**Práctica 5. Creación de una aplicación utilizando componentes propios y de biblioteca, utilizando un asistente de diseño.**

**Competencia a desarrollar**

Diseña e implementa componentes y librerías para lograr la reutilización de código.

**Introducción**

Esta práctica vuelve a reforzar, una vez más, la creación de componentes propios creados por el desarrollador pero en esta ocasión uno de tipo no gráfico combinándolo con componentes gráficos elaborados por terceros, resaltando su complementariedad y las implicaciones de usarlos dentro de una aplicación a través de un asistente de diseño.

El manejo de componentes mediante un asistente de diseño como el del ambiente integrado del ´*NetBeans*´, permite apreciar cómo se pueden manipular las propiedades de los componentes disponibles por java y el código necesario para la funcionalidad de los componentes a utilizar en una aplicación, resaltando las ventajas de un desarrollo basado en componentes.

**Correlación con los temas y aplicación en el contexto**

Al crear componentes propios del usuario y su uso a través de su propia biblioteca en una aplicación relaciona a los temas 2.3 y 2.4, también se vincula inevitablemente al tema 2.2 al utilizar componentes de bibliotecas del lenguaje java. Además se agrega algo a lo considerado por el temario que es el uso de componentes desarrollados por terceros. Todo esto se correlaciona dentro de una aplicación.

El contexto donde se aplica es dentro del ámbito de componentes java del paquete swing, así como el uso de programación java para la creación y manejo de componentes disponibles

**Material y equipo necesario**

Equipo de cómputo: Laptop o PC

Software: Aquí se emplea el “IDE” NetBeans con una versión del jdk 1.8.0
Acceso a internet. Para importar componentes
Algoritmo y/o código del método numérico de Newton-Raphson

**Metodología**

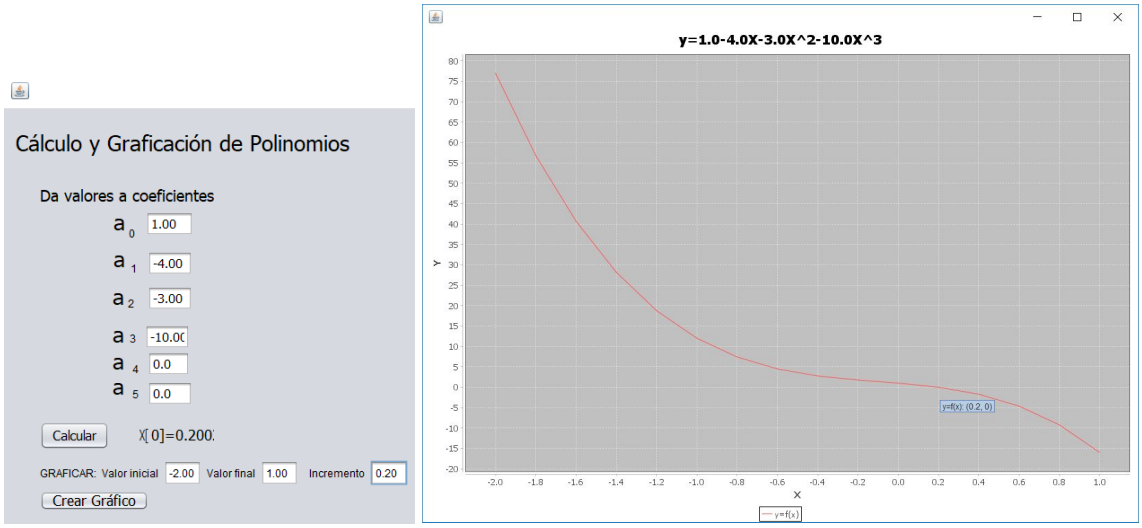
La realización de esta práctica se divide en tres partes. En la primera se expone la descripción de una aplicación que integra el uso del componente que debe crearse, los componentes a importar y los componentes del lenguaje a utilizar para poder integrarlos y manipularlos. Esta aplicación sirve como marco de prueba para los componentes implicados. En la segunda parte se exponen los requerimientos funcionales y el diseño del componente a crear. En la tercera

parte se indican los pasos para llevar a cabo la práctica iniciando por la creación del componente propio, luego la importación del componente de terceros a utilizar y como última parte del desarrollo se hacen uso de los componentes en la aplicación expuesta

detallando la implementación, haciendo que el estudiante complemente parte del código, lo cual propicia su involucramiento. Así mismo debe ir conformando una bitácora de seguimiento de lo que se vaya solicitando durante el desarrollo.

Descripción de la aplicación:

La aplicación donde se involucran los componentes a utilizar tiene como propósito obtener la solución(calcularla) y graficar polinomios de tipo: $a_0+a_1X+a_2X^2+....+a_nX^n$, . Su interfaz y resultados de una ejecución se muestra a continuación:




Los componentes involucrados son:

I. Componente propio del usuario. Encargado principalmente de calcular la solución del polinomio y otras responsabilidades, como por ejemplo desplegarlo en la forma en que se muestra en la parte superior del gráfico y de calcular el valor de la variable 'Y' dado un valor dela variable 'X'.
II. Componente de terceros (externo). Responsable de crear la gráfica correspondiente del polinomio.
III. Componentes propios de java sofisticados como el JFormattedTextField que proporciona una manera práctica de leer datos de diversos tipos.

Descripción de la funcionalidad y diseño del componente propio, denominado "***Polinomio***""

Funcionalidad: Dado un polinomio de grado n de la forma:

$$a_0+a_1X+a_2X^2+....+a_nX^n$$

Su principal función es obtener su solución. Para el caso de ser de grado uno solo se debe regresar un valor de X, cuando es de grado dos se deben obtener los valores de $X_1$ y $X_2$, pudiendo ser reales o complejos, para un grado mayor se debe ir obteniendo el valor uno por uno mediante un algoritmo de métodos numéricos.

Otras funciones son:

Regresar el polinomio formado como una cadena de caracteres.

Obtener el valor de 'Y' (valor de polinomio) para un valor dado de 'X'.

Obtener su derivada.

Diseño del componente "Polonomio"

| Elemento | Descripción | Característica | Implementación |
| --- | --- | --- | --- |
| Coeficientes | Valores de $a_0, a_1, a_{2...}\ a_n$ | Pueden ser de tipo flotante o doble, positivos o negativos. Deben proporcionarse por lo menos dos valores | Un arreglo de tipo Double |
| Variables dependientes | Resultados de X, también conocidas como raíces del polinomio. | Obtenidos mediante la aplicación de los algoritmos utilizados para la solución | Métodos que regresan la solución utilizando los algoritmos. |

| Interface | Función | Parámetros |
| --- | --- | --- |
| setCoeficientes | Establece los valores de los coeficientes | Un arreglo de valores numéricos de tipo doublé |
| getCoeficientes | Regresa un conjunto de valores de tipo doublé de dos o tres elementos correspondiente a los valores de los coeficientes, | ' |
| getGrado | Regresa un valor numérico entero correspondiente al grado del polinomio(1, 2,3,..) | ' |
| getY | Regresa el valor de la evaluación del polinomio dado un valor de X. | Un valor de tipo doublé |
| getDerFx | Regresa el cálculo de la derivada del polinomio Fx para un valor dado de 'X' | Un valor de tipo doublé(valor de 'X') |

Desarrollo

I. Creación del componente Polinomio

1. Creación de la clase del componente Polinomio.

a) Dentro de NetBeans crea un nuevo proyecto java de tipo biblioteca (Library), puedes llamarlo: 'comp_ polinomio', dentro de ahí crea un paquete llamado 'polinomio'

b) Dentro del paquete polinomio crea una clase Java llamada 'Polinomio' que implemente la interface 'Serializable'

2. Creación de los atributos y constructores de la clase Polinomio.

a) Creación de los atributos

```
protected double coeficiente[];
protected static int       MAX_GRADO=3;//para este caso muestra
protected static int       MIN_GRADO=1;
protected int              grado;
```

b) Crea al constructor que debe tener todo componente y dentro de ahí el código:

```
coeficiente = new double[MAX_GRADO+1];
   for(int c=0; c<=MAX_GRADO;c++)
     coeficiente[c]= 0.0d;
```

c) Crea otro constructor que tenga como parámetro un arreglo de elementos de tipo 'double' y el código siguiente conde la variable coeficiente (sin el this) es el parámetro comunicado

```
if(coeficiente.length>1)
     this.coeficiente = coeficiente;
  else
    throw new Exception("Número de coeficientes debe ser > 1");
```

3. Creación de los métodos de actualización y acceso de los tributos

a) Actualización-fijación de valores de los coeficientes.

```
public void setCoeficiente(double coefs[])
{ grado = coefs.length-1;
  //Asegura el minimo de coeficientes
  if(coefs.length>1)
    coeficiente = coefs;
}
```

b) Obtención del grado del polinomio

```
public int getGrado()
{
   return grado;
}
```

c) ? Realiza el método que sirve para recuperación de valores de los coeficientes

4. Creación de los métodos que manipulan los atributos

a) Obtención de 'Y' para un valor dado de 'X'

```
public double   getY(double x)
{ double y=0.0;
  for(int c=0;c<=grado;c++)
    y+= coeficiente[c]*Math.pow(x,c);
  return y;
}
```

b) Calculo de la derivada del polinomio para un valor dado de 'X'

```
public double getDerFx(double x)
    { double dx=0;
      for(int c=1;c<grado; c++)
        dx+= c*coeficiente[c]*Math.pow(x,c-1);
      return dx;
    }
```

c) Regresa una cadena con el polinomio formado como se indica abajo, completa el código

```
public String getPolinomio()
{ // Regresa una cadena representado al polinomio, como por ejemplo
   // 1.0-3.0X+2.0X^2
  String polinomio="";
  // ? realiza el código requerido
  return polinomio;
}
```

5. Obtención de las raíces del polinomio según su grado

a) Encabezado y código para cuando es de grado UNO

```
public double[] getRaices() throws Exception {
double x[]=null;
     if(grado==1)
       {//Ecuacion lineal
        // Calculo de X
         x = new double[grado];
        if(coeficiente[0]!=0)
          x[0]= -1*coeficiente[0]/coeficiente[1];
         else
          throw new Exception("Coeficiente de X debe debe ser !=0");
       }
```

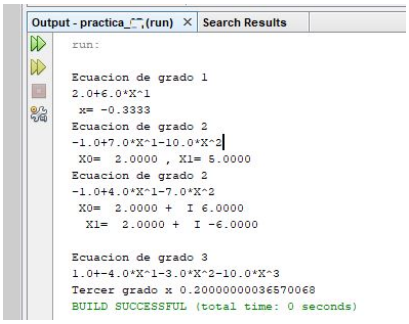
b) Parte del código para cuando es de grado DOS, complementa para el caso de valores imaginarios indicado por los símbolos ?

```
else if(grado==2)
      { // Ecuacion cuadrática
       // Regresa las raices de la ecuación X[0] y X(1) si son reales
       // Si son imaginarias: x[0] es el valor real, y x[1] y x[2] los valores imaginarios
       //Usando la fórmula general
        x = new double[4];
        double a = coeficiente[2];
        double b = coeficiente[1];
        double c = coeficiente[0];
        double rad= Math.pow(b,2.0)- 4*a*c;
        if(rad>=0) // Raices reales
        {
         x[0] = (-b + Math.sqrt(rad))/(2*a);
         x[1] = (-b- Math.sqrt(rad))/(2*a);
        }
        else  // Raices imaginarias
       { x[0] = x[2]= -b/(2*a); // parte real
         x[1] = ? // Parte imaginaria +
         x[3] = ? //Parte imaginaria -
       }
      }
```

c) Parte del código para cuando es de grado TRES o mayor, ocupando el método de Newton-Raphson, aquí debes implementar el algoritmo o adaptar el código (que hayas hecho en la materia de Métodos Numéricos), regresando los valores de las raíces en X[0], x[1], X[2], X[3] ….., he aquí parte del código :

```
else if(grado>2)
     { //Utilizando el Método Newton-Raphson
       double derivada;
       final int MAX_ITERA =100;
       final double PRECISION = 0.001;
        x = new double[grado];
        double error=9999.0, xi=0, x1=0;
        int i=0,n=0;
        for(n=0;n<grado-2;n++){
         i=0;
         do{ xi = x1;
           x1= xi- getY(xi)/getDerFx(xi);
```

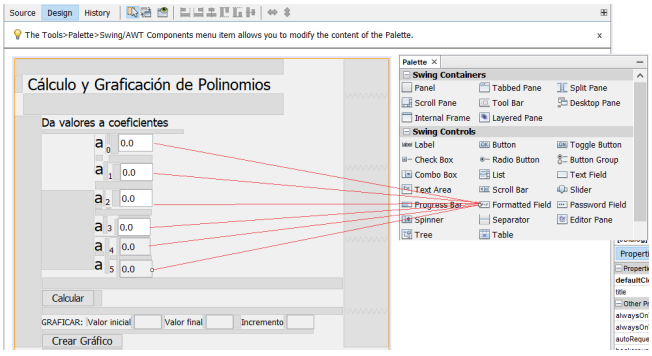
6. Crea una clase de prueba, usa los polinomios abajo mostrados debiéndote dar los valores que ahí se indican




7. Genera el *.jar y agrégalo a la biblioteca creada en la práctica 4.

II. Importar el componente externo ´para graficar el polinomio
   1. Accede a la dirección: http://www.jfree.org/jfreechart/api.html y lee las indicaciones para descargarlo
   2. Descarga los elementos indicados y desempaqueta el archivo jfreechart-1.0.19.zip
   3. Revisa su api de los elementos que la conforman en http://www.jfree.org/jfreechart/api/javadoc/index.html

III. Crear la aplicación con el asistente de NetBeans
   1. Crea un nuevo proyecto java, puedes llamarle practica_5 y dentro de él crea un paquete fuente llamado aplicación. Ahora crea una clase de tipo JDialog Form, llámala 'Aplicación'.
   2. En el modo de diseño agrega los elementos mostrados abajo, indicando con líneas rojas que los campos de edición son de tipo Formatted Field, al igual que lo que va después de valor inicial , valor final e incremento. Lo demás son etiquetas(JLabel) y dos botones (JButton) uno para activar el calculo y otro para graficar. Después del botón [Calcular] existe una etiqueta sin nada de texto, llámala raíces que es donde se mostrará el resultado del cálculo.

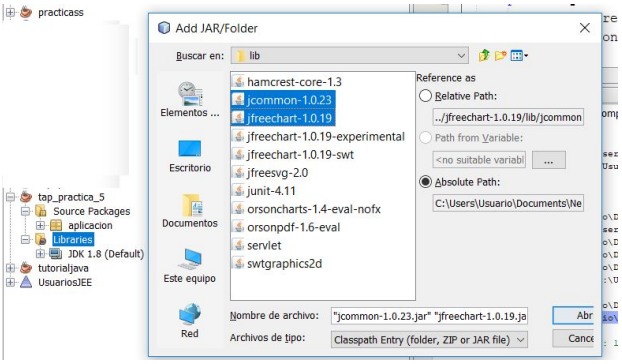



3. Renombra los campos de edición(JFormatted Field) que están después de las etiquetas $a_{0,}$ $a_{1,}$ …. como a0, a1, a2,….
4. Los campos que van después de botón [Graficar] son útiles para indicar el rango de valores a graficar, renómbralos como vInicial,vFinal e inc , respectivamente.

5. En el modo de código fuente, agrega los atributos a utilizar e inícialos en el constructor, como se indica

```
private int MAX_NUM_COEF = 5;
    private Polinomio polinomio;
    private double  coef[];
    private boolean calculo;
    private JFreeChart grafico;
    public Aplicacion(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        coef = new double[MAX_NUM_COEF];
        calculo = false;
        a0.setValue(0.0);
        a1.setValue(0.0);
        a2.setValue(0.0);
        a3.setValue(0.0);
        vInicial.setValue(0.0);
        vFinal.setValue(0.0);
        vInc.setValue(0.0);
    }
```

6. Agrega en la biblioteca del proyecto practica_5, los componentes a utilizar : Polinomio y JFreeChart, para el caso del segundo agrega los dos elementos mostrados desde la carpeta lib de JFreeChart, previamente descargada




7. Para cada campo a0,a1 ,… agrega en su método correspondiente a su evento de acción lo siguiente

```
private void a0ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        coef[0]= valor((Number)a0.getValue());
        if(coef[0]==0.0)
        {
          a0.requestFocus();
          JOptionPane.showMessageDialog(this, "El valor del coeficiente a0 debe ser !=0");
        }
        else
          a1.requestFocus();


    }
```

Hazlo para cada campo, asignado su valor respectivo a x[0], x[1],…
El método valor() regresa el valor doble del campo pasado como parámetro. Tienes que realizarlo.

8. Dentro del método correspondiente al evento de acción asociado al botón [Calcular] agrega lo mostrado a continuación

```
private void jButton1ActionPerformed1(java.awt.event.ActionEvent evt) {
     String tRaices=""; // Texto a mostrar de las raices
     double vRaices[]=null; // Valores de las raices
     double coefs[]  = null; // Valores de los coeficientes
     int nRaices=0; // numero de raices obtenidas
     int nCoef = 0; // numero de coeficientes
     polinomio = new Polinomio();
     //Valida que sean los coeficientes minimos para una ecuacion lineal
     calculo= verificaValores();
     if(calculo)
     {    nCoef = 2;
         //vRaices = new double[1];
         if(coef[2]!=0)
         {

             nCoef++;
           //  vRaices = new double[4];
         }
         if(coef[3]!=0)
         {
            // vRaices = new double[3];
            nCoef++;
         }
       coefs= new  double[nCoef];
       for(int c=0;c<nCoef;c++)
         coefs[c] = coef[c];
       // Se fijan los valores de los coeficientes
       polinomio.setCoeficiente(coefs);

         try {
             vRaices = polinomio.getRaices();
             nRaices = polinomio.getRaices().length;
         } catch (Exception ex) {
             Logger.getLogger(Aplicacion11.class.getName()).log(Level.SEVERE, null, ex);
         }
       if(polinomio.getGrado()==1)
          tRaices = String.format(" X= %6.4f",vRaices[0]);
       else
       if(polinomio.getGrado()==2)
        if(nRaices==2)
           tRaices= String.format(" X0=  %6.4f , X1= %6.4f ",vRaices[0],vRaices[1]);
         else
           tRaices= String.format(" X0=%6.4f + I %6.4f\t \t X1=  %6.4f - I %6.4f\t",vRaices[0],vRaices[1],vRaices[0],vRaices[1]);
       else
         if(polinomio.getGrado()>2)
           for(int r=0;r<nRaices;r++)
             tRaices += String.format("X[%5d]=%6.4f",r,vRaices[r]);

     }
     if(tRaices!="")
     {
       raices.setText(tRaices);
       vInicial.setEditable(true);
       vFinal.setEditable(true);
       vInc.setEditable(true);

     }
    }
```

9. Realiza un análisis de lo que realiza el código respecto al uso del componente ‘Polinomio’ y transcríbelo en tu bitácora de seguimiento.

10. Para el caso del botón [Graficar] tienes que incluir en su método correspondiente de su evento de acción lo siguiente, solo hace falta darle algunas propiedades al elemento ‘fgrafico’  de visibilidad, tamaño y ubicación  antes de cerrar el bloque del método.

```
    private void creaGraficoActionPerformed(java.awt.event.ActionEvent evt) {
            // TODO add your handling code here:


     if(valor((Number)vInicial.getValue())!=0 && valor((Number)vFinal.getValue())!=0 && valor((Number)vInc.getValue())!=0 )
     {

     XYSeries serieXY = new XYSeries("y=f(x)");
     double vf= valor((Number)vFinal.getValue());
     double vi= valor((Number)vInicial.getValue());
     double inc= valor((Number)vInc.getValue());
     while(vi<vf)
     { serieXY.add(vi, polinomio.getY(vi));
       vi+=inc;
      }
      XYDataset datosXY = new XYSeriesCollection(serieXY);
      grafico = ChartFactory.createXYLineChart(
            "y="+polinomio.getPolinomio(), "X", "Y", datosXY,
            PlotOrientation.VERTICAL, true, true, false);
        ChartFrame fgrafico = new ChartFrame("", grafico);
```

11. Realiza un análisis de lo que realiza el código respecto al uso del componente ‘JFreeChart’ y transcríbelo en tu bitácora de seguimiento.

**Sugerencias didácticas**

Documenta el código de tu aplicación

Lleva una bitácora de seguimiento

Realiza la practica en grupos de trabajo de 2 a 4 estudiantes.

Comparte con tus compañeros lo realizado

**Reporte del estudiante**

Se deberá presentar pruebas de la ejecución

Se entregará:

La bitácora de seguimiento
Código fuente del componente y de la aplicación
Componente 'Polinomio'

**Bibliografía**

1. Deitel y Deitel. Como programar en Java. Prentice Hall.Septima Edición, 2008
2. Oracle Documentación, The Java™ Tutorials. https://docs.oracle.com/javase/tutorial/uiswing/components/index.html. Último acceso 16/11/18
3. http://www.jfree.org/jfreechart/ Último acceso 4/12/2018.
4. http://www.jfree.org/jfreechart/api/javadoc/index.html Último acceso 4/12/2018