



ESCUELA COLOMBIANA DE INGENIERÍA
PROGRAMACIÓN ORIENTADA A OBJETOS 2021-1
Diseño y Pruebas. Interacción entre objetos.
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

En este laboratorio vamos a construir una calculadora para vectores en espacios bi-dimensionales : [CalVectorial](#)

Conociendo el proyecto [\[En lab02.doc\]](#)

1. El proyecto BlueJ “[calVectorial](#)” 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. Tiene una carpeta llamada doc que es creada por BlueJ para mostrar la documentación en la pestaña documentación una vez abierto el package. Por cada clase, blueJ tiene 3 archivos el .java que es el código fuente. El .ctxt que marca la documentación presente en el código y el .class que es el compilado del código fuente.

2. Explore el proyecto en BlueJ

¿Cuántas clases tiene? ¿Cuál es la relación entre ellas?

El proyecto cuenta con 4 clases las cuales son: CalVectorial, Vector, Ángulo y AnguloTest

Las relaciones que hay entre las clases son: de CalVectorial a Vector, de vector a Angulo y de AnguloTest a Angulo.

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

La clase principal del proyecto es CalVectorial, el reconocimiento es porque el objetivo del laboratorio es construir una calculadora para vectores en espacios bidireccionales.

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

La clase “unit Test” AnguloTest, la cual el propósito es realizar pruebas de unidad en el código del proyecto que se está trabajando

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 de cada clase: encabezado y métodos)
No, la documentación está incompleta y falta gran parte del código en las clases

4. Revisen las fuentes del proyecto, ¿en qué estado está cada clase? (Detallen el estado de las fuentes considerando: código, documentación y comentarios) ¿Qué son el código, la documentación y los comentarios?

Tanto en la clase `CalVectorial`, en la `Vector` y en la de `Ángulo` encontramos los métodos declarados pero no contruidos, es decir que no funcionarían por el momento y la documentación está específica de qué es lo que quiere hacer cada método.

Son la estructura de cada clase que contenga el proyecto, es decir, el código es quien me dará el funcionamiento de la clase y sus métodos. Por otra parte la documentación es la que me da la descripción ya sea de la clase o de sus métodos que parámetros tiene, que me retorna y demás. Y por último, los comentarios se diferencian de la documentación porque esta va dirigida a quien lo lea y la otra está dirigida hacia los métodos.

Ingeniería reversa [En lab02.doc CalVectorial.asta]

MDD MODEL DRIVEN DEVELOPMENT

1. Complete el diagrama de clases correspondiente a `CalVectorial`.
(No incluya la clase de pruebas)
2. ¿Qué tipo de contenedor está definido? Consulte la especificación y el API Java ¹¿Qué diferencias hay entre el nuevo contenedor y el `ArrayList` que conocemos?
El `HashMap` contiene dos tipos de datos: las `keys`(que identifican cada valor y son irrepetibles) y los valores que pueden tomar cualquier valor y se pueden repetir. Básicamente permite recuperar objetos con las `keys` y no con los índices como se hacía en el `ArrayList` así la obtención de datos es constante y no lineal como en el `ArrayList`.

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

De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)

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

1. Revisen el código de la clase `AnguloTest`. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuántos métodos son de prueba? ¿cómo los reconocen?
La clase `AnguloTest` cuenta con 8 etiquetas: las cuales son: `setUp`, `deberiaCrearBienLosAngulo`, `deberiaDarElValorEnGrados`, `deberiaDarElValorEnRadianes`, `deberiaDarElValorEnGradianes`, `deberiaSumar`, `deberiaRestar`, `deberiaMultiplicar` y `tearDown`.

La clase `AnguloTest` cuenta con 8 métodos.

6 de los métodos son de prueba, los reconocemos ya que las etiquetas con `@test` nos están diciendo que son de prueba.

2. Ejecuten los tests de la clase `AnguloTest`. (click derecho sobre la clase, `Test All`)
¿Cuántas pruebas se ejecutan? ¿Cuántos pasan las pruebas? ¿por qué?
Al ejecutar los tests de la clase `AnguloTest`, se ejecutan en total 7 pruebas, pero todas salen con fallos, es decir, que ninguna de estas pasan las pruebas. Y la razón para que esto suceda es porque los tests están directamente relacionados con la clase `Ángulo` y eso es lo que se quiere probar con estos tests, pero por el momento la clase `Ángulo` no tiene funcionalidad ya que sus métodos sólo están definidos o declarados.
3. Estudie las etiquetas encontradas en 1. Expliquen en sus palabras su significado.
En la clase `Ángulo Test` encontré 3 tipos de etiquetas que son: `@before`, `@test` y `@after`.
Supongo que por los nombres, el `@before` configura los tests de prueba antes de llamarlos, es como para darles una inicialización, el `@test` realiza los métodos que son de prueba para verificar su funcionalidad y por último el `@after` termina con los tests de prueba anteriores.

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

4. Estudie los métodos [assertTrue](#), [assertFalse](#), [assertEquals](#), [assertArrayEquals](#), [assertNull](#) y [fail](#) de la clase [assert](#) del API [JUnit](#)². Explique en sus palabras que hace cada uno de ellos.
El método `assertTrue` válida si la condición es verdadera
El método `assertFalse` válida si la condición es falsa
El método `assertEquals` compara dos tipos de datos numéricos
El método `assertArrayEquals` según el tipo lo que hace es igualar dos objetos y ver si cada uno de ellos cuenta con las mismas componentes, puede trabajar con arreglos y matrices.
El método `assertNull` si lo que se le entrega al parámetro es verdadero retorna null de lo contrario retorna false
El método `fail` dice si falla una prueba sin mensaje.
5. Investiguen la diferencia entre un fallo y un error en [JUnit](#). Escriba código, usando los métodos del punto 4., para lograr que los siguientes tres casos de prueba se comporten como lo prometen [deberiaPasar](#), [deberiaFallar](#), [deberiaError](#).

Una falla significa que su prueba se ejecutó correctamente e identificó un defecto en su código.

Un error podría significar un error en su código, pero uno que ni siquiera estaba probando. También podría significar que el error está en la prueba en sí.

Código implementado.

```
@Test
public void deberiaPasar(){
    assertTrue(true);
}

@Test
public void deberiaFallar(){
    assertFalse(true);
}

@Test
public void deberiaError(){
    int[] number = new int[]{0};
    assertTrue(number[2] < 3);
}
```

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 [AnguloTest](#) pasen.

1. Determinen los atributos de la clase [Angulo](#). Justifique la selección.

```
/** Constantes para indicar que el argumento está en radianes */
public static final int RADIANES = 0;
/** Constantes para indicar que el argumento está en grados */
public static final int GRADOS = 1;
/** Constantes para indicar que el argumento está en gradianes */
public static final int GRADIANES = 2;

/** Constante para maximo error admitido al comparar dos angulos.
 * Recuerde que los cálculos en el computador con variables de punto flotante
 * tienen una precisión limitada, y se requiere un margen de tolerancia
 */

public static final double MAXERROR = 0.000000000000001;
```

² (<http://junit.org/javadoc/latest/>)

- Implementen únicamente los métodos de [Angulo](#) necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?

Desarrollando **CalVectorial**

BDD - MDD

[En lab02.doc, CalVectorial.asta, *.java]

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

- Definir los métodos base de correspondientes al ciclo actual.**
- Generar y programar los casos de prueba**
(piense en los debería y los noDebería)
- Diseñar los métodos**
(Use diagramas de secuencia. En astah, adicione el diagrama al método.
Revisen en los ejemplos el estándar y la configuración de los diagrama de secuencia.)
- Generar y programar los casos de prueba de los métodos de la solución**
(piense en todos los debería y en todos los noDebería) [OPCIONAL]
- Escribir el código correspondiente (no olvide la documentación)**
 - Ejecutar las pruebas de unidad (vuelva a 3 (a veces a 2). si no están en verde)**
- Completar la tabla de clases y métodos. (Al final del documento)**

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

Ciclo 2 : Operaciones binarias básicas: asignar el resultado de sumas y restas

Ciclo 3 : Operaciones unarias básicas: asignar unitario, horizontal y vertical Ciclo 4 :

Operaciones binarias avanzadas: producto y proyecciones Ciclo 5 : Defina una nueva funcionalidad.

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

Ciclo	CalVectorial	CalVectorialTest

RETROSPECTIVA

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