

ESCUELA COLOMBIANA DE INGENIERÍA

PROGRAMACIÓN ORIENTADA A OBJETOS

Diseño y Pruebas. Interacción entre objetos.

2025-1

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 : **Designing**  Use [CRC cards](#) for design sessions. **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.
- ✓ 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

En matemáticas discretas, en particular en teoría de grafos, un **grafo** es una estructura que consiste en un conjunto de objetos donde algunos pares de objetos están relacionados. Los objetos son llamados vértices y cada uno de los pares de vértices relacionados se llama arco.

Conociendo el proyecto [En lab02.doc]

1. El proyecto "[graphCalculator](#)" contiene una construcción parcial del sistema. Revisen el directorio donde se encuentra el proyecto. Describan el contenido en términos de directorios y de 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 de la aplicación? ¿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? (Detallen el estado de documentación: encabezado y métodos)
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 diferencia hay entre el código, la documentación y los comentarios?

Ingeniería reversa [En lab02.doc GraphCalculator.asta]

MDD MODEL DRIVEN DEVELOPMENT

1. Realicen el diagrama de clases correspondiente al proyecto. (No incluyan la clase de pruebas)
2. ¿Cuáles contenedores están definidos? ¿Qué diferencias hay entre el nuevo contenedor, el [ArrayList](#) y el vector `[]` que conocemos? Consulte el API de java.
3. En el nuevo contenedor, ¿Cómo adicionamos un elemento? ¿Cómo lo consultamos? ¿Cómo lo eliminamos?

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 implementaremos algunos métodos en la clase `GraphTest`

1. Revisen el código de la clase `GraphTest`. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuántos métodos son de prueba? ¿cómo los reconocen?
2. Ejecuten los tests de la clase `GraphTest`. (click derecho sobre la clase, Test All) ¿cuántas pruebas se ejecutan? ¿cuántas pasan? ¿por qué? Capturen la pantalla.
3. Estudie las etiquetas encontradas en 1 (marcadas con @). Expliquen en sus palabras su significado.
4. Estudie los métodos `assertTrue`, `assertFalse`, `assertEquals`, `assertNull` y `fail` de la clase `Assert` del API `JUnit`¹. Explique en sus palabras que hace cada uno de ellos.
5. Investiguen y expliquen la diferencia que entre un fallo y un error en `JUnit`. Escriba código, usando los métodos del punto 4., para codificar los siguientes tres casos de prueba y lograr que se comporten como lo prometen `shouldPass`, `shouldFail`, `shouldErr`.

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 `GraphTest` pasen.

1. Determinen los atributos de la clase `Graph`. Justifique la selección.
2. Determinen el invariante de la clase `Graph`. Justifique la decisión.
3. Implementen los métodos de `Graph` necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?
4. Capturen los resultados de las pruebas de unidad.

Desarrollando `GraphCalculator`

BDD - MDD

[En lab02.doc, `GraphCalculator.asta`, *.java]

Para desarrollar esta aplicación vamos a considerar algunos ciclos. En cada ciclo deben realizar los pasos definidos a continuación.

1. **Definir los métodos base de correspondientes al mini-ciclo actual.**
2. **Definir y programar los casos de prueba de esos métodos**
Piensen en los `deberia` y los `noDeberia` (`should` and `shouldNot`)
3. **Diseñar los métodos**
Usen diagramas de secuencia. En `astah`, creen el diagrama sobre el método correspondiente.
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)**
6. **Completar la tabla de clases y métodos. (Al final del documento)**

Ciclo 1 : Operaciones básicas de la calculadora: crear una calculadora y asignar y consultar un grafo

Ciclo 2 : Operaciones unarias: insertar y eliminar arcos; consultar si un conjunto de vertices pertenece al graph y retornar el camino que pasa por un conjunto de vértices

Ciclo 3 : Operaciones binarias: union, intersección, diferencia y junta

BONO Ciclo 4 : Defina dos nuevas operaciones

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

Ciclo	GraphCalculator	GraphCalculatorTest

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?
7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.