**ESCUELA COLOMBIANA DE INGENIERÍA**   
**PROGRAMACIÓN ORIENTADA A OBJETOS**   
**2019-02**   
**Laboratorio 2/6**

**Conociendo el proyecto**

**1. El proyecto BlueJ “calMat” 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.**

Se puede ver la extension .class que son las clases del sistema, esta la extension .ctxt que es la documentacion que se genera apartir de las clases y la extension .java que seria el java byte code.   
Algo nuevo que vemos es el nombre de una clase que es: MatrizTest; ya que hay una clase de Matriz tambien.

**2. Exploren el proyecto en BlueJ ¿Cuántas clases tiene? ¿Cuál es la relación entre ellas? ¿Cuál es la clase principal? ¿Cómo la reconocen? ¿Cuáles son las clases “diferentes”? ¿Cuál es su propósito? Para las siguientes dos preguntas sólo consideren las clases “normales”:**

Tiene 3 clases. Matriz es la clase principal. La reconocemos porque es la unica que esta relacionada con las otras dos clases. La clase ‘diferente’ es la de MatrizTest (es de otro color, tiene un nombre diferente a los normales y dice arriba de ella ‘’unit test’’). Su proposito es tener las pruebas de unidad o pruebas unitarias que son de gran importancia a la hora de programar orientado a objetos.

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

La documentacion no esta completa pero nos ayuda a saber que neceistamos hacer repecto a metodos y demas.

**Matriz**   
Encabezado: No tiene la documentacion de lo que hace la clase.  
Métodos: Pero algunos metodos tienen documentacion.

**Calmat**  
Encabezado: Si tiene la documentaion de lo que hace la clase.  
Métodos: Pero no tiene la documentacion de ningun metodo.

**4. Revisen el código del proyecto, ¿en qué estado está cada clase? (Detalle el estado de codificación)**

**Matriz:** No tiene documentacion de encabezado, tiene algunos metodos documentados y los que no tienen documentacion tienen comentarios de ayuda a la hora de indagar el codigo para empezar a programar.

**Calmat:** Tiene documentacion de encabezado , no tiene documentacion en ningun metodo pero estos metodos tienen tambien cometarios de ayuda a la hora de empezar a programar para completar la clase como en la clase Matriz.

**Ingeniería reversa**  
**MDD MODEL DRIVEN DEVELOPMENT**

**1. Genere el diagrama de clases correspondiente a CalMat con todos sus elementos. (No incluya la clase de pruebas)**

**2. ¿Qué tipos de contenedores tienen sus colecciones? Consulte la especificación y el API Java ¿Qué diferencias hay entre ellos?**

Los contenedores que existen dentro de la clase son un stack perteneciente al paquete de java java.util.stack, y arrays definidos con “[ ]”, la diferencia entre los tipos de contenedores son las operaciones, el stack se comporta como un objeto de tipo “lifo” y cuenta con operaciones relacionadas a esto, los arrays son similares a las listas de python salvo en el sentido de la definicion del tamaño estos cuentan con operaciones similares a las listas.

**Conociendo Pruebas en BlueJ**   
**De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)**  
  
**1. Revisen el código de la clase MatrizTest. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuantos métodos son de prueba? ¿cómo los reconocen?**

@Before, @Test y @After. Tiene 11 metodos, de los cuales 8 son de prueba (los reconocimos por su arroba test).

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

Se ejecutan 8 pruebas de las cuales 3 pasan la prueba y 5 no. El codigo nunca falla, las que fallan o no son las pruebas, en este caso pasaron la prueba los tests que debian fallar, errar o no permitirse. Las pruebas que fallaron fueron las relacionadas con la construccion de la matriz y la proteccion de los datos de esta.

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

Como son etiquetas, son muy claras, el traducirlas serian: @Antes, @Prueba y @Despues. Lo cual suponemos que es que debe hacer el codigo antes de la prueba, despues de la prueba y durante la prueba. Y efectivamente ponemos el codigo que debe ejecutarse antes, durante y despues de la prueba.

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

**assertTrue:**

Hay dos metodos de assertTrue

assertTrue(boolean la condicion) .Dice si la condicion se cumple.

assertTrue(String mensaje, boolean la condicion) . Dice si la condicion se cumple y muestra un mensaje.

**assertFalse:**

Hay dos metodos de assertFalse

assertFalse (boolean la condicion) .Dice si la condicion no se cumple.

assertFalse (String mensaje, boolean la condicion) . Dice si la condicion no se cumple y muestra un mensaje.

**assertEquals:**

Hay dos metodos de assertEquals

assertEquals(String mensaje, Object esperado, Object actual). Dice si dos objetos son iguales y muesra un mensaje.

assertEquals(Object esperado, Object actual). Dice si dos objetos son iguales.

**assertNull:**

Hay dos metodos de assertNull

assertNull(Object objeto). Dice si un objeto es nulo.

assertNull(String mensaje, Object objeto). Dice si un objeto es nulo y muestra un mensaje.

**fail:**

Hay dos metodos de fail

fail() . No pasa la prueba.

fail(String mensaje). No pasa la prueba y muestra un mensaje.

**5. Investiguen la diferencia que entre un fallo y un error en Junit. Escriba código usando los métodos anteriores para lograr que los siguientes tres casos de prueba se comporten como lo prometen deberiaPasar, deberiaFallar, deberiaErrar.**

La diferencia entre un fallo y un error en junit es el momento y las condiciones en las que ocurren, un fallo ocurre cuando el codigo funciono pero el valor esperado es diferente al recibido, un error ocurre si durante el uso del programa la maquina virtual de java estalla.

Int valor = 1

* Public void deberiaPasar(){

Int valor = 1;

assertEquals(valor,1);

}

* Public void deberiaFallar(){

Int valor = 1;

assertEquals(valor,2);

}

* Public void deberiaErrar(){

Int valor = 1;

AssertEquals(valor[1],2);

}

**Prácticando Pruebas en BlueJ**   
**De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)**

**1. Determinen las estructuras de datos necesarias para almacenar los elementos de una matriz. Justifique la selección.**

Java Arrays debido a que facilita el manejo de los elementos dentro de una matriz. Con un arreglo se facilitan las operaciones a la hora de acceder a los elementos.

**2. Implementen los métodos necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?**

Implementamos los metodos y todas las pruebas pasaron.

Matriz (int e, int f, int c)  
Matriz (int elementos[][])  
Matriz (int d [])  
Matriz (int n)  
equals(Object objeto)  
get(int f, int c)  
Creamos: esValida(int[][] elementos)

**Desarrollando BDD – MDD**

**Para desarrollar esta aplicación vamos a considerar los siguientes ciclos de desarrollo.**

Ciclo 1 : Operaciones de pila: empile, desempile y consulte

Ciclo 2 : Operaciones entre matrices: sumar, restar y multiplicar (elemento a elemento y matricial)

Ciclo 3 : Operaciones de agregración para matrices: suma, minimo, máximo, promedio

Ciclo 4 : Operaciones de agregración por columnas o filas: suma, minimo, máximo, promedio

Ciclo 5 : Proponga dos nuevas funcionalidades.

### Proponemos poder sacar la matriz opuesta de una matriz dada y la matriz traspuesta de una matriz dada.

**En cada mini-ciclo deben realizar los pasos definidos a continuación.**

1. Definir los métodos base de correspondientes al ciclo actual.

2. Generar y programar los casos de prueba (piense en los deberia y los noDeberia)

**Deberia pasar:**

Deberia crear la matriz opuesta de una matriz dada.

Deberia crear la matriz transpuesta de una matriz dada.

**No deberia pasar:**

Sumar y restar matrices de diferentes dimensiones.

Multiplicar una matriz A con **c** columnas por la matriz B que tenga un numero diferente a **c** filas.

3. Diseñar los métodos (use diagramas de secuencia. En astah, adicione el diagrama al método)

4. Escribir el código correspondiente (no olvide la documentación)

5. Ejecutar las pruebas de unidad (vuelva a 3 (a veces a 2). si no están en verde)

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ciclo** | **CalMat** | **CalMatTest** | **Matriz** | **MatrizTest** |
| 1.  2.  3.  4.  5. | Metodos asociados:  empile  desempile  consulte  multiplicarMatrialcal  multiplicarPunto  opereMatriz  opereMatrices  ok  opereFilas  opereColumnas | Metodos asociados:  deberiaEmpilarUnaMatriz  deberiaDesempilarUnaMatriz  deberiaOperarMatriz  deberiaOperarMatrices  noDeberiaOperarMatrices | Metodos asociados:  dimension  get  equals  toString  sume  Reste  esValida  opereMatriz  getMatriz  minimoMatriz  maximoMatriz  promedioMatriz  opereColumnas  opereFila  sumaFilas  sumaColumnas  getFila  getColumna  opuesta  transpuesta | Metodos asociados:  deberiaCrearMatricesDadosLosElementos  deberiaProtegerLosDatos  deberiaCrearMatricesDadoValor  noDeberiaCrearMatricesInValidas  deberiaCrearMatricesDiagonales  deberiaCrearMatricesIdentidad  noDeberiaSumarMatricesInValidas  deberiaTransponerLaMatriz  deberiaSacarLaOpuestaDeLaMatriz |

**RETROSPECTIVA**

**1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)** Se trabajo aproximadamente unas 15 horas.

**2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?** Un muy buen laboratorio a la hora de entender la importancia de desarrollar primero las pruebas de unidad o unitarias.

**3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?**Progamar a pares porque hace que la reparticion del trabajo sea mucho mas equitativa y se pueda supervisar(ayudar) uno al otro.

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?** Entender la sobrecarga, la sobreescritura, las pruebas de unidad. Una cosa es entender la teoria, otra es ya hacerlo practico.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?** Fue de mucho tiempo de ensayo y eror, lo que uno hacia lo modificaba el otro y generaba conflicto. Dedicarle tiempo estando juntos hasta el punto de poder dividir de mejor manera el trabajo y aprendiendo sobre todo lo nuevo que nos pedia este laboratorio.

**6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?** Escuchar la idea del otro. Seguir viendonos y dedicandole el tiempo necesario a los diferentes trabajos.