

Practica

JUnit

Enero-Febrero 2019

Indice

¿Requisitos del trabajo a realizar?	3
Diseño y realización de pruebas.	3
Metodos a realizar.	3
Análisis y Diseño.	4
Main.	5
Exception.	6
Catalogo de requisitos.	6
Complejidad ciclomática de McCaba.	7
Resultados del modulo 10.	7
Pruebas de caja blanca-Caminos Básicos.	8
Complejidad Ciclomática.	8
Complejidad Ciclomática bien planteado.	9
Pruebas con JUnit	10
Suite.	12
Coverage	13
Plan de Pruebas	14
Reflexiones	15
Java.doc	15

PRUEBAS JUnit

¿Requisitos del trabajo a realizar?

[DISEÑO Y REALIZACIÓN DE PRUEBAS]

-Suponiendo que una empresa requiere de nuestros servicios como desarrolladores de un proyectos, para un fin determina.

-El proyecto de realizara de forma modular, a la hora de la repartición del trabajo se a decido que cada elemento del equipo realice 2 módulos (Procesos).

Para garantizar la corrección del desarrollo,se propone establecer la planificación de las pruebas.

Por lo qué hay que probar el software, seguir un orden concreto en la realización de pruebas.

Requisitos.

- 1.- Hay que programar al menos una clase Java. No se requiere método main
 - 2.- Hay que programar al menos un método en la clase Java
 - 3.- Hay que calcular la complejidad ciclomática del método que se programa
 - 4.- Hay que indicar los casos de prueba
 - 5.- Hay que hacer un plan de prueba, con resultados esperados y resultados obtenidos
 - 6.-Hay que programar con Junit, indicando la asociación con el caso de prueba:
- Al menos 2 test cases
- Al menos 1 test suite
- Se debe controlar que se pueda producir excepción en algún método

METODOS A REALIZAR

Método que calcule la suma de un conjunto de números negativos almacenados en memoria. Deberá de dar excepción si algún número es positivo.
Método que calcule el número primo n. Debe devolver excepción si se le pide un $n > 10$;

ANEXO I

La estructura de la documentación deberá contener al menos los siguientes puntos:

Análisis y Diseño

Definición del sistema

Definición del alcance del sistema

Establecimiento de requisitos

Catálogo de requisitos

Construcción

Complejidad ciclomática de McCabe

Conjunto de caminos básicos

Plan de pruebas

Casos de prueba con resultado esperado

Código fuente junit de cada test case (Caso de pruebas) y test suite

PRUEBAS JUnit

Análisis y Diseño Método 10

Método que calcule el número es o no primo. Debe devolver un mensaje si introducimos un numero mayor que diez.

Para la realización de este proceso debemos tener en cuenta los requisitos que debe cumplir. Entre estos se encuentran.

(Requisitos establecidos)

- El método debe dar un resultado de tipo Boolean. Verdadero o Falso.
- Debe comprobar si es o no es primo.
- En el caso de que el usuario introduzca un numero mayor a 10 el método lanzara un mensaje.(Exception)

Código.

Esta clase tiene 2 metodos, el metodo iniciar() y el metodo esPrimo(), vemos que la clase iniciar tiene un parametro el cual es un numero que sera nuestro limite(hasta que numero quiero contar numeros primos), y el numero esPrimo() tiene un parametro que es un unico numero, el metodo determina si el numero es primo y regresa un valor de "true" o Verdadero si lo es, si el numero no es primo regresa un valor falso (false).

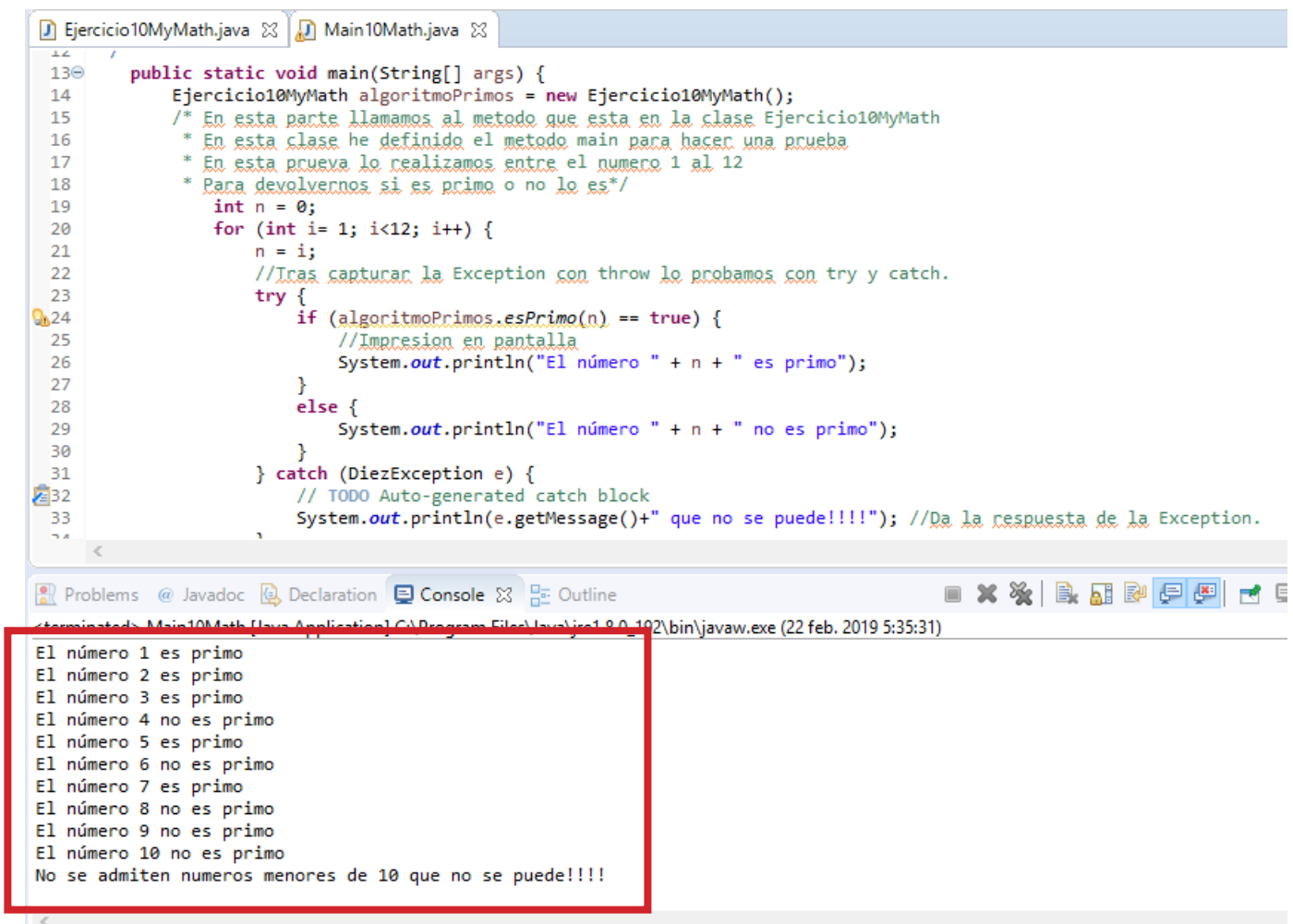
```
Ejercicio10MyMath.java
2 /**
3  * Esta clase permite definir el metodo esPrimo
4  * @author klever
5  *
6  */
7 public class Ejercicio10MyMath {
8     /**
9      * Comenzamos lo que debe realizar el modulo.
10     * En este caso nos piden un método que calcule el número primo
11     * n. Debe devolver excepción si se le pide un n > 10
12     * @throws DiezException
13     * @param n - valor ingresado por el usuario */
14
15     public static boolean esPrimo(int n) throws DiezException {
16         // TODO Auto-generated method stub
17         int aux;
18         if(n>10) {
19             //creamos una Exception para no que se habia requerido
20             throw new DiezException("No se admiten numeros menores de 10");
21         }
22         else {
23             //Esta es la parte de las instrucciones del proceso,
24             //esta parte se vera en profundidad en el PDF.
25             for (int cont=2;cont<n;cont++) {
26                 aux=n%cont;
27                 if(aux==0)
28                     return false; //respuesta del subprocesso
29             }
30             return true; //respuesta del subprocesso
31         }
32     }
33 }
34 //Para ver el resultado del Metodo voy a un método main que lo llamare Main10Math(Clase)
35 }
```

Propuesta de código.

PRUEBAS JUnit

MAIN.

Aunque no es imprescindible a la hora de realizar pruebas el método main, dado que se realizara con JUnit , he decido crearlo de todas formas, para poder así verificar su funcionamiento antes de hacer pruebas unitarias.



The screenshot shows an IDE with two tabs: 'Ejercicio10MyMath.java' and 'Main10Math.java'. The 'Main10Math.java' tab is active, displaying the following Java code:

```
13 public static void main(String[] args) {
14     Ejercicio10MyMath algoritmoPrimos = new Ejercicio10MyMath();
15     /* En esta parte llamamos al metodo que esta en la clase Ejercicio10MyMath
16      * En esta clase he definido el metodo main para hacer una prueba
17      * En esta prueba lo realizamos entre el numero 1 al 12
18      * Para devolvernos si es primo o no lo es*/
19     int n = 0;
20     for (int i = 1; i<12; i++) {
21         n = i;
22         //Tras capturar la Exception con throw lo probamos con try y catch.
23         try {
24             if (algoritmoPrimos.esPrimo(n) == true) {
25                 //Impresion en pantalla
26                 System.out.println("El número " + n + " es primo");
27             }
28             else {
29                 System.out.println("El número " + n + " no es primo");
30             }
31         } catch (DiezException e) {
32             // TODO Auto-generated catch block
33             System.out.println(e.getMessage()+" que no se puede!!!!"); //Da la respuesta de la Exception.
34         }
35     }
36 }
```

Below the code editor, the 'Console' tab is active, showing the output of the program:

```
El número 1 es primo
El número 2 es primo
El número 3 es primo
El número 4 no es primo
El número 5 es primo
El número 6 no es primo
El número 7 es primo
El número 8 no es primo
El número 9 no es primo
El número 10 no es primo
No se admiten numeros menores de 10 que no se puede!!!!
```

En este RUN se están pasando números del 1 al 11 dando así los diferentes resultados esperados. Incluida la Exception.

PRUEBAS JUnit

Exception

Para poder crear una exception, se a tenido que crear una clase aparte, en mi Proyecto lo de definido de la siguiente forma

```
15- public static boolean esPrimo(int n) throws DiezException {
16-     // TODO Auto-generated method stub
17-     int aux;
18-     if(n>10) {
19-         //creamo una Exception para no que se habia requerido
20-         throw new DiezException("No se admiten numeros menores de 10");
21-     }
22- }
```

En la clase del metodo

```
Ejercicio10MyMath.java  Main10Math.java  DiezException.java
1 package com.islas.filipinas.entornos.trabajos;
2 /**
3  * En esta clase se define la clase Exception del ejercicio 10
4  * @author klever
5  *
6  */
7 public class DiezException extends Exception {
8     public DiezException(String m) {
9         super(m);
10    }
11 }
12 }
```

Clase Exception

Catálogo de requisitos

Lo mas básico del método se deben de cumplir .

Decir si un numero es primo
Decir si un numero no es primo
Exception (si hay alguno numero mayor a 10)

En resumen este método debe poder decirte si un numero es o no es primo.

PRUEBAS JUnit

Complejidad ciclomática de McCabe

Este es el apartado donde mas tiempo le di, dado que, quería conseguirlo de una forma que se vio en clase .

Utilizando un software en especifico que me ayude en esta labor. Ademas de Hacerlo de forma normal. En un principio lo estaba intentando con Visual Studio pero no encontraba la manera ni el pluing necesario.

Busque en muchos foros leí muchos post pero nada me servia.

Hasta que por allí en una perfil de github di con mi respuesta, se trata de un programa que ya tiene muchos años llamado R esource S tandard M etrics de Msquare Technologies, genera un archivo html similar a javadoc

<http://msquaredtechnologies.com/RSM-Wizard.html>

Resultados del Modulo 10

Función : `com.islas.filipinas.entornos.trabajos.Ejercicio10MyMath.esPrim`
o

Parámetros : `(int n)`

Complejidad Ciclométrica Vg Detalle

Base de funciones: 1

Bucles para / foreach: 1

Condicional si / else if: 2

Complejidad	Parámetros 1	Retorno 2	Cyclo Vg 4	Total	7
LOC 14	eLOC 10	lLOC 6	Comentario 5	Líneas	18

~~ Total File Summary ~~

LOC 17	eLOC 12	lLOC 7	Comment 17	Lines	34
--------	---------	--------	------------	-------	----

~~ Resumen Funcional de Archivo ~~

Cantidad de Funciones de Archivo 1

Función Total LOC 14 Función Total Pts LOC: 0.3

Función Total eLOC 10 Función Total Pts eLOC: 0.2

Función Total lLOC 6 Pts de la función total lLOC: 0.1

Parámetros de la función total: 1 Retorno de la función total.: 2

Complejidad total del ciclo: 4 Complejo de la función total .. 7

Función máx. LOC 14 Función prom. LOC ..: 14.00

Función Max eLOC 10 Función promedio eLOC.: 10.00

Función Max lLOC 6 Función promedio lLOC.: 6.00

Número máximo de parámetros de función: 1 Prom parámetros de la función: 1.00

Función Max .. Returns: 2 Prom devuelve la función ..: 2.00

Complejo Max interfaz. : 3 Avg Interface Complex. : 3.00

Max Cyclomatic Complex ..: 4 Avg Cyclomatic Complex ..: 4.00

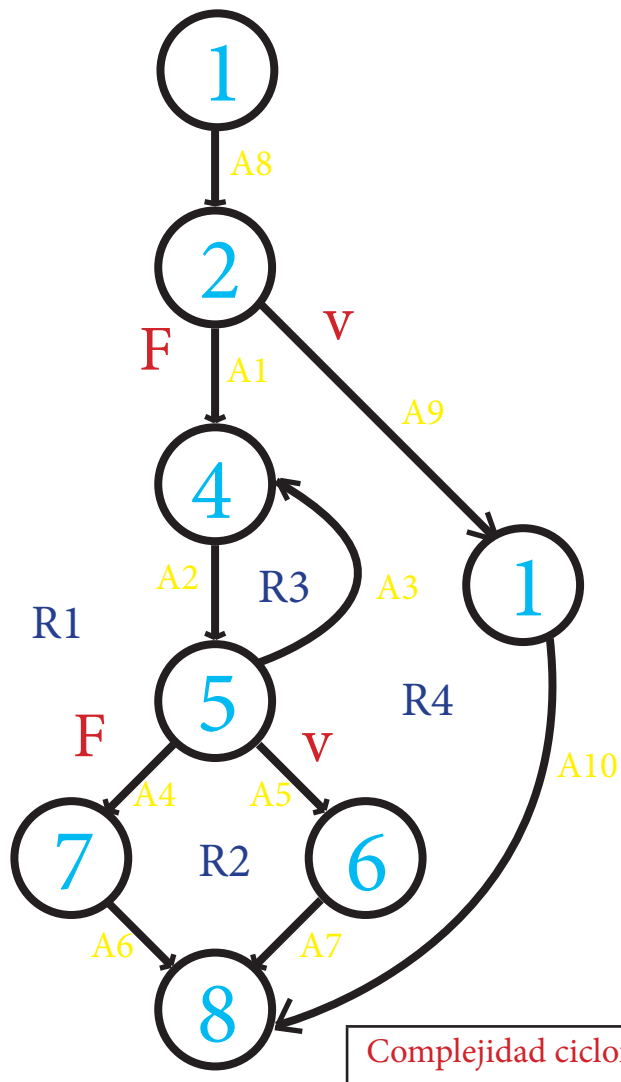
Max Total Complexity: 7 Avg Total Complexity ..: 7.00

Fin del archivo: [E: \ trabajo de Junit \ proyecto \ KleverUyana_Junit \ src \ com \ isla s \ filipinas \ entornos \ trabajos \ Ejercicio10MyMath.java](#)

PRUEBAS JUnit

Complejidad ciclomática de McCabe Modulo 10

En un principio la complejidad ciclomática que yo había realizado tenia esta forma.



Aristas=10
Regiones=4
Nodos=8

1

V(G)	Aristas-Nodos+2
V(G)	10-8+2=4

2

V(G)	Nodos predicado+1
V(G)	2+1=3

4

V(G)	Regiones
V(G)	4

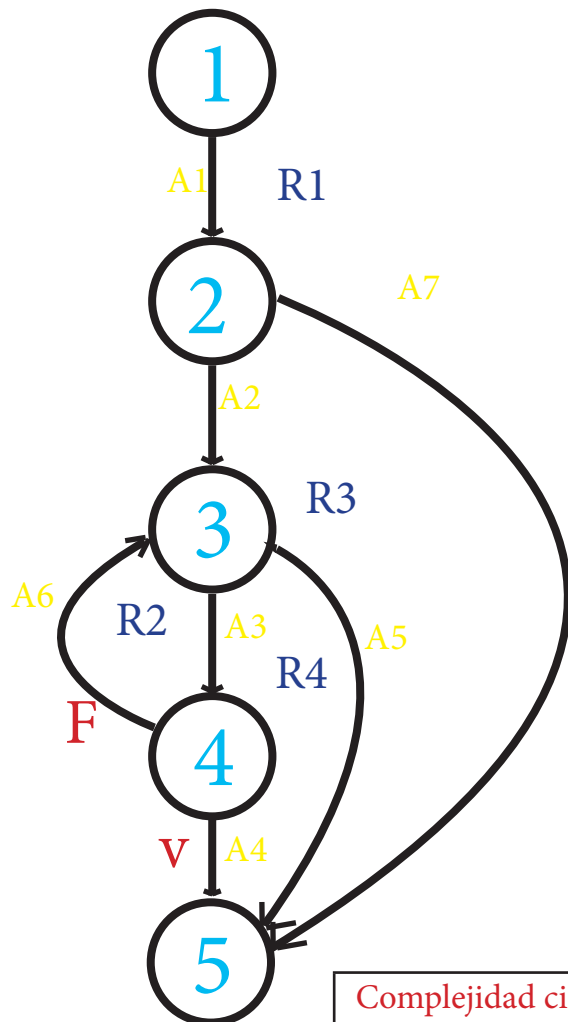
Complejidad ciclomática = 4 Programa Sencillo sin mucho riesgo.

Pruebas de caja Blanca - Caminos Básicos

Camino	Entrada	Salida
1-2-4-5-7-8	b=F ,b=F	No es Primo
1-2-3-4-6	a=F ,b=V	Es primo
1-2-3-8	a=V	Exception

PRUEBAS JUnit

Complejidad ciclomática de McCabe Modulo 9 bien planteado.



Aristas=7
Regiones=4
Nodos=5

1

V(G)	Aristas-Nodos+2
V(G)	7-5+2=4

2

V(G)	Nodos predicado+1
V(G)	3+1=4

4

V(G)	Regiones
V(G)	4

Complejidad ciclomática = 4	Programa Sencillo sin mucho riesgo.
-----------------------------	-------------------------------------

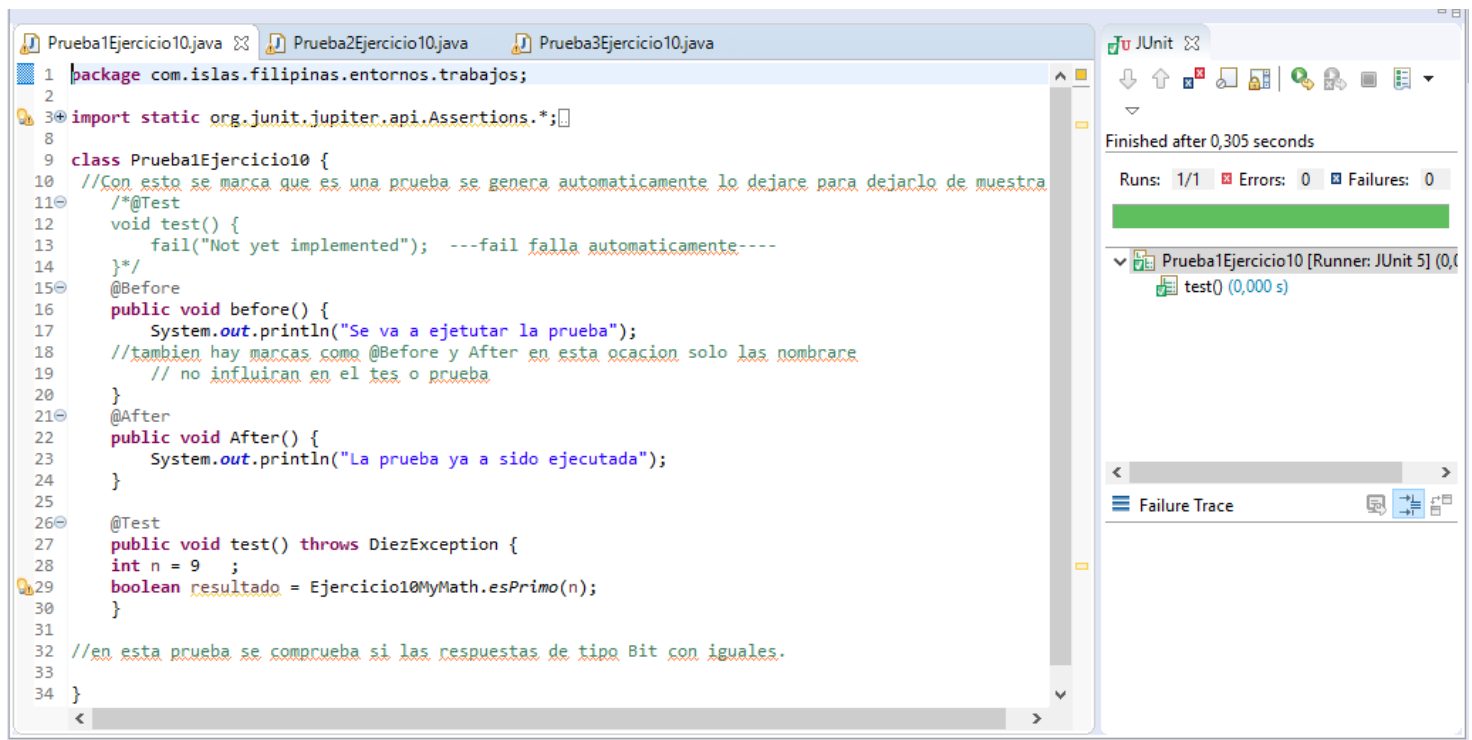
Pruebas de caja Blanca - Caminos Básicos

Camino	Entrada	Salida
1-2-5	a=V	Exception
1-2-3-4-5	a=F, b=V	V= pero Return false
1-2-3-4-3-5	a=F, b=F	F= pero Return True
1-2-3-5	a=F? Es como dar una respuesta sin comprobar no tiene sentido.	En este camino lo que hace es que si a inicio de FOR (i) le damos un valor de 5, al ser mayor que lo que hemos puesto no va entrar en el FOR.

PRUEBAS JUnit

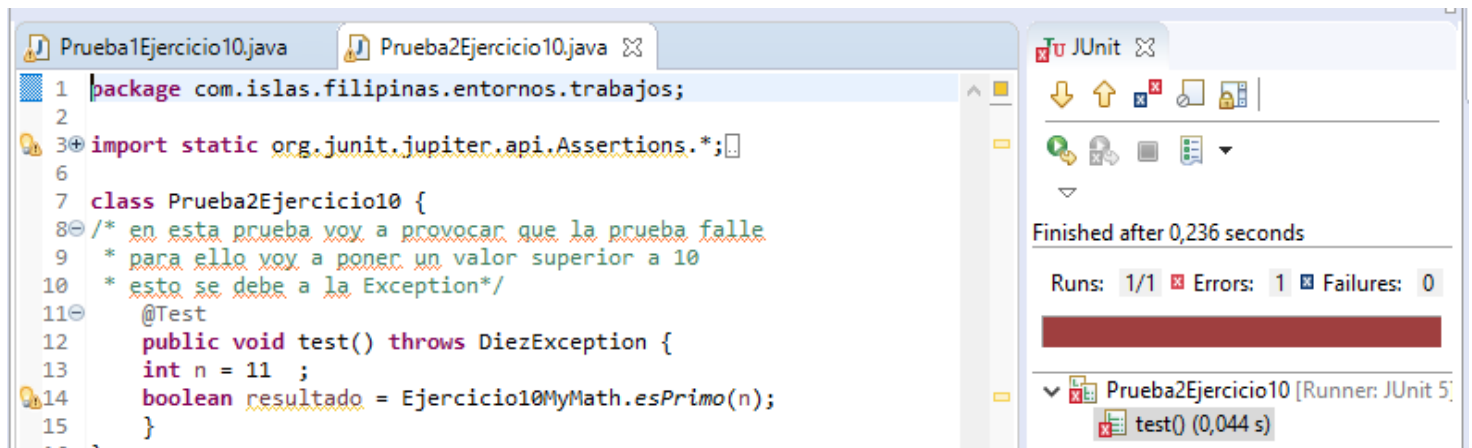
Pruebas con JUnit.

En estas pruebas, quise realizar ciertas pruebas para ver que se puede realizar con la biblioteca de JUnit.



Utilice las etiquetas `@Before` y `@After` para probarlo, en este caso solo las utilice para imprimir cadenas. Sin ninguna fusión en la prueba.

Di como dato `n=9`, que es primo por lo tanto la prueba es correcta

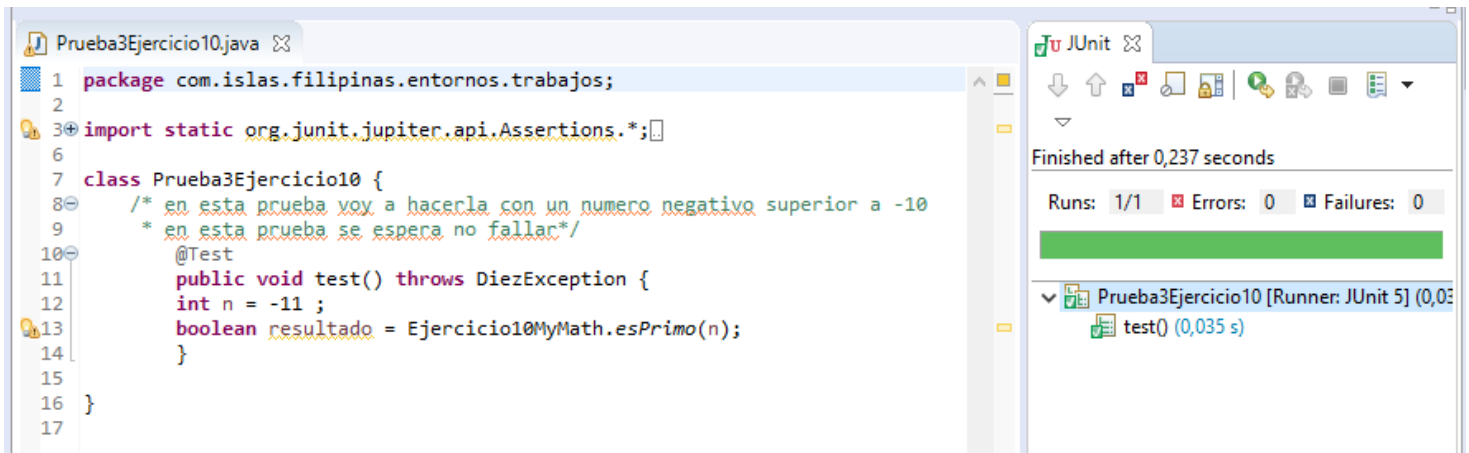


En este caso se realizó la prueba con un número mayor que 10 no primo

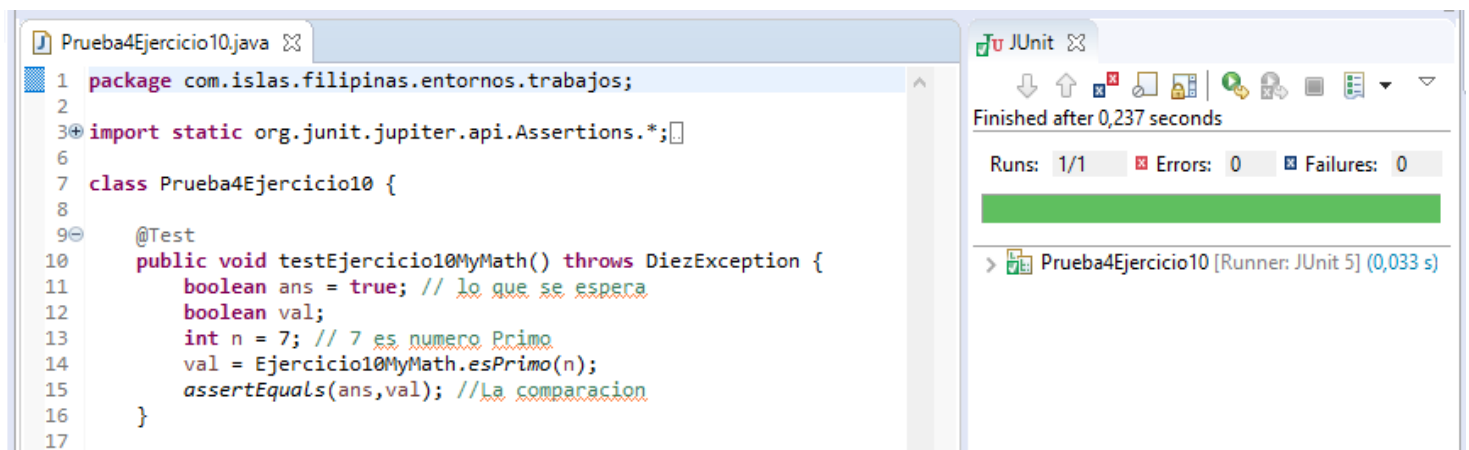
El resultado incorrecto era obvio, dado que no está declarada la `exception.class` en el test.

PRUEBAS JUnit

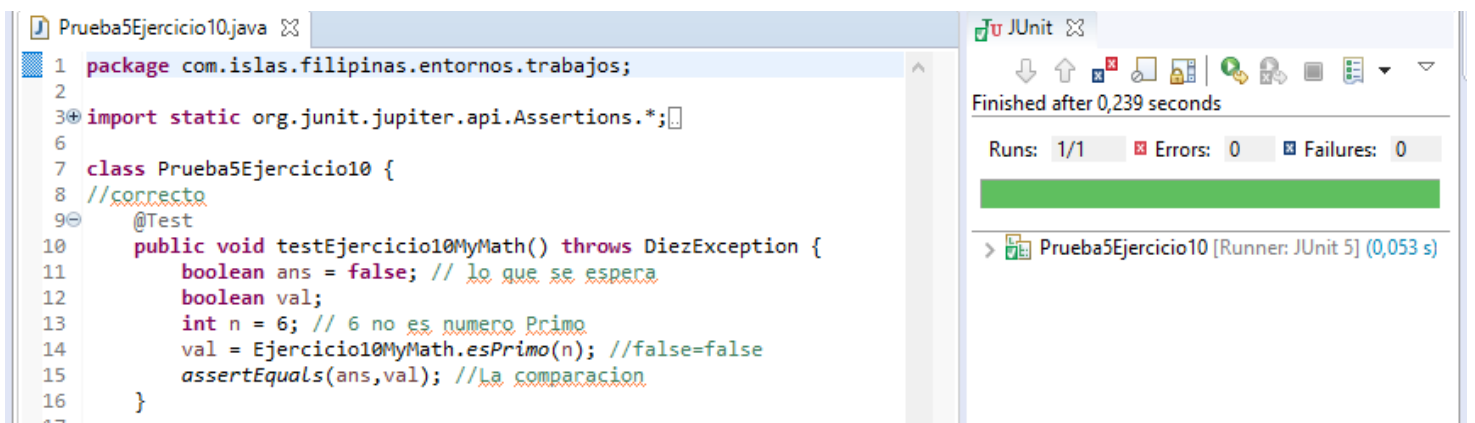
Pruebas con JUnit.



Aquí se metido en la prueba el numero con numero negativo (-11), en esta prueba se salta el FOR y dice que es primo aun que no lo sea. este es el camino 4.



En esta Prueba se añadido lo que se espera y no comparamos con lo que nos da el método, con ayuda de `assertEquals`. En este caso esta todo bien.



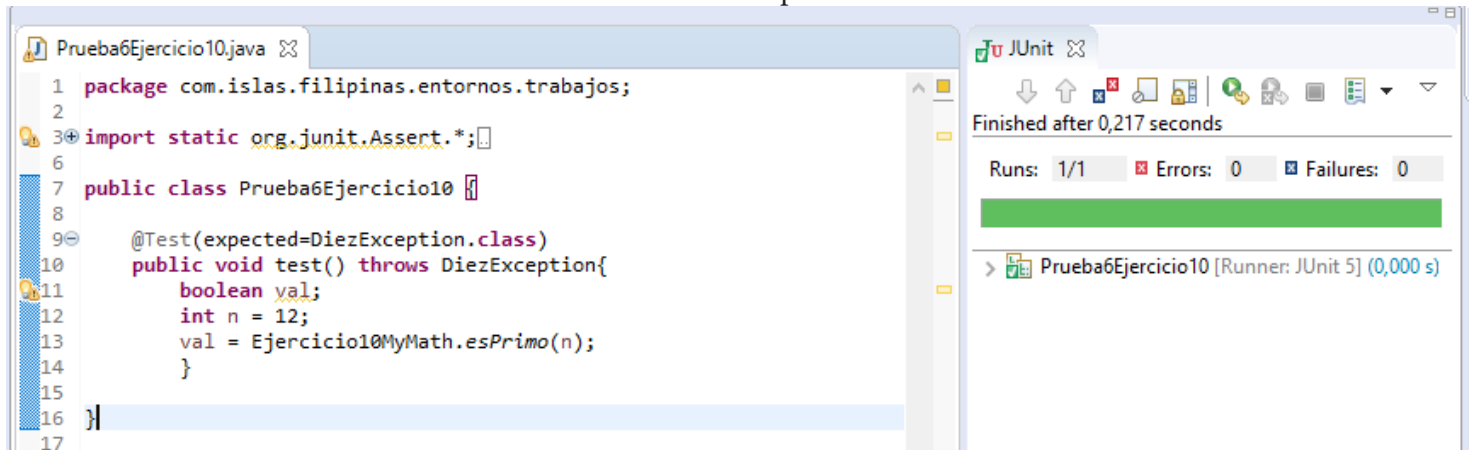
`assertEquals`. En este caso esta todo bien.

No primo = No Primo

PRUEBAS JUnit

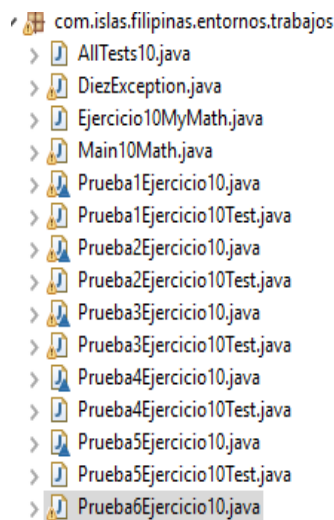
Pruebas con JUnit.

Prueba Exception

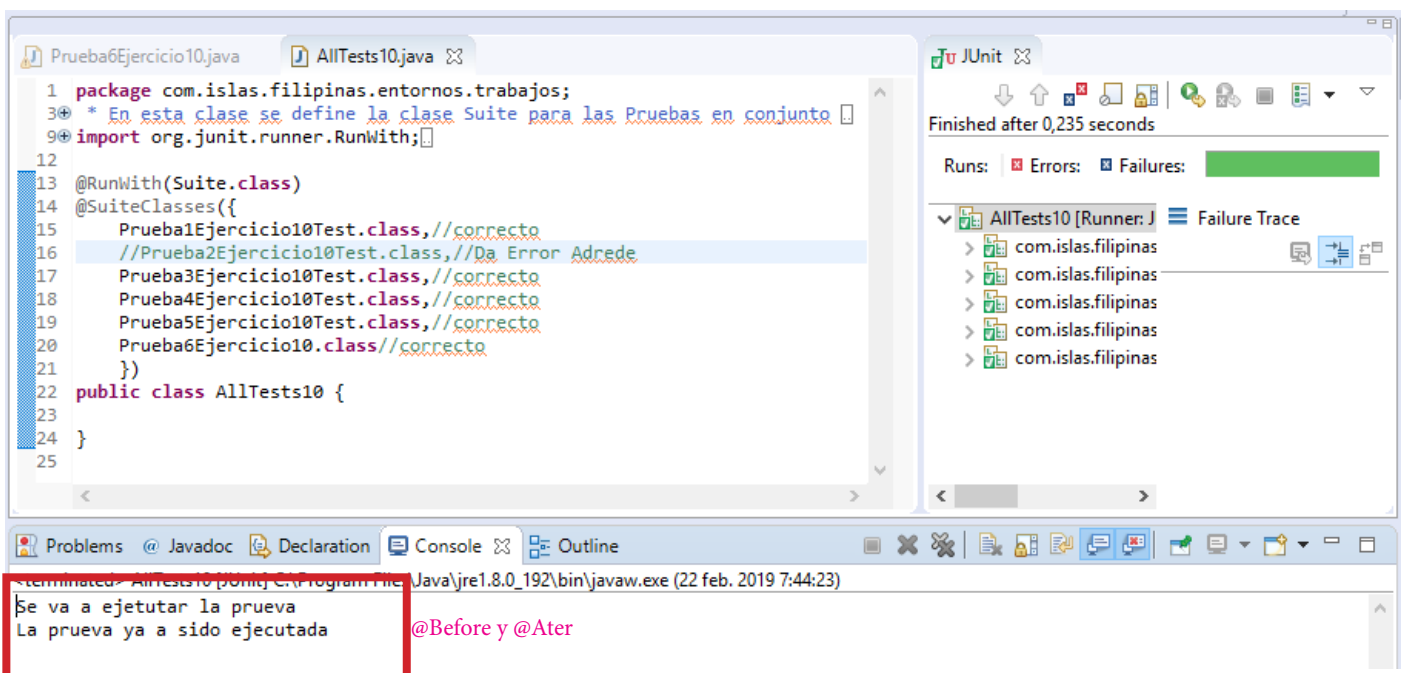


Pruebas con SUITE.

Dado que el Suit solo lo realiza Junit4, y las pruebas(1,2) la había hecho en Junit5 tuve que volver a realizarlas



Como se había esperado, las pruebas conjuntas estaban bien ejecutadas.



PRUEBAS JUnit

CAVERAGE

TODO BIEN, SIN PROBLEMA.

The screenshot displays an IDE with a Java source file `Ejercicio10MyMath.java` and its JUnit coverage report. The code implements a method `esPrimo` that checks if a number `n` is prime. It throws a `DiezException` if `n` is less than 10. The coverage report shows that the code is 89.3% covered, with 25 covered instructions, 3 missed instructions, and a total of 28 instructions.

```
11  * n. Debe devolver excepci3n si se le pide un n > 10
12  * @throws DiezException
13  * @param n - valor ingresado por el usuario */
14
15  public static boolean esPrimo(int n) throws DiezException {
16      // TODO Auto-generated method stub
17      int aux;
18      if(n>10) {
19          //creamo una Exception para no que se habia requerido
20          throw new DiezException("No se admiten numeros menores de 10");
21      }
22      else {
23          //Esta es la parte de las instrucciones del proceso,
24          //esta parte se vera en profundidad en el PDF.
25          for (int cont=2;cont<n;cont++) {
26              aux=n%cont;
27              if(aux==0)
28                  return false; //respuesta del subprocesso
29          }
30          return true; //respuesta del subprocesso
31      }
32  }
33  }
34  //Para ver el resultado del Metodo voy a un m3todo main que lo llamare Main10Math(Clase)
35  }
```

JUnit Coverage Report:

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
> Prueba6Ejercicio10.java	55,6 %	5	4	9
> Ejercicio10MyMath.java	89,3 %	25	3	28
> DiezException.java	100,0 %	4	0	4
> Prueba1Ejercicio10Test.java	100,0 %	17	0	17
> Prueba3Ejercicio10Test.java	100,0 %	9	0	9
> Prueba4Ejercicio10Test.java	100,0 %	16	0	16
> Prueba5Ejercicio10Test.java	100,0 %	16	0	16

PRUEBAS JUnit

Plan de pruebas

Para poder realizar el plan de pruebas tuve que tomar como referencia los caminos de pruebas de caja Blanca o caminos básicos.

Pruebas de caja Blanca - Caminos Básicos

Camino	Entrada	Salida
1-2-5	a=V	Exception
1-2-3-4-5	a=F, b=V	V= pero Return false
1-2-3-4-3-5	a=F, b=F	F= pero Return True
1-2-3-5 1ª	a=F? Es como dar una respuesta sin comprobar no tiene sentido.	En este camino lo que hace es que si a inicio de FOR (i) le damos un valor de 5, al ser mayor que lo que hemos puesto no va entrar en el FOR.

NªPrueba	Nombre de Prueba	Datos dados	Resultados Esperados	Resultados Optenidos
1ª	Prueba3Ejercicio10	9	F= pero Return True	F= pero Return True
2ª	Prueba4Ejercicio10	7	F= pero Return True	F= pero Return True
3ª	Prueba5Ejercicio10	6	V= pero Return false	V= pero Return false
4ª	Prueba6Ejercicio10	11	EXception	Exception

Runs: 5/5 ✖ Errors: 0 ✖ Failures: 0

```
▼ AllTests10 [Runner: JUnit 5] (0,053 s)
  > com.islas.filipinas.entornos.trabajos.Prueba1Ejercicio10Test (0,024 s)
  ▼ com.islas.filipinas.entornos.trabajos.Prueba3Ejercicio10Test (0,002 s)
    test (0,002 s)
  ▼ com.islas.filipinas.entornos.trabajos.Prueba4Ejercicio10Test (0,022 s)
    testEjercicio10MyMath (0,022 s)
  ▼ com.islas.filipinas.entornos.trabajos.Prueba5Ejercicio10Test (0,001 s)
    testEjercicio10MyMath (0,001 s)
  ▼ com.islas.filipinas.entornos.trabajos.Prueba6Ejercicio10 (0,003 s)
    test (0,003 s)
```

PRUEBAS JUnit

Reflexiones

En el segundo ejercicio ya con mas experiencia ya he podido realizar las pruebas con mas soltura.
Entregare La complejidad realizada matemáticamente en el trabajo, en forma de html.

JAVA.DOC

En Clase se trato este tema, yo lo intente generar con éxito.
Por lo que lo voy a incluir en el trabajo en una carpeta llamada Documentación.
para profundizar en el se tiene que acceder desde un navegador
Tiene esta pinta.

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

KleverUyana

Packages

Package	Description
com.islas.filipinas.entornos.trabajo9	
com.islas.filipinas.entornos.trabajos	

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Class Summary	
Class	Description
AllTests9	
Ejercicio9MyMath	Classe que contiene el metodo de suma de numeros negativos
Main9Math	Clase prinsipal del sistema ejercicio 9

Exception Summary	
Exception	Description
NuevaException	En esta clase se define la clase Exception del ejercicio 9



Class Summary	
Class	Description
AllTests10	
Ejercicio10MyMath	Esta clase permite definir el metodo esPrimo
Main10Math	Clase prinsipal del sistema ejercicio 10
Prueba4Ejercicio10Test	

Exception Summary	
Exception	Description
DiesException	En esta clase se define la clase Exception del ejercicio 10