Guía Práctica de Laboratorio
Sesión 5: INTRODUCCIÓN A LA PROGRAMACIÓN
MLBC - VACJ - CJFV

En esta sesión,

- debes analizar cuales son los procedimientos generales que deben realizar los objetos de una clase para comportarse de manera correcta
- implementar los métodos REQUERIDOS de las clases que se describen en esta práctica

TAREA

1. **Complejo** Los números complejos incluyen todas las raíces de los polinomios, a diferencia de los reales. Todo número complejo puede representarse como la suma de un número real y un número imaginario (que es un múltiplo real de la unidad imaginaria, que se indica con la letra i). Por ejemplo: 5 + 4i es un número complejo.

   Considera el modelo para poder

   a) sumar,
   b) restar,
   c) multiplicar y
   d) mostrar números complejos.

   El siguiente código tiene lo mínimo necesario para representar un objeto de tipo Complejo:

```
1  class Complejo{
2     private double real;
3     private double imaginario;
4     public Complejo(double real, double imaginario){
5       this.real      = real;
6       this.imaginario = imaginario;
7     }
8     public Complejo sumar(Complejo otro){
9       return null; // metodo OPA
10    }
11    public Complejo restar(Complejo otro){
12      return null; // metodo OPA
13    }
14    public Complejo multiplicar(Complejo otro){
15      return null; // metodo OPA
16    }
17    public String mostrar(){
18      return null; // metodo OPA
19    }
20 }
```

   El siguiente pedazo de código muestra a objetos de la clase Complejo respondiendo a mensajes acordes al comportamiento ofrecido:

```
1  Complejo num1, num2;
2  Complejo num3, num4, num5;
3  Complejo num6, num7;
4  String num;
5  num1 = new Complejo(3.0, 5.5);
6  num2 = new Complejo(4.0, 7.0);
7  num3 = num1.sumar(num2);
8  num4 = num1.multiplicar(num3);
9  num5 = num1.restar(num4);
10 num6 = new Complejo(6.0, 0.0);
11 num7 = new Complejo(0.0, 9.7);
12 num  = num3.mostrar();
13 num  = num6.mostrar();
14 num  = num7.mostrar();
```

Se han declarado siete objetos de clase **Complejo**: *num1, num2, num3, num4, num5, num6 y num7.*

num1 representa al complejo *3.0 + 5.5i*,

num2 representa a *4.0 + 7.0i*.

Al finalizar este pedazo de código el:

- *num3* representaría al complejo *7.0 + 12.5i*.
- *num4* representaría al complejo *-47.75 + 76i*.
- *num5* representaría al complejo *50.75 - 81.5i*.
- la primera vez que se pide a un objeto de tipo Complejo mostrar, *num* representa a la cadena *"7.0 + 12.5i"*
- la segunda vez que se pide a un objeto de tipo Complejo mostrar, *num* representa a la cadena *"6.0"*
- la tercera vez que se pide a un objeto de tipo Complejo mostrar, *num* representa a la cadena *"9.7i"*

Mejora el método mostrar de la clase Complejo, de tal manera que considere los casos especiales en los que solamente tiene una de las dos partes.

2. **Vector** Un vector puede utilizarse para representar una magnitud física, quedando definido por un módulo y una dirección u orientación. Su expresin geométrica consiste en segmentos de recta dirigidos hacia un cierto lado, asemejándose a una flecha.

   Considera el modelo para que se pueda:

   a) sumar con otro vector

   b) multiplicar con otro vector

   c) calcular los grados del vector considerando su dirección

   d) decir a que cuadrante del eje de coordenadas el vector apunta

   e) calcular la magnitud del vector

   El modelo a continuación representa al vector $\overrightarrow{AB}$

```
1  /** clase que representa a un vector entre el punto A y B, con direccion
2  *   de A a B, tambien representado AB con direccion ->
3  **/
4  class Vector{
5        private int ptoAX;
6        private int ptoAY;
7        private int ptoBX;
8        private int ptoBY;
9        public Vector(int ptoAX, int ptoAY, int ptoBX, int ptoBY){
10         this.ptoAX = ptoAX;
11         this.ptoAY = ptoAY;
12         this.ptoBX = ptoBX;
13         this.ptoBY = ptoBY;
14       }
15       public Vector sumar(Vector otro){
16         return null;
17       }
18       public Vector multiplicar(Vector otro){
19         return null;
20       }
21       public double calcularGrados(){
22         return 0.0;
23       }
24       public String ubicarCuadrante(){
25         return null;
26       }
```

```
27        public double calcularMagnitud(){ // esto es equivalente al modulo
28          return 0.0;
29        }
30  }
```

Para comprender mejor el modelo y cómo éste se usa, te presentamos el siguiente pedazo de código que muestra un ejemplo particular:

```
1   Vector vec1, vec2, vec3;
2   double grados, magnitud;
3   String cuadrante;
4   vec1 = new Vector(1, 1, 3, 6);
5   vec2 = new Vector(2, 2, 5, 6);
6   vec3 = vec1.sumar(vec2);
7   grados = vec3.calcularGrados();
8   magnitud = vec2.calcularMagnitud();
9   cuadrante = vec1.ubicarCuadrante();
```

Se han declarado tres objetos de clase **Vector**: *vec1, vec2 y vec3*.

vec1 representa al vector $\overrightarrow{AB}$ donde A es el punto *(1,1)* y B es el punto *3,6*;

vec2 representa al vector $\overrightarrow{CD}$ donde C es el punto *(2,2)* y D es el punto *5,6*

Al finalizar este pedazo de código el:

- *vec3* representaría al vector $\overrightarrow{XY}$ donde X es el punto *(1,1)* e Y es el punto *6,10*
- *grados* representaría al ángulo en grados *29.054604099077146*.
- *magnitud* representaría al número real *5.0*.
- *cuadrante* sera "Noreste"

Todos los vectores del ejemplo están en cuadrante "Noreste". Hay algunos vectores que no están en ningún cuadrante, por ejemplo el vector $\overrightarrow{XY}$ donde X es el punto *(0, 0)* e Y es el punto *(0, 1)* ya que se encuentra sobre la linea. En este caso si se le pregunta a este vector su cuadrante debería responder "Ningun". Para esta versión de implementación, en caso de que el vector se encuentre en más de un cuadrante, la respuesta debe ser "Ningun"

Implementa el método: ubicarCuadrante.

AYUDA: para implementar estos métodos, revisa la clase *Math* de la biblioteca de *java.lang*, ingresa a *https://docs.oracle.com/javase/7/docs/api/*

3. **Cuentas, Cuentas** El dinero es algo que mueve al mundo, y las instituciones que regulan y manejan este concepto son los bancos, es así que ellos permiten a las personas guardar de manera segura el dinero que tiene a través de Cuentas Bancarias, de las cuales se tiene: el numero de cuenta que generalmente es un numero grande, el saldo que tiene, el cliente y la moneda de la cuenta (Bs., $us). Refleja estos datos mínimos en un modelo Orientado a Objetos.

   En un futuro, es posible:

   a) depositar un monto en la cuenta

   b) retirar un monto de la cuenta

   c) transferir un monto a otra cuenta

   d) verificar si el numero de la cuenta es igual a *nroCta*

   e) consultar el saldo de la cuenta

   Con este contexto, se ha modelado una Cuenta Bancaria como sigue:

```
1
2  class CuentaBancaria{
3    private String numeroCuenta;
4    private double saldo;
5    private String cliente;
6    private String moneda;
7
8    public CuentaBancaria(String numeroCuenta, String cliente, String moneda){
9       this.numeroCuenta = numeroCuenta;
10      this.cliente      = cliente;
11      this.moneda       = moneda;
12      saldo             = 0.0;
13   }
14
15   public void depositar(double monto){
16   }
17   public boolean retirar(double monto){
18      return false;
19   }
20   public boolean transferir(double monto, String otroNroCuenta){
21      return false;
22   }
23   public boolean verificarNroCuenta(String numeroCuenta){
24      return false;
25   }
26   public double consultarSaldo(){
27      return 0.0;
28   }
29 }
```

Para comprender mejor el modelo y cómo éste se usa, te presentamos el siguiente pedazo de código que muestra un ejemplo particular:

```
1  CuentaBancaria cta1, cta2;
2  double saldo;
3  String reporte;
4  cta1 = new CuentaBancaria("1000785436", "Luis Choque", "Bs");
5  cta2 = new CuentaBancaria("1030565886", "Clark kent", "$us");
6  saldo  = cta1.consultarSaldo();
7  cta1.depositar(100.0);
8  saldo  = cta1.consultarSaldo();
9  saldo  = cta2.consultarSaldo();
10 reporte = cta2.retirar(200);
11 reporte = cta1.retirar(70);
```

Se han declarado dos objetos de clase **CuentaBancaria**: *cta1 y cta2*.

cta1 representa a la cuenta bancaria cuyo numero de cuenta es *1000785436*, le pertenece al cliente *Luis Choque* y la cuenta esta en *Bs.*.

cta2 representa a la cuenta bancaria cuyo numero de cuenta es *1030565886*, le pertenece al cliente *Clark kent* y la cuenta esta en *$us.*.

Al finalizar este pedazo de código el saldo es consultado tres veces a distintos objetos, en el tiempo el saldo tomaría valores de:

- primera consulta *saldo* es *0.0*
- segunda consulta *saldo* es *100.0*
- tercera consulta *saldo* es *0.0*
- despues del primer intento de retiro de dinero, el *reporte* deberia ser el mensaje "Saldo insuficiente, transacción inválida"
- despues del segundo intento de retiro de dinero, el *reporte* deberia ser el mensaje "Transacción exitosa, su retiro fue de 70 Bs."

Implementa los métodos: retirar de la Cuenta Bancaria.

4. **Estante de libros** Lulú está en la tarea de construir un estante de libros, pero quiere hacer uno a medida, por lo que esta reuniendo sus libros, cuentos y revistas. No le pone atención a que tipo de escrito es, sino más bien al tamaño del libro tanto ancho como alto, para poder planear bien su estante. Por el momento, se requiere modelar los libros para ayudar a Lulú.

   Los libros deberían poder:

   a) compararse con otro libro considerando su tamaño por altura

   b) compararse con otro libro considerando su tamaño por anchura

   c) compararse con otro libro por tamaño

   d) decir su alto

   e) decir su ancho

   Recordemos que cuando se comparan dos cosas, se puede obtener uno de tres posibles resultados: $>, < o =$

   Considera el siguiente código que representa a objetos de tipo Libro:

```
1  class Libro{
2      private int altura;
3      private int ancho;
4      public Libro(int altura, int ancho){
5          this.altura = altura;
6          this.ancho  = ancho;
7      }
8      public int comparar(Libro otro){
9          return 0;
10     }
11     public int compararAltura(Libro otro){
12         return 0;
13     }
14     public int compararAncho(Libro otro){
15         return 0;
16     }
17     public int getAltura(){
18         return 0;
19     }
20     public int getAncho(){
21         return 0;
22     }
23 }
```

   Para comprender mejor el modelo y cómo éste se usa, te presentamos el siguiente pedazo de código que muestra un ejemplo particular:

```
1  Libro libro1, libro2, libro3, libro4, libro5;
2  int resultado;
3  libro1 = new Libro(10, 20);
4  libro2 = new Libro(20, 25);
5  libro3 = new Libro(20, 30);
6  libro4 = new Libro(35, 10);
7  libro5 = new Libro(20, 25);
8  resultado = libro1.comparar(libro2);
9  resultado = libro3.comparar(libro2);
10 resultado = libro4.comparar(libro2);
11 resultado = libro2.comparar(libro5);
```

   Se han declarado cuatro objetos de clase **Libro**: *libro1, libro2, libro3, libro4 y libro5.*

   libro1 representa al libro cuya altura es *10* y el ancho es *20.*

   libro2 representa al libro cuya altura es *20* y el ancho es *25.*

libro3 representa al libro cuya altura es *20* y el ancho es *30*.

libro4 representa al libro cuya altura es *35* y el ancho es *10*.

libro5 representa al libro cuya altura es *20* y el ancho es *25*.

Al finalizar este pedazo de código el resultado es establecido con la respuesta al mensaje comparar que se envía a un libro. El resultado cambia de valor cuatro veces:

- primer valor de *resultado* es *-1*
- segundo valor de *resultado* es *1*
- tercer valor de *resultado* es *1*
- cuarto valor de *resultado* es *0*

La forma de comparar por el tamaño de un libro con otro, es considerar su altura primero, en caso de no poder definir una resultado claro se considera el ancho de los libros para definir un resultado final.

El resultado es un valor entero que puede ser: -1, 0 o 1. -1 para indicar que el resultado de un libro con otro es menor, es decir el primer libro tienen tamaño menor que el segundo, 0 para indicar que los dos libros tienen igual tamaño y 1 para indicar que el primer libro tiene tamaño mayor al segundo.

Implementa los métodos de la Clase Libro

5. **Preguntas** Una de las preocupaciones de todo estudiante es responder preguntas. Bueno ahora la pregunta es: ¿Qué es una pregunta? ¿Qué tiene? Si te dieran la tarea de hacer preguntas, ¿qué considerarías importante en toda pregunta? responde la pregunta a través de un modelo ;)

   Bueno, tu modelo debe estar adecuado para en el futuro formar un banco de preguntas, por lo que cada pregunta debería:

   a) mostrarse

   b) dada una respuesta *resp* indicar si coincide con la respuesta correcta

   c) dar la respuesta correcta

   Considera el siguiente modelo de la Clase Pregunta:

```
1  class Pregunta{
2     private String enunciado;
3     private String respuesta;
4     public Pregunta(String enunciado, String respuesta){
5        this.enunciado = enunciado;
6        this.respuesta = respuesta;
7     }
8     /**
9     *  Metodo que devuelve el enunciado de la pregunta
10    **/
11    public String mostrar(){
12       return "";
13    }
14    public boolean verificar(String respuesta){
15       return false;
16    }
17    public String mostrarRespuesta(){
18       return "";
19    }
20 }
```

   Para comprender mejor el modelo y cómo éste se usa, te presentamos el siguiente pedazo de código que muestra un ejemplo particular:

```
Pregunta preg1, preg2, preg3;
String pregunta;
boolean resultado;
preg1 = new Pregunta("Las clases modelan....", "tipo de objeto");
preg2 = new Pregunta("Que se evaluan?", "expresiones");
preg3 = new Pregunta("La instancia de una clase tambien se denomina...", "objeto");
pregunta = preg1.mostrar();
pregunta = preg2.mostrar();
pregunta = preg3.mostrar();
resultado = preg1.verificar("clases");
resultado = preg3.verificar("objeto");
```

Se han declarado tres objetos de clase **Pregunta**: *preg1, preg2 y preg3.*

preg1 representa a la pregunta cuyo enunciado es *"Las clases modelan...."* y la respuesta correcta es *"tipo de objeto"*.

preg2 representa a la pregunta cuyo enunciado es *"Que se evaluan?"* y la respuesta correcta es *.exˆpresiones"*.

preg3 representa a la pregunta cuyo enunciado es *"La instancia de una clase tambien se denomina..."* y la respuesta correcta es *.oˆbjeto"*.

Al finalizar este pedazo de código la pregunta es establecida con la respuesta al mensaje mostrar que se envía a una Pregunta. La pregunta cambia de valor tres veces:

- primer valor de *pregunta* es *"Las clases modelan...."*
- segundo valor de *pregunta* es *"Que se evaluan?"*
- tercer valor de *pregunta* es *"La instancia de una clase tambien se denomina..."*

Por otro lado, cuando se envía el mensaje verificar se obtendrá respuestas que se almacenan en resultado:

- primer valor de *resultado* es *false*
- segundo valor de *resultado* es *true*

Implementa los métodos de la Clase Pregunta

6. **Ecuaciones** Las ecuaciones son un tema básico dentro la matemática, ocupan un lugar especial las ecuaciones de segundo grado que tienen la forma:

   $ax^2 + bx + c = 0$

   Para cualesquier coeficiente a, b y c. Considerando esta ecuación se puede encontrar las raices que de x, de acuerdo a la siguiente fórmula:

   $-b \pm \frac{\sqrt{b^2-4ac}}{2a}$

   Para esta versión considera que las raices que se pueden calcular son reales.

   En este sentido se tiene el siguiente modelo:

```
class EcuacionSegundoGrado{
  private double coeficienteA;
  private double coeficienteB;
  private double coeficienteC;
  private double raiz1;
  private double raiz2;

  public EcuacionSegundoGrado(double coeficienteA,
                              double coeficienteB,
                              double coeficienteC ){
    this.coeficienteA = coeficienteA;
    this.coeficienteB = coeficienteB;
    this.coeficienteC = coeficienteC;
    raiz1             = 0.0;
```

```
15        raiz2              = 0.0;
16    }
17    public String  calcularRaices(){
18        return "";
19    }
20    public double getRaiz1(){
21        return 0.0;
22    }
23    public double getRaiz2(){
24        return 0.0;
25    }
26 }
```

Considera el siguiente escenario:

```
1  EcuacionSegundoGrado ecuacion1, ecuacion2, ecuacion3, ecuacion4;
2  String resultado;
3  double raiz1, raiz2;
4  ecuacion1 = new EcuacionSegundoGrado(1.0, 2.0, 1.0);
5  ecuacion2 = new EcuacionSegundoGrado(0.0, 2.0, 1.0);
6  ecuacion3 = new EcuacionSegundoGrado(1.0, 2.0, 20.0);
7  ecuacion4 = new EcuacionSegundoGrado(1.0, -1.0, -2.0);
8  resultado = ecuacion1.calcularRaices();
9  resultado = ecuacion2.calcularRaices();
10 resultado = ecuacion3.calcularRaices();
11 resultado = ecuacion4.calcularRaices();
12 raiz1 = ecuacion1.getRaiz1();
13 raiz2 = ecuacion1.getRaiz2();
14 raiz1 = ecuacion2.getRaiz1();
15 raiz2 = ecuacion2.getRaiz2();
16 raiz1 = ecuacion3.getRaiz1();
17 raiz2 = ecuacion3.getRaiz2();
18 raiz1 = ecuacion4.getRaiz1();
19 raiz2 = ecuacion4.getRaiz2();
```

Se han declarado cuatro objetos de clase **EcuacionSegundoGrado**: *ecuacion1, ecuacion2, ecuacion3 y ecuacion4.*

ecuacion1 representa a la ecuación: $1{,}0x^2 + 2{,}0x + 1{,}0$ por lo que su coeficienteA es *1.0*, su coeficienteB es *2.0* y su coeficienteC es *1.0*

ecuacion2 representa a la ecuación: $2{,}0x + 1{,}0$ por lo que su coeficienteA es *0.0*, su coeficienteB es *2.0* y su coeficienteC es *1.0*

ecuacion3 representa a la ecuación: $1{,}0x^2 + 2{,}0x + 20{,}0$ por lo que su coeficienteA es *1.0*, su coeficienteB es *2.0* y su coeficienteC es *20.0*

ecuacion4 representa a la ecuación: $1{,}0x^2 - 1{,}0x - 2{,}0$ por lo que su coeficienteA es *1.0*, su coeficienteB es *-1.0* y su coeficienteC es *-2.0*

Al finalizar este pedazo de código el resultado es establecido con la respuesta al mensaje calcularRaices que se envía a cada ecuacion. El resultado cambia de valor cuatro veces:

- primer valor de *resultado* es *"Raices calculadas"*
- segundo valor de *resultado* es *"Raices calculadas"*
- tercer valor de *resultado* es *"No se pudo calcular raices reales"*
- cuarto valor de *resultado* es *"Raices calculadas"*

Por otro lado, cuando se envía el mensaje getRaiz1 y getRaiz2 se obtendrá respuestas que se almacenan en raiz1 y raiz2 respectivamente:

- primer valor de *raiz1* es *-1.0*

- primer valor de *raiz2* es *-1.0*
- segundo valor de *raiz1* es *-0.5*
- segundo valor de *raiz2* es *-0.5*
- tercer valor de *raiz1* es *0.0*
- tercer valor de *raiz2* es *0.0*
- cuarto valor de *raiz1* es *2.0*
- cuarto valor de *raiz2* es *-1.0*

Implementa los métodos de la Clase EcuacionSegundoGrado