

# ESCUELA COLOMBIANA DE INGENIERÍA



## PROGRAMACIÓN ORIENTADA A OBJETOS

### 2020-1

#### Laboratorio 2/6

#### 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](#). **Testing**  All code must have [unit tests](#).

#### ENTREGA

- ✓ Incluyan en un archivo **.zip** los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
- ✓ En el foro de entrega deben indicar el estado de avance de su laboratorio y los problemas pendientes por resolver.
- ✓ Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin

#### 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 matrices de fraccionarios con memoria `calmatfra`

##### Conociendo el proyecto [En lab02.doc]

1. El proyecto BlueJ "`calmatfra`" 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.
2. Explore 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**":

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

##### Ingeniería reversa [En lab02.doc `calmatfra.asta`]

##### MDD MODEL DRIVEN DEVELOPMENT

1. Genere el diagrama de clases correspondiente a `calmatfra` 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<sup>1</sup>
  - <sup>1</sup>¿Qué diferencias hay entre ellos?

---

1 <https://docs.oracle.com/javase/8/docs/api/>

## 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 `FraccionarioTest`.

1. Revisen el código de la clase `FraccionarioTest`. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuántos métodos son de prueba? ¿cómo los reconocen?
2. Ejecuten los tests de la clase `FraccionarioTest`. (click derecho sobre la clase, `Test All`) ¿cuántos tests se ejecutan? ¿cuántos pasan las pruebas? ¿por qué?
3. Estudie las etiquetas encontradas en 1. Expliquen en sus palabras su significado.
4. Estudie los métodos `assertTrue`, `assertFalse`, `assertEquals`, `assertNull` y `fail` de la clase `assert` del API `JUnit`<sup>2</sup>. Explique en sus palabras que hace cada uno de ellos.
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`, `deberiaError`.

## Practicando 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 `FraccionarioTest`.

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

## Desarrollando

### BDD - MDD

[En lab02.doc, `calmatfra.asta`, \*.java]

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

- Ciclo 1 : Operaciones de básicas: asigne, consulte
- Ciclo 2 : Operaciones aditivas: sume y reste
- Ciclo 3 : Operaciones multiplicativas: multiplicación
- 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 `deberia` y en todos los `noDeberia`) [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	Calmatfra	CalmatfraTest	Matriz	MatrizTest	Fraccionario	FraccionarioTest

## **RETROSPECTIVA**

- 1.** ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?  
(Horas/Hombre)
- 2.** ¿Cuál es el estado actual del laboratorio? ¿Por qué?
- 3.** Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
- 4.** ¿Cuál consideran fue el mayor logro? ¿Por qué?
- 5.** ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
- 6.** ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?