ICC 2024-1

Proyecto 1 - Intervalos numéricos

Pedro Ulises Cervantes González confundeme@ciencias.unam.mx
Julio Vázquez Alvarez julio.vazquez@ciencias.unam.mx
Ian Israel García Vázquez iangarcia@ciencias.unam.mx
Erick Bernal Márquez erick07@ciencias.unam.mx
Yessica Janeth Pablo Martínez yessica_j_ pablo@ciencias.unam.mx

Elaborado por:
-Cano Tenorio Alan Kevin (321259967)
-Malinalli Escobedo Irineo (42412101)

Sobre el proyecto:

Este proyecto trata sobre los intervalos numéricos, los cuales, son los objetos de interés para este trabajo. Pueden tener cuatro formas:

- Un intervalo cerrado [a, b]
- Un intervalo abierto (a, b)
- El intervalo (a, b]
- El intervalo [a, b)

A continuación, se describe el trabajo realizado para poder cumplir con los requisitos de este proyecto.

El proyecto se divide en 3 archivos con la extensión *.java*:

- ➢ *ServiciosIntervalo.java:* Este archivo es la interfaz que se implementó a una clase llamada "*Intervalo*". Contiene los encabezados de diferentes métodos que se desarrollaron en el archivo *Intervalos.java* , estos métodos son "Los servicios que debe ofrecer un Intervalo definido en los números reales delimitado por los valores a y b"

- ➢ *Intervalos.java* : Este archivo es la clase llamada *"Intervalo"* con la cual, creamos los objetos de tipo "Intervalo" y además, es la clase que tiene implementada la interfaz "*ServiciosIntervalo*". A esta clase se le asignaron diferentes variables para que se pudieran desarrollar de manera eficaz los diferentes métodos de la interfaz a lo largo de esta clase, y también para utilizar el método constructor para poder mostrar un objeto de forma correcta. La estructura de este archivo se divide en:
  - ❖ Las variables:
    - ➔ *private boolean aAbierto*: Sí es verdadero, indica que el inicio del intervalo es abierto, sí es falso, indica que el inicio del intervalo es cerrado.
    - ➔ *private boolean bAbierto*: Sí es verdadero, indica que el final del intervalo es abierto, sí es falso, indica que el final del intervalo es cerrado.

- ➔ *private double a* : Es el número dentro de los reales con el que inicia el intervalo.
- ➔ *private double b* : Es el número dentro de los reales con el que termina el intervalo.
- ➔ *private double longa:* Indica la longitud del intervalo “a” o del objeto que utiliza los métodos.
- ➔ *private Intervalo intervalo:* es el intervalo que utiliza los métodos.

Estas son de acceso *private* por buenas prácticas de programación, y para poder utilizarlas se utilizaron métodos de acceso y métodos mutantes.

- ❖ Métodos de acceso “getters” : estos devuelven las variables de tipo *private* para que puedan ser utilizadas por la clase *Main.*
- ❖ Métodos mutantes “setters” : le asignan el mismo valor de las variables *private* a las variables obtenidas por los getters.
- ❖ Métodos constructores: se crearon dos diferentes para tener en cuenta cuando se crea un intervalo con valores y otro sin valores:
  - ➔ *public Intervalo (boolean aAbierto, double a, double b, boolean bAbierto)* : Muestra un intervalo tomando en cuenta si cada uno de sus extremos son abiertos o cerrados y los valores de inicio y final.
  - ➔ *public Intervalo ()* : no muestra ningún intervalo si no se le colocan valores.
- ❖ Métodos de la interfaz: son aquellos que fueron implementados al momento de implementar la clase con la interfaz, estos métodos son:
  - ➔ *public boolean contiene (double x)* : se creó una variable *boolean existe*, la cual, será devuelta al definirse por el uso de un operador ternario, este toma el tipo de intervalo que se está usando y, a partir de este, se comparan los valores de *a, b* y *x* de la forma que está definida en las instrucciones del proyecto.
  - ➔ *public boolean estaVacio ()* : se creó una variable *boolean vacio,* la cual, será devuelta al definirse por el uso de un operador ternario, en el cual, toma en cuenta si el intervalo es cerrado y si los valores de *a* y *b* son iguales.

- ➔ *public double longitud ():* por medio de la resta de b-a se define la variable *longa* para que devuelva la longitud de un intervalo.
- ➔ *public boolean mideMas (Intervalo otro):* se creó una variable *double longb* que se define como la longitud del intervalo *otro,* esto se hizo mandando a llamar el método *public double longitud ().* Esto se hizo con la finalidad de poder definir una variable *boolean midemas* por medio de un operador ternario que compara si *longa>longb* y devuelva su valor.
- ➔ *public boolean esIgual (Intervalo otro):* Se define la variable *boolean igual* con un operador ternario y la comparación de dos intervalos. Esta variable devolverá un true o false depende el resultado de la comparación.
- ➔ *public Intervalo desplaza (double c)* : Como su nombre lo indica toma un intervalo y los desplaza en "c" unidades, para crearlo tomamos los valores originales "de a y b", a cada uno le sumamos "c" que es el desplazó que tendrá el nuevo intervalo finalmente regresamos un objeto de tipo intervalo que contiene las variables ya modificadas
- ➔ public boolean contiene (Intervalo otro); Comparamos los extremos de dos intervalos para ver si un intervalo estaba contenido dentro de otro y regresamos una variable de tipo booleano la cual indicaba si esto se cumplía o no
- ➔ public Intervalo intersecta (Intervalo otro); Comparamos dos intervalos para ver cual era la intersección de estos hay varios posibles casos
    - Que no haya intersección
    - Que el intervalo 1 esté dentro del intervalo 2
    - Que el intervalo 2 esté dentro del intervalo 1
    - Que el intervalo 1 y 2 sean diferentes y el intervalo 2 intercepte el intervalo 1 por la derecha
    - Que el intervalo 1 y 2 sean diferentes y el intervalo 2 intercepte al intervalo 1 por la izquierda
- ➔

- ➔ public Intervalo intermedio (Intervalo otro); El método intermedio compara un intervalo(1) con otro intervalo(2) para saber si entre estos existe un intervalo intermedio en caso de que no retorna un objeto de tipo vacío en caso de que sí exista dicho intervalo retorna los valores de b(del intervalo 1) y a(del intervalo 2
- ➔ public String muestra (); El método muestra se utilizó un operador ternario para que los datos del intervalo se imprimen en pantalla planteando los cuatro posibles intervalos que podríamos tener:
    - [a, b]
    - (a, b)
    - (a, b]
    - [a, b)

➢ *Main.java* : Este archivo es la clase principal del programa, por el cual, se crean los objetos y se muestra en la terminal los diferentes métodos de la clase *Intervalo.* La estructura de esta clase es la siguiente:

- ❖ *public static void main(String[] args)* : por medio de este se va a ejecutar todo lo que se coloque en su cuerpo. Su cuerpo está conformado por:
    - ➔ Objetos: Se crearon 4 intervalos, los cuales, cada uno representa los cuatro tipos de intervalos que existen. Se definieron por el método constructor y por valores que se le brindaron para poder ser usados.
    - ➔ Objetos copia: Debido a que el método “desplazar” modifica los valores de los intervalos, se optó por crear objetos copia, los cuales, tienen exactamente los mismos valores que los originales. Estos serán mostrados solos en dicho método para que no se afecten los resultados de los métodos posteriores a ese.
    - ➔ Métodos de la interfaz: Primero se muestran los 4 intervalos creados para, posteriormente, por medio de la definición de las variables usadas en diferentes parámetros y mandando a llamar los métodos de la clase *Intervalo,* se imprimen y ejecutan los diferentes métodos de la interfaz para los 4 intervalos creados.

Inconvenientes que se presentaron:

Uno de los inconvenientes que se presentó al momento de realizar este proyecto fue con el método *"intermedio"* dado que no sabíamos que se podían crear objetos dentro de un método, también dentro del mismo método queríamos que ya se mostrará el intervalo de la forma (a,b], sin embargo lo que se retorna es un tipo un intervalo y no String por lo que se tiene que mandar a llamar el *método intermedio* que fue el que se creó y aparte mandar a llamar el *método muestra* para que en la terminal se muestre el intervalo creado.

En general el mayor inconveniente fue que no sabíamos que podíamos crear objetos dentro de métodos para así retornarlos.

Otro inconveniente que se me presentó fue con el *método es igual* dado que no sabíamos que se podían comparar dos objetos.

¿Cómo ejecutarlo?

Primero se descarga el archivo. Zip que se subió al classroom y guardalo en la carpeta que desees

Posteriormente se descomprime el .zip (solo tienes que dar doble clic sobre el archivo)

Abre la terminal y métete a la carpeta en la cual guardaste el archivo, para hacer esto escribe

computadora:~$ cd + "nombre de la carpeta que descargaste"

En este caso mis archivos están guardados en una carpeta llamada "Proyecto01CanoTenorio". posteriormente escribe el comando *ls*

Si te aparecen los mismos archivos que en la imagen ya están dentro de la carpeta correcta

En la terminal escribe Javac *Java

Esto compila todos los archivos que tengas en esa carpeta

Finalmente escribe Main Java y en en la terminal se mostrará lo siguiente:

```
El intervalo 1 es: (2.0,10.0)
El intervalo 2 es: [3.0,3.0)
El intervalo 3 es: (4.0,5.0]
El intervalo 4 es: [1.0,7.0]

¿El número 3.0 está en el intervalo 1? true
¿El número 3.0 está en el intervalo 2? false
¿El número 3.0 está en el intervalo 3? false
¿El número 3.0 está en el intervalo 4? true

¿El intervalo 1  está vacio? false
¿El intervalo 2  está vacio? true
¿El intervalo 3  está vacio? false
¿El intervalo 4  está vacio? false

La longitud del intervalo 1 es de: 8.0
La longitud del intervalo 2 es de: 0.0
La longitud del intervalo 3 es de: 1.0
La longitud del intervalo 4 es de: 6.0

¿El intervalo 1 mide mas que el intervalo 2? true
¿El intervalo 2 mide mas que el intervalo 3? false
¿El intervalo 3 mide mas que el intervalo 4? false
¿El intervalo 4 mide mas que el intervalo 1? false

¿El intervalo 1 es igual al intervalo 2? false
¿El intervalo 2 es igual al intervalo 3? false
¿El intervalo 3 es igual al intervalo 4? false
¿El intervalo 4 es igual al intervalo 1? false

Si el intervalo 1 es dezplazado en -3.0 unidades, quedaria:(-1.0,7.0)
Si el intervalo 2 es dezplazado en -3.0 unidades, quedaria:[0.0,0.0)
Si el intervalo 3 es dezplazado en -3.0 unidades, quedaria:[1.0,2.0]
Si el intervalo 4 es dezplazado en -3.0 unidades, quedaria:[-2.0,4.0]

¿El intervalo [3.0,3.0) está contenido en el intervalo (2.0,10.0)? true
¿El intervalo (4.0,5.0] está contenido en el intervalo [3.0,3.0)? false
¿El intervalo [1.0,7.0] está contenido en el intervalo (4.0,5.0]? false
¿El intervalo (2.0,10.0) está contenido en el intervalo [1.0,7.0]? false

El intervalo que resulta de la interseccion de los intervalos 1 y 2 es: (3.0,3.0)
El intervalo que resulta de la interseccion de los intervalos 2 y 1 es: [0.0,0.0)
El intervalo que resulta de la interseccion de los intervalos 3 y 4 es: (4.0,5.0]
El intervalo que resulta de la interseccion de los intervalos 4 y 3 es: [4.0,5.0]

El intervalo, que cabe entre el intervalo 1 y el 2 es: [0.0,0.0]
El intervalo, que cabe entre el intervalo 2 y el 3 es: [0.0,4.0)
El intervalo, que cabe entre el intervalo 3 y el 4 es: [0.0,0.0]
El intervalo, que cabe entre el intervalo 4 y el 1 es: [0.0,0.0]
```