**Laboratorio 02. POOB-02**

Jefer Alexis González Romero  
Angel Nicolas Cuervo Naranjo

**Conociendo el proyecto**

**1.** El proyecto BlueJ “polynomialCalculator” contiene una construcción parcial del sistema. Revisen el directorio donde se encuentra el proyecto. Describan el contenido considerando los directorios y las extensiones de los archivos.

Dentro del directorio *polynomialCalculator* están las clases que actualmente hay, junto con el paquete de *bluej* y el archivo de documentación *README*, las extensiones que hay para lo anterior son: *.class* que contiene código *bytecode* en su forma binaria, *.ctxt* que contiene información extra para una clase*, .java* que contiene el código fuente escrito en lenguaje de programación java, *.bluej* que contiene el paquete de *BlueJ*, y *.TXT* que contiene la información sobre el proyecto.

En el directorio *doc* se encuentra la documentación del proyecto, tiene archivos con extensiones *.html, .js, .css, .zip.* y tiene otros dos directorios uno llamado *jquery* con .js y *.css,* el otro *resources* con *.png*.

**2.** Exploren el proyecto en BlueJ ¿Cuántas clases tiene? ¿Cuál es la relación entre ellas? ¿Cuál es la clase principal para la aplicación? ¿Cómo la reconocen? ¿Cuáles son las clases “diferentes”? ¿Cuál es su propósito?

**a)** Tiene 3 clases: *PolynomialCalculator, Polynomial, Fraction y FractionTest.*

**b)** La relación entre estas clases es que *PolynomialCalculator* usa de *Polynomial* para su atributo *variables* y *FractionTest* usa de *Fraction* para las pruebas.

**c)** La clase principal es *PolynomialCalculator.*

**d)** La reconocemos debido a que es la encargada de realizar las operaciones a fin al propósito de la aplicación.

**e)** La clase diferente es *FractionTest.*

**f)** Su propósito es probar el funcionamiento de la clase *Fraction.*

**3.** Generen y revisen la documentación del proyecto: ¿está completa la documentación de cada clase? (Detallen el estado de documentación de cada clase: encabezado y métodos)

No está completa la documentación, debido a que las clases *Polynomial y PolynomialCalculator* no tienen descripción, sus métodos no tienen descripción, no tienen los parámetros que reciben y lo que retorna cada uno.

**4**. Revisen las fuentes del proyecto, ¿en qué estado está cada clase? (Detallen el estado de las fuentes considerando dos dimensiones: la primera, atributos y métodos, y la segunda, código, documentación y comentarios) ¿Qué son el código, la documentación y los comentarios?

***Fraction***

No tiene atributos, tiene métodos, pero sin su codificación para que cumplan con lo que debe hacer cada uno, todos los métodos tienen la documentación con los parámetros y lo que debe retornar.

***Polynomial***

No tiene documentación, atributos y solo tiene un método, que es el constructor, pero este no está codificado.

***Polinomialcalculator***

Solo tiene un atributo, los métodos no realizan ninguna función, tiene cometarios, pero no tiene la documentación de los métodos y la clase.

**Código:** son las instrucciones ordenadas de un programa, lo que determina como se usa y manipula la información en búsqueda de realizar una tarea en específico.

**Documentación:** es una descripción detallada del código de nuestro programa. Que nos habla sobre las diferentes implementaciones de un programa, parámetros y funciones dentro de la implementación.

C**omentarios:** son líneas en el código que nos ayudan a describir en específico que hace una instrucción en concreto. Muy útiles a la hora de cubrir lo que la documentación no puede detallar.

**Ingeniería reversa**

**MDD MODEL DRIVEN DEVELOPMENT**

**1.** Complete el diagrama de clases correspondiente al proyecto (No incluya la clase de pruebas)

En el archivo PolynomialCalculator.astah

**2.** ¿Cuál nuevo contenedor está definido? Consulte la especificación y el API Java ¿Qué diferencias hay entre el nuevo contenedor y el ArrayList que conocemos?

El nuevo contenedor es *HashMap,* la diferencia con *ArrayList* es que sus objetos están ordenados de manera secuencial, en cambio el nuevo contenedor no y en este último se almacenan parejas llave-valor.

**Conociendo Pruebas en BlueJ**

**De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)**

Para poder cumplir con las prácticas XP vamos a aprender a realizar las pruebas de unidad usando las herramientas apropiadas. Para eso consideraremos implementaremos algunos métodos en la clase FractionTest

**1.** Revisen el código de la clase FractionTest. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuantos métodos son de prueba? ¿cómo los reconocen?

**a)** Solo se tiene una etiqueta que es @Test.

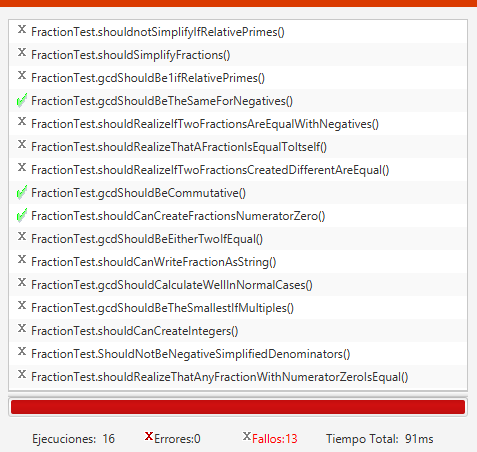
**b)** Tiene 16 métodos.

**c)** Los 16 métodos son de prueba.

**d)** Se reconocen porque antes de la declaración del método está la etiqueta @Test.

**2.** Ejecuten los tests de la clase FractionTest. (click derecho sobre la clase, Test All) ¿cuántas pruebas se ejecutan? ¿cuántas pasan? ¿por qué?

Se ejecutan 16 pruebas de las cuales solo pasan 3. Esto se debe a que los que pasaron están verificando que den el mismo valor al llamar en ambos casos el método *cgd* de clase *Fraction* con unos parámetros establecidos, y por ahora este método retorna 0 sin importar los parámetros que se ingresen. En los otros casos se compara un valor especifico con el retorno del método gcd y esto provoca que rechace la prueba.



**3.** Estudie las etiquetas encontradas en 1. Expliquen en sus palabras su significado.

@Test Describe a los métodos asociados a las pruebas que se le realizaran a una clase en concreto. Enviando diferentes valores y evaluando casos de prueba para verificar que nuestro código y programa funcionan.

**4.** Estudie los métodos assertTrue, assertFalse, assertEquals, assertArrayEquals, assertNull y fail de la clase assert del API JUnit 2 . Explique en sus palabras que hace cada uno de ellos.

**asserTrue:** Revisa si una condición es verdadera.

**assertFalse:** Verifica si una condición es falsa.

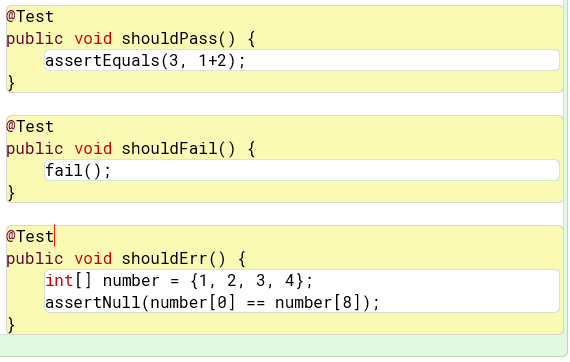
**assertEquals:** Verifica que dos objetos sean iguales.

**asserNull:** Este método valida que un objeto sea nulo.

**fail:** Hace que una prueba no sea aprobada.

**5.** Investiguen la diferencia que entre un fallo y un error en Junit. Escriba código, usando los métodos del punto 4., para lograr que los siguientes tres casos de prueba se comporten como lo prometen shouldPass, shouldFail, shouldErr.

Un fallose produce en Junit cuando alguno de los casos de prueba con los que estamos probando alguna clase/objeto arrojan un resultado contrario al esperado por el método de prueba. Por otro lado, los errores son discrepancias que se presentan al ejecutar nuestras pruebas.



**Prácticando Pruebas en BlueJ**

**De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)**

Ahora vamos a escribir el código necesario para que las pruebas de FractionTest pasen.

**1.** Determinen los atributos de la clase Fraction. Justifique la selección.

Para la clase *Fraction* se usarán dos atributos de tipo entero, uno en el que guardaremos el valor del numerador y otro para denominador. Esto porque para formar la fracción solo se necesitan esos valores.

**2.** Revise el invariante de la clase Fraction. Justifique la decisión.

La invariante es que siempre la fracción va a estar en su forma más simplificada, debido a que el máximo común divisor es usado para la simplificación de fraccionarios y la clase pide que se represente en formato simplificado.

**3.** Implementen únicamente los métodos de Fraction necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?

Se implementaron:

* gcd(int a, int b)
* numerator()
* denominator()
* fraction(int numerator, int denominator)
* fracton(int integer)
* toString()
* equals(Fraction other)

**Desarrollando PolynomialCalculator**

**BDD – MDD**

|  |  |  |
| --- | --- | --- |
| **Ciclo** | **PolynomialCalculator** | **PloynomialCalculatorTest** |
| 1 | PolynomialCalculator()  create(String nombre)  assign(String set, int fraction[][])  evaluate(String a, int[] x)  toString(String set)  ok()  getVariables() | shouldCanCreateVariables()  shouldCanAssignPolynomialsWithArraysOfIntegers()  shouldCanEvaluatePolynomials()  shouldNotEvaluatePolynomialsToNonexistentVariables()  shouldCanWritePolynomialsAsString()  shouldNotWritePolynomialsToNonexistentVariables()  shouldCanSayIfTheLatOperationWasSccesfull()  shouldCanGetAllVariables() |
| 2 | assign(String a, String b, char op String c)  add(Polynomial a, Polynomial b)  substract(Polynomia a, Polynomial b) | shouldCanAssignTheValueOfASum()  shouldCanAssignTheValueOfASubstraction() |
| 3 | multiply(Polynomia a, Polynomial b)  divide(Polynomia a, Polynomial b) | shouldCanAssignTheValueOfAMultiplication()  shouldCanAssignTheValueOfADivision() |
| 4 | assign(String a, char op, String b)  derivate(Polynomia a)  integrate(Polynomia a) | shouldCanAssignTheValueOfADerivation()  shouldCanAssignTheValueOfAIntegration() |
| 5 | assign(String a, char op, String b, int n) | shouldCanAssignTheValueOfAMultipleDerivation()  shouldCanAssignTheValueOfAMultipleIntegration() |

**RETROSPECTIVA**

**1**. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)

10 horas por cada persona.

**2.** ¿Cuál es el estado actual del laboratorio? ¿Por qué?

El laboratorio está totalmente terminado.

**3.** Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

Nos resultó más útil la programación de las pruebas primero, al hacer primero las pruebas podíamos ver que no estaba funcionando y lo arreglábamos, y no teníamos que ir haciendo nuevas pruebas cada vez que lo necesitáramos.

**4.** ¿Cuál consideran fue el mayor logro? ¿Por qué?

Hacer el método de división, ya que, nos falló mucho en el momento de su realización.

**5.** ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

Los diagramas de secuencias, ya que no sabíamos como realizarlos. Para resolverlo averiguamos como se hacían y así pudimos hacerlos.

**6.** ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

La comunicación fue clave para la elaboración de este laboratorio. Nos comprometemos a dedicarle más tiempo al laboratorio, elaborándolo con más anticipación y a mejorar el nivel de conocimiento del lenguaje.