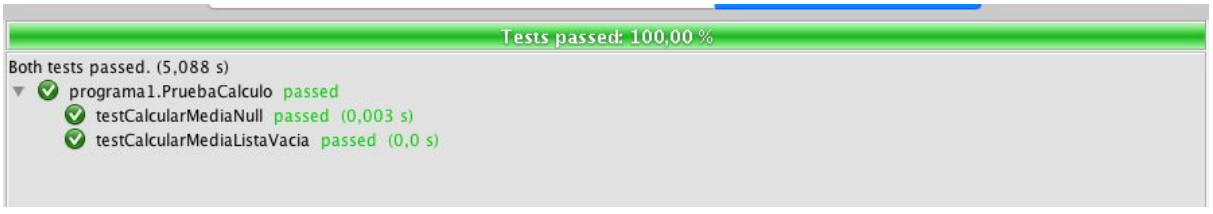


10. calcularMedia(null)

10.1 Implementación prueba y comprobación de que esta falla.

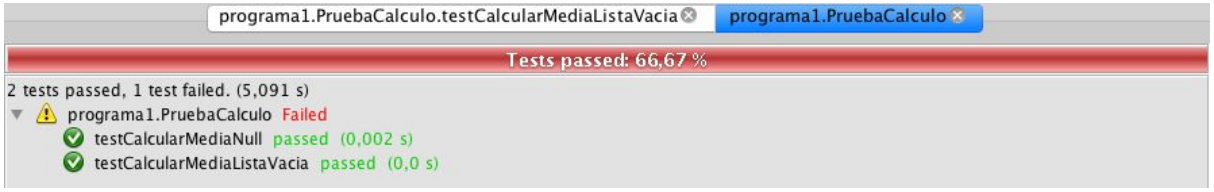
Se puede ver que el primer método pasa, pero el segundo al no ser validado arroja error ya que recibe un objeto null como parámetro.

10.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.



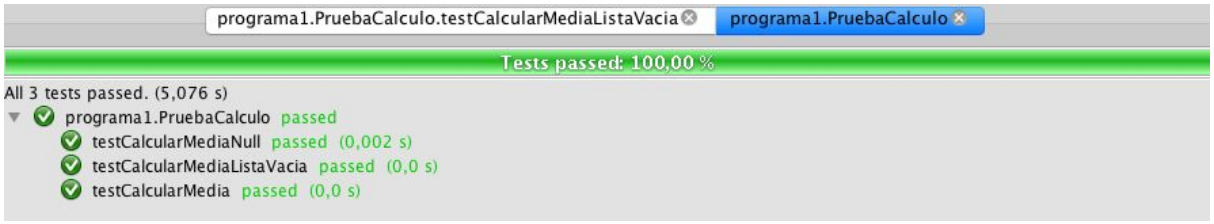
11. calcularMedia(lista)

11.1 Implementación prueba y comprobación de que esta falla.



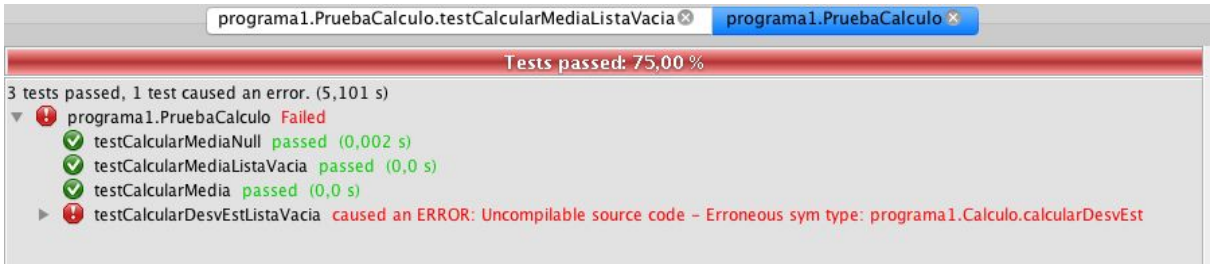
Luego de validar los anteriores casos, se procede a probar el método que arrojará el valor final. Debido a que no se ha implementado, éste arroja error.

11.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.



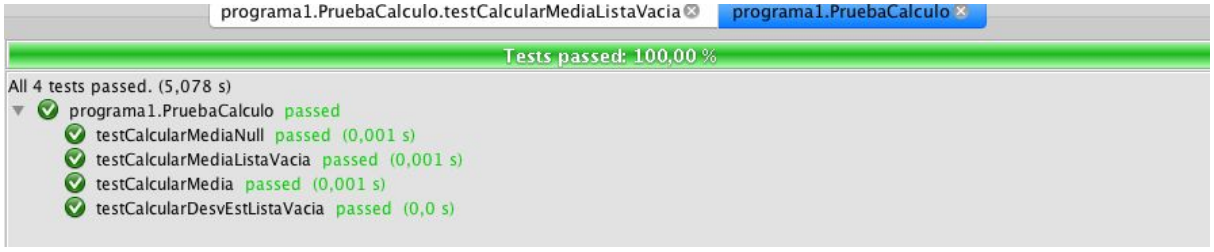
13. calcularDesvEst(listaVacia)

13.1 Implementación prueba y comprobación de que esta falla.



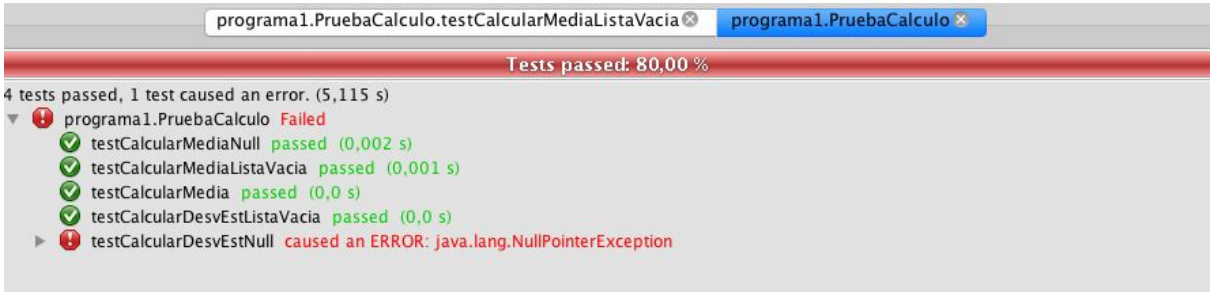
El primer caso de prueba para calcular la desviación estandar. Se mira si la lista que recibe es vacía. Arroja error ya que no se ha validado.

13.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.



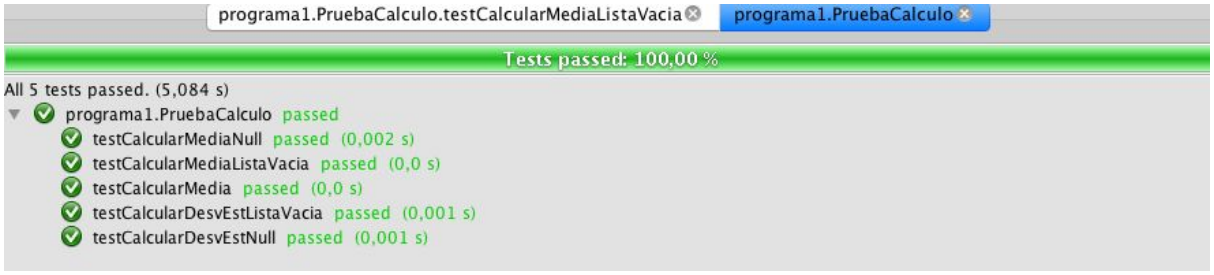
14. calcularDesvEst(null)

14.1 Implementación prueba y comprobación de que esta falla.



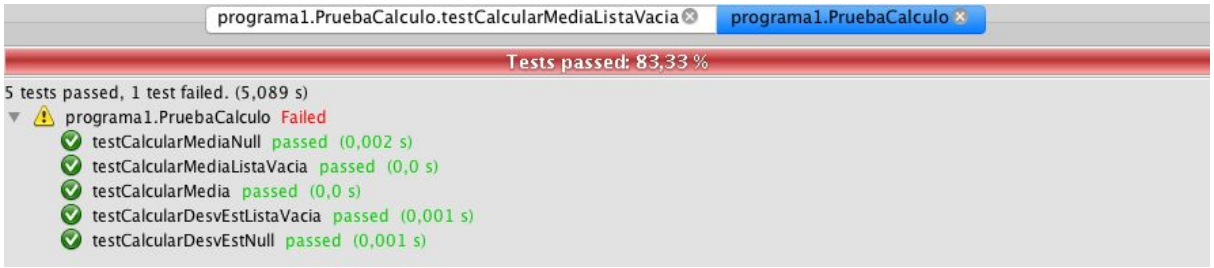
Aunque ya se valida que la lista recibida esté vacía, arroja error debido a que el nuevo caso de prueba, con parámetro null, no se ha implementado.

14.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.



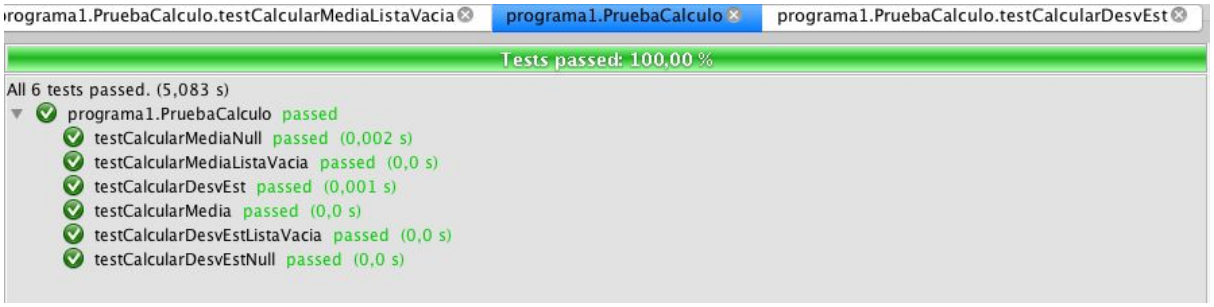
15. calcularDesvEst(lista)

15.1 Implementación prueba y comprobación de que esta falla.



El último caso de prueba es validar si el resultado de la operación es el que se supone correcto inicialmente en el desarrollo.

15.2 Implementación método para que la prueba sea exitosa y nueva ejecución de las pruebas.



Luego de hacer las pruebas e implementar el código necesario para que estas pasen, se comprueba que todo el programa funciona.

Referencias

- Herranz Roldán, José Ignacio. “TDD como metodología de diseño de software”. 17 de Enero del 2011. Madrid, España. Consultado el día 15 de agosto de 2017 en: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/.
- Rafael. Pruebas Unitarias con JUnit. Lunes, 14 Julio 2014. Consultado el dia 15 de agosto de 2017 en: http://www.compujuy.com.ar/blogs/item/16-pruebas-unitarias-con-junit-en-netbeans