LABORATORIO #2: INTERACCIÓN ENTRE OBJETOS

A. CONOCIENDO EL PROYECTO

1. Revisando el directorio

Revisando el contenido nos dimos cuenta de que existen tres clases (Set, SetCalculator, SetTest), una crea una calculadora y otra clase realiza test sobre la calculadora; además de esto ya hay código ya escrito.

2. Explorando el proyecto

¿Cuántas clases tiene?

El proyecto posee tres clases.

¿Cuál es la relación entre ellas?

Clase utiliza a →

SetCalculator → Set

SetTest → Set

¿Cuál es la clase principal?

La clase principal es SetCalculator.

¿Cómo la reconocen?

Sin esta clase la implementación de la calculadora no se puede realizar.

¿Cuáles son las clases "diferentes"?

La clase diferente es la de SetTest.

¿Cuál es su propósito?

Hacer casos de prueba sobre la calculadora.

3. Generando y revisando documentación

Documentación Class Set: No proporciona una descripción, versión de código ni autor. En cuanto a los métodos, no tienen la descripción de su propósito. Para los métodos que tienen parámetros, le faltan las condiciones requeridas.

Documentación Class SetCalculator: En el encabeza, al igual que el anterior no posee una descripción de la clase, el constructor tampoco posee descripción al igual que los métodos. Para aquellos métodos que tienen parámetros no muestra los requisitos de estos.

4. Revisando las fuentes el proyecto

Class Set: El código está incompleto, pues hay métodos que no tienen implementado una operación. Por otro lado, el constructor no posee los mínimos requerimientos para crear el objeto.

No posee comentarios ni documentación que permitan al programador conocer cuál es la función de cada método.

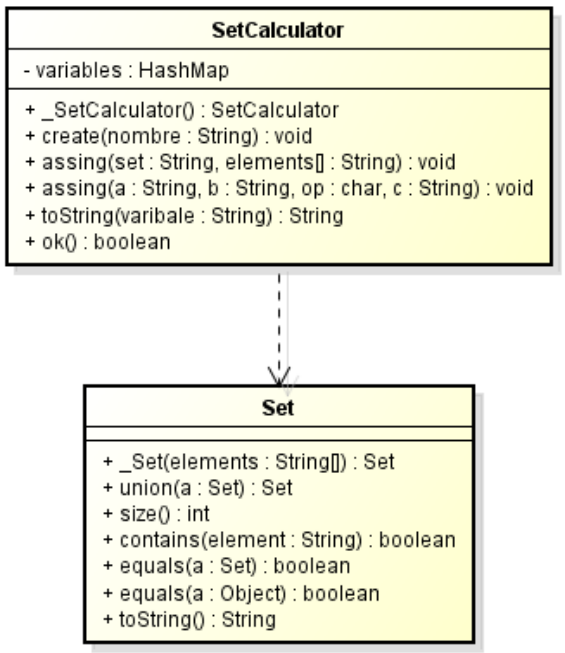
Class SetCalculator: El constructor no posee los mínimos requerimientos para crear el objeto. Los métodos están incompletos, si poseen los comentarios para conocer la función que debería cumplir cada método.

¿Qué son el código, la documentación y los comentarios?

El código son las instrucciones creadas para definir el comportamiento del objeto, en cuanto a la documentación esta sirve como guía para que el programador pueda conocer que métodos existen y como se usan. Finalmente, los comentarios permiten dar una explicación de un fragmento de código.

B. INGENIERÍA INVERSA

1. Diagrama de clases



2. Contenedor

El contenedor que está definido es $HashMap(map <? extends\ k, ? extends\ V > m)$.

Construye un nuevo HashMap con las mismas asignaciones que el Mapa especificado.

ArrayList	HashMap
+ Implementa la interfaz List. + Colección orientada a objetos a los que se accede mediante un índice. +	+ Implementa la interfaz Map (objeto que asigna claves a valores). + Toma un objeto como clave para otro objeto (valor). + Para acceder a un valor es necesario conocer la clave.

C. CONOCIENTO PRUEBAS EN BLUEJ

1. Revisando código

¿Cuáles etiquetas tiene?

@Before

@Test

@After

¿Cuántos métodos tiene?

Hay 8 métodos en la clase setTest.

¿Cuántos métodos son de prueba?

Son seis métodos de prueba, aquellos que tienen la etiqueta de test (@test).

2. Ejecutando test

¿Cuántas pruebas se ejecutan?

6 pruebas.

¿Cuántas pruebas pasan?

1 prueba pasó, esto porque en la clase Set el método size retorna siempre 0, y las demás pruebas esperaban un valor diferente de 0.

3. Estudiando etiquetas

La etiqueta **@Before** significa que se ejecutara antes de cada método.

> Al realizar pruebas, es común encontrar que se necesiten objetos inicializados antes de que estas puedan ejecutarse. @Before hace que ese método se ejecute antes que el método @Test.

La etiqueta @**Test** significa que son metodos de prueba

> La etiqueta @Test dice que el método se puede ejecutar como un caso de prueba. Cualquier excepción lanzada por la prueba será reportada como una falla. En caso contrario se asume que la prueba ha tenido éxito.

La etiqueta **@After** significa que se ejecutara después de cada método

> @After hace que se ejecute después del método @Test. Se garantiza que todos los métodos @After se ejecutarán incluso si un método @Before o @Test genera una excepción.

4. Estudiando métodos

`assertTrue`: Afirma que una condición es verdadera si no es así, lanza un error.

`assertFalse`: Afirma que una condición es falsa si no es así, arroja un mensaje de error.

`assertEquals`: Afirma que dos objetos son iguales, si no lo son lanza un error. Si ambos objetos son nulos se consideran iguales.

`assertArrayEquals`: Afirma que dos arrays de objetos son iguales, si no lo son lanza error. Si ambos son nulos se consideran iguales.

`assertNull`: Afirma que un objeto es nulo, si no es así lanza un error.

`fail`: Indica que falla una prueba.

5. Investigando y usando métodos

Una falla significa que debe volver a escribir el código que se está probando. Un error significa que puede ser la prueba unitaria la que necesita reescribir.

`shouldPass`

`shouldFail`

`shouldErr`

D. PRÁCTICANDO PRUEBAS EN BLUEJ

1. Determinando atributos

int size, String[ ] elements, estos definen un conjunto.

2. Indicando invariantes

Los tipos de elementos del conjunto.

3. Implementados métodos

```
Constructor
Set (String[ ] elements):
getElements():String [ ]
size(): int
contains(String element): boolean
equals(Set a): Boolean
toString(): String
```

E. DESARROLLANDO

Ciclo 1: Operaciones básicas de la calculadora: declarar, asignar un valor y consultar.

Métodos base:

Definimos las operaciones básicas en la calculadora para que el usuario pueda asignar un conjunto a una variable con cualquier nombre, además de esto si el usuario quiere verificar si tiene un conjunto asignado o no puede hacerlo con una consulta.

Ciclo 2/3/4/5: Operaciones binarias básicas: asignar el resultado de unión e intersección.

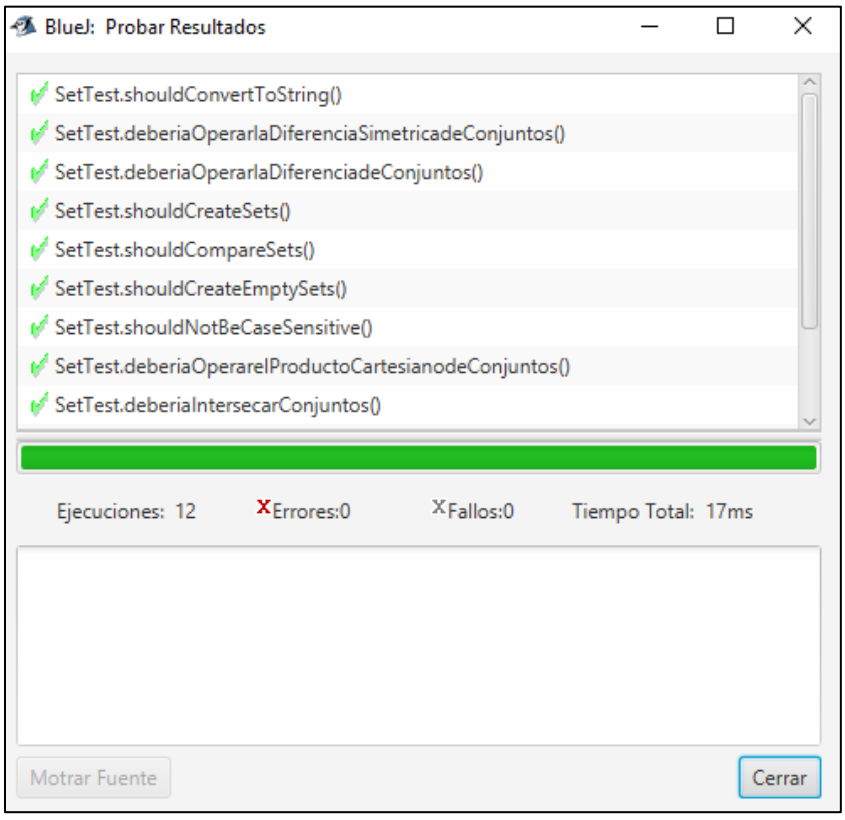
Métodos base:

Implementamos las operaciones en la clase Set para que se pueda realizar entre distintos conjuntos, además, aquel resultado de la operación lo asignamos a un conjunto nuevo

Generar y programas casos de prueba:

- Los casos de prueba planteados fueron de acuerdo con los posibles escenarios. Un caso en el que ambos conjuntos sean vacíos, el caso en donde alguno de ellos sea vacío y por último el caso donde ambos conjuntos tienen elementos.

Ejecutar pruebas:



Completen la siguiente tabla indicando el número de ciclo y los métodos asociados de cada clase.

Ciclo	SetCalculator	SetCalculatorTest
1	create(String nombre) assign(String set, String[] elements) query(String set) assign(String a, String b, char op, String c)	--
2	**Class Set** union(Set a) intersection(Set a)	deberiaUnirConjuntos() deberiaIntersecarConjuntos()
3	**Class Set** difference(Set a) symmetricDifference(Set b)	deberiaOperarlaDiferenciadeConjuntos() deberiaOperarlaDiferenciaSimetricadeConjuntos()
4	Ok() **Class Set** cartesianProduct(Set b)	deberiaOperarelProductoCartesianodeConjuntos()
5	Pertenece(String a,String element)	elementoQuePerteneceAlConjunto()

Retrospectiva

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

   (10/Angie Natalia Mojica)
   (10/Daniel Antonio Santanilla)

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

   Quedo pendiente realizar una búsqueda sobre el comportamiento de `shouldPass shouldFail shouldErr`

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

   Programación a pares: pues todo el trabajo se produjo de forma conjunta, donde simultáneamente estábamos analizando, diseñando, comprobando e intentando programar cada instrucción, consideramos que fue bastante útil, pues entre ambos fuimos aclarando conceptos y generando ideas.

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

El mayor logro fue haber aprendido un poco más de cómo usar HashMap y lo útiles que estos pueden llegar a ser, a su vez, se logró dejar más claro cómo hacer diagramas de secuencia.

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

Un problema que se nos generó fue el de realizar el diagrama de secuencias, esto debido a que se nos hace un poco confuso ver que métodos interactúan con otros y cuáles de estos realizan retornos de valores en astah.

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

Como equipo se logró mantener una buena comunicación, haber destinado un tiempo específico para desarrollar el laboratorio y cumplirlo.