

**ESCUELA COLOMBIANA DE INGENIERÍA**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**LABORATORIO N°2**

* **Brayan Santiago Buitrago Murillo**
* **Cesar Eduardo Gonzalez Cuentas**

**Agosto 2019**

**Bogotá**

## OBJETIVOS

Desarrollar competencias básicas para:

1. Desarrollar una aplicación aplicando BDD y MDD.
2. Realizar diseños (directa e inversa) utilizando una herramienta de modelado (astah)
3. Manejar pruebas de unidad usando un *framework ( junit)*
4. Apropiar nuevas clases consultando sus especificaciones (API java)
5. Experimentar las prácticas XP : **Coding** Code the [unit test first](http://www.extremeprogramming.org/rules/testfirst.html). **Testing** All code must have [unit tests](http://www.extremeprogramming.org/rules/unittests.html).



## CONTEXTO

# Objetivo

Los lenguajes que soportan operaciones vectorizadas son una alternativa muy interesante para implementar soluciones simples y eficientes a problemas computacionales. Para aproximarnos a este tipo de lenguajes vamos a construir una calculadora de pila de matrices calMat. Las calculadoras de pila operan sacando los operandos necesarios del tope de la pila y adicionando el resultado.

# Conociendo el proyecto [En lab02.doc]

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.

En el directorio encontrak0s archivos.class como Tambien .java y .ctxt, temenos un archive de Texto README, dispuesto a editarlo depende el trabajo que se realizara y una carpeta Doc que contiene archivos HTML y archivos de exto y una carpeta de configuraciones que tiene una imagen.gif

1. Exploren el proyecto en BlueJ

¿Cuántas clases tiene?

Hay 3 clases:

1. Callmat
2. Matriz
3. MatrizTest

¿Cuál es la relación entre ellas?

Callamt tiene como atributo a Matriz en un Stack

¿Cuál es la clase principal? ¿Cómo la reconocen?

Callmat es la que usa a matriz

¿Cuáles son las clases “diferentes”? ¿Cuál es su propósito?

La clase de MatrizTest, ya que esta se usa para realizer pruebas de unidad en los disintos metodos de las clases

Para las siguientes dos preguntas sólo consideren las clases “**normales**”:

1. 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)

Calmat no tiene nada de documentacion, la cual ya se realize por nuestra parte

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

Esta solo declarados los metodos, no estan desarrollados en todas las clases, y en la clase de test hay desarrolados 5 pruebas

**Ingeniería reversa [En CalMat.asta] MDD MODEL DRIVEN DEVELOPMENT**

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

LISTO

1. **¿Qué tipos de contenedores tienen sus colecciones? En Calmat una pila de matrices que son los operandos(para realizar las operaciones de las matrices).**

En Matriz el contenedor es una matriz de enteros inicializada con un cero.

**Consulte la especificación y el API Java 1¿Qué diferencias hay entre ellos?**

La clase Stack LIFO (last In First Out) y se diferencian sus operaciones básicas las cuales son: push, pop, peek, empty y search. Y el otro usa su misma clase para definirse la cual es una lista de listas de enteros, y funciona con las operaciones típicas de matrices.

Además de que en las matrices se manejan con dos subíndices y las Stacks solo con uno

**Prácticando Pruebas en BlueJ [En lab02.doc \*.java] De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT) Ahora vamos escribir el código necesario para que las pruebas de MatrizTest.**

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

Listas de listas, es la definición de las matrices y es la estructura de datos necesaria ya que es la ideal para guardar conjuntos de datos del mimo tipo.

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

POR HACER.

# Conociendo Pruebas en BlueJ [En lab02.doc \*.java]

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

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

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?

## Hay 9 metodos que estan etiquetados con “@” y 10 metodos en total, de los cuales 8 son de prueba y estan etiquetados con “@test”

1. 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 Ejecutaron 8 de los cuales 3 pasaron de manera correcta, 5 fallaron y cero errores, no están bien implementados

1. Estudie las etiquetas encontradas en 1. Expliqen en sus palabras su significado.

## @Test: Es un método en el cual se esta realizadon una prueba de unidad de determinada función del sistema que se espera

## @Before Cuando se escriben pruebas existen momentos en los que se necesitan objetos similares que sean creados ara que las pruebas se puedan lleva a acabo, y se ejecutara antes de todo

1. Estudie los métodos assertTrue, assertFalse, assertEquals, assertNull y fail de la clase assert del API JUnit [[1]](#footnote-1). Explique en sus palabras que hace cada uno de ellos

Assert True: Verifica si la condición es verdad de lo contrario envia un AssertionError

AsserFalse: Vetrifica si la condición es falsa y si no lo es envia un asserion error

AssertEquals: Verifica si dos objetos son iguales, se puede dejar un mensaje con la respuesta

Assert Null: Verifica si un objeto es nulo y se puede enviar un mensaje

Fail: es un método que lanza un mensaje

1. 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.

Error: Un error es porque ocurrió algo y no se pudo compilar, o llevar acabo sin que llegue al final por una acción humana

Fallo: Un fallo es cuando funciona y corre pero no envia lo que debería salir, lo que se esperaba

# Prácticando Pruebas en BlueJ [En lab02.doc \*.java]

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

Ahora vamos escribir el código necesario para que las pruebas de MatrizTest.

1. Determinen las estructuras de datos necesarias para almacenar los elementos de una matriz. Justifique la selección.
2. Implementen los métodos necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?

# Desarrollando BDD - MDD

[En lab02.doc, calMat.asta, \*.java]

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

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)**
3. **Diseñar los métodos (use diagramas de secuencia. En astah, adicione el diagrama al método)**
4. **Generar y programar los casos de prueba de los métodos de la solución (piense en todos los debería y en todos los noDebería) [OPCIONAL]**

## 5. Escribir el código correspondiente (no olvide la documentación)

**6. 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 |
|  |  |  |  |  |

## RETROSPECTIVA

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

18 horas por Hombre

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

Terminado por el compromiso

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

Todas, son utiles

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

Acabar estaba re largo

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

Entender Java

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

Trabajar con todo a la linea

1. [(http://junit.org/javadoc/latest/](http://junit.org/javadoc/latest/)) [↑](#footnote-ref-1)