Reporte de Código(COD-1).

Materia:

Modelado Orientado a Objetos.

Proyecto:

Marco con funciones de cálculo para operaciones estadísticas.

Fecha: *junio 2022*

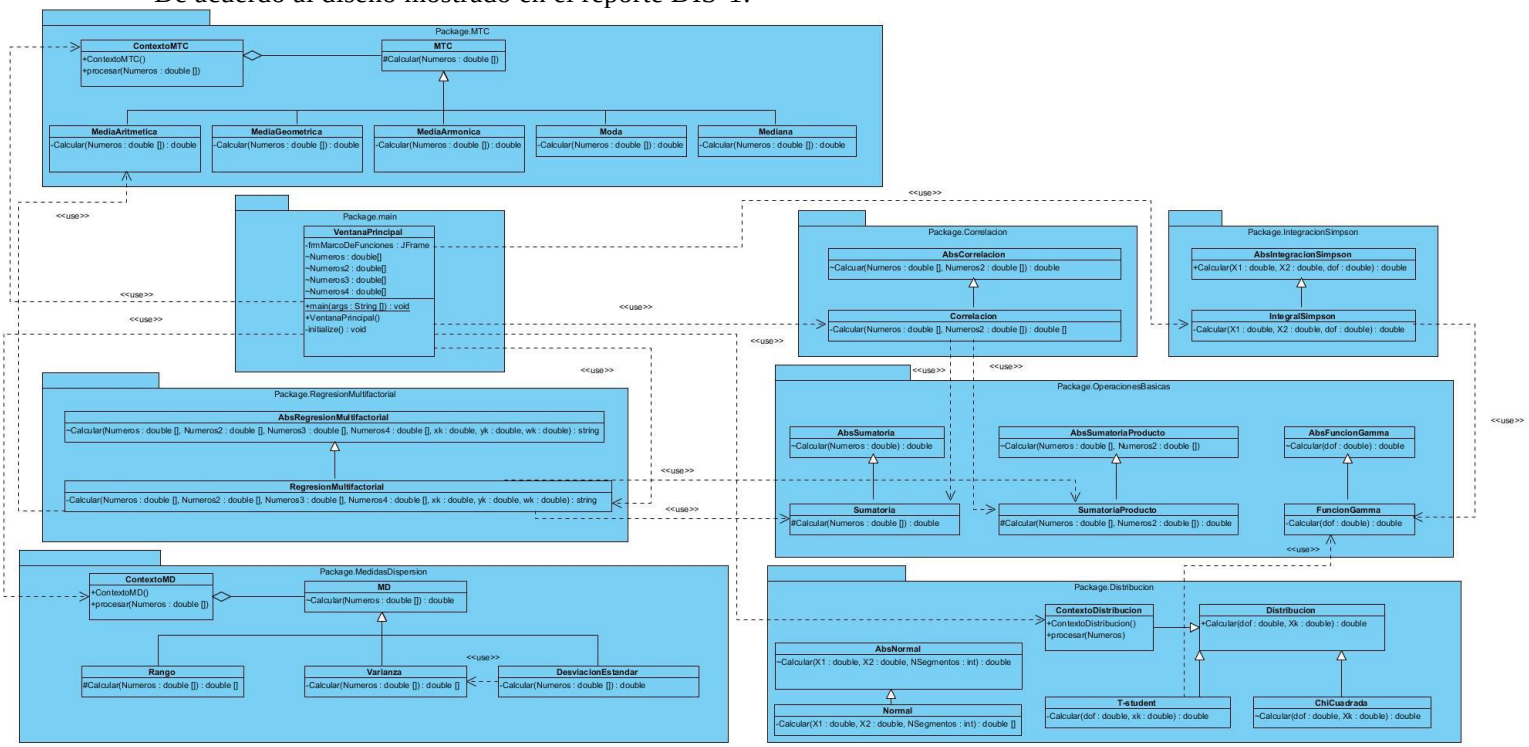
Alumno:

Edgar Valentin Ruiz Padilla

Maestría en Ciencias Computacionales.
Ingeniería de Software

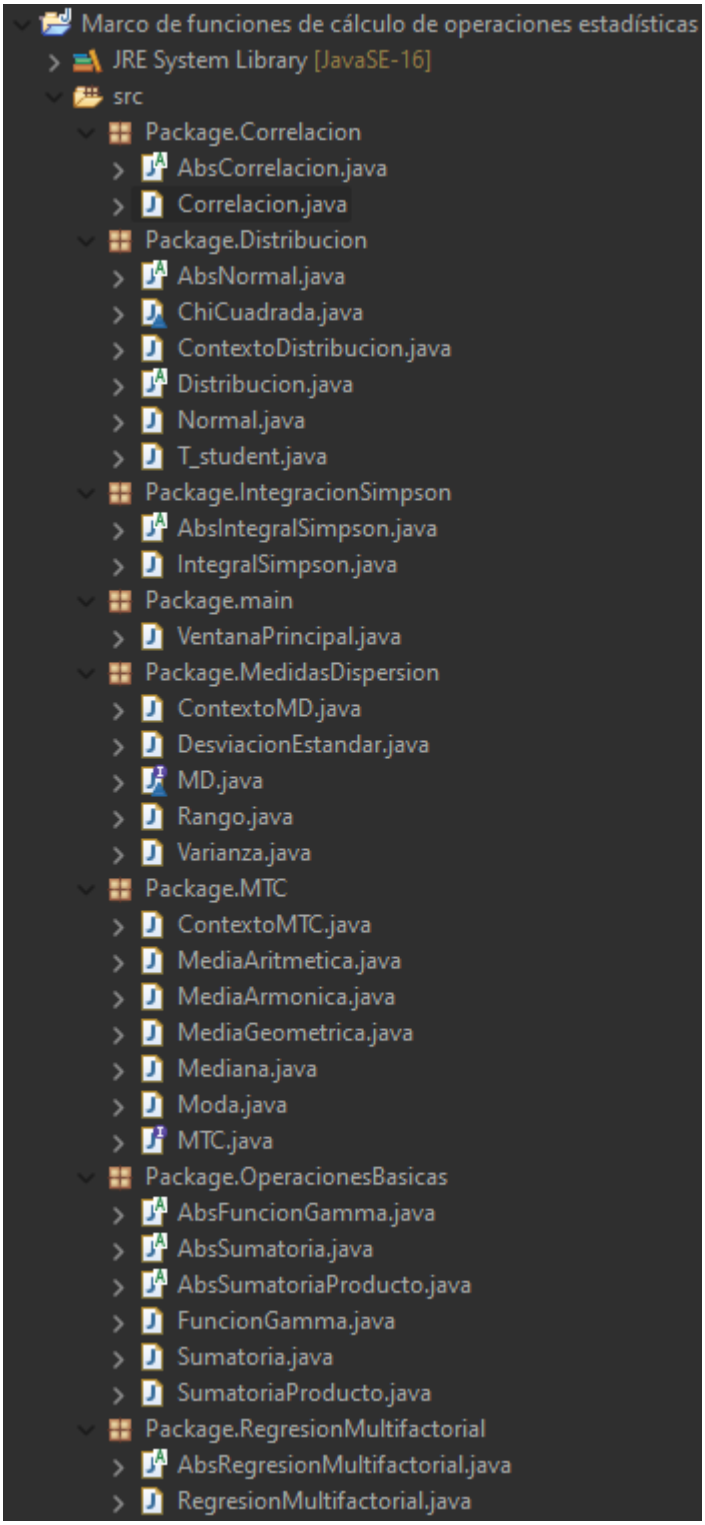
Reporte de Código

El presente documento que contiene el código fuente del programa de computadora, se incluye las librerías utilizadas y el código ejecutable, se tiene como anexo 2 videos de ejecución del programa con nombres: Marco de funciones#1.mkv y Marco de funciones#2.mkv
De acuerdo al diseño mostrado en el reporte DIS-1:



Se construye la siguiente estructura de clases:

Maestría en Ciencias Computacionales.
Ingeniería de Software



Package.Correlacion

Clase AbsCorrelacion

```
package Package.Correlacion;

public abstract class AbsCorrelacion {
	public abstract double calcular(double Numeros[],double Numeros2[]);
}
```

Clase Correlacion

```
package Package.Correlacion;
import Package.OperacionesBasicas.*;

public class Correlacion extends AbsCorrelacion{
	@Override
```

```
	public double calcular(double[] Numeros, double[] Numeros2) {
		// TODO Auto-generated method stub
		AbsSumatoria Sumatoria=new Sumatoria();
		double SUMX = Sumatoria.calcular(Numeros2);
		double SUMY =Sumatoria.calcular(Numeros);
		AbsSumatoriaProducto SumatoriaPrducto=new SumatoriaProducto();
		double SUMXX = SumatoriaPrducto.calcular(Numeros, Numeros2);
		double SUMXY = SumatoriaPrducto.calcular(Numeros, Numeros2);
		double SUMYY = SumatoriaPrducto.calcular(Numeros2, Numeros2);
		int n = Numeros.length;
		double sum1 = (n * SUMYY) - (SUMY * SUMY);
		double sum2 = (n * SUMXX) - (SUMX * SUMX);
		double sum3 = Math.sqrt(sum1 * sum2);
		return ((n * SUMXY) - (SUMX * SUMY)) / sum3;
	}

}
```

Package.Distribucion

Clase AbsNormal

```
package Package.Distribucion;

public abstract class AbsNormal{
	public abstract double[] calcular(double X1,double X2, int NSegmntos);
}
```

Clase ChiCuadrada

```
package Package.Distribucion;

class ChiCuadrada extends Distribucion{

	@Override
	public double calcular(double dof, double X1) {
		// TODO Auto-generated method stub
		return 0;
	}

}
```

Clase ContextoDistribucion

```
package Package.Distribucion;

public class ContextoDistribucion {
	private Distribucion strategy;

	public ContextoDistribucion(Distribucion strategy) {
		this.strategy = strategy;
	}
	public double procesar(double dof, double Xk) {
      return strategy.calcular(dof, Xk);
   }

}
```

Clase Distribucion

```
package Package.Distribucion;

public abstract class Distribucion {
	public abstract double calcular(double dof,double xk);
}
```

Clase Normal

```
package Package.Distribucion;

public class Normal extends AbsNormal{
	double DistribucionT[];
	@Override
	public double[] calcular(double X1, double X2, int NSegmntos) {
		// TODO Auto-generated method stub
		DistribucionT = new double[NSegmntos + 1];
		double Avance = (X2 - X1) / NSegmntos;
		for (int a = 0; a <= NSegmntos; a++) {
			DistribucionT[a] = X1 + (a * Avance);
			// System.out.println(DistribucionT[a]);
	}
		return DistribucionT;
	}
}
```

Maestría en Ciencias Computacionales.
Ingeniería de Software

Clase T_student

```
package Package.Distribucion;
import Package.OperacionesBasicas.*;

public class T_student extends Distribucion{

	@Override
	public double calcular(double dof, double Xk) {

		double Fx = 0;
		Fx = 1 + (Math.pow(Xk, 2) / dof);
		Fx = Math.pow(Fx, -((dof + 1) / 2));
		AbsFuncionGamma FuncionGamma=new FuncionGamma();
		Fx = Fx * FuncionGamma.calcular((dof + 1) / 2) / ((Math.pow(dof * Math.PI, .5)) * FuncionGamma.calcular(dof
/ 2));
		return Fx;
		// TODO Auto-generated method stub
	}

}
```

Package.IntegracionSimpson

Clase AbsIntegralSimpson

```
package Package.IntegracionSimpson;

public abstract class AbsIntegralSimpson {
	public abstract double calcular(double X1,double X2,double dof);
}
```

Clase IntegralSimpson

```
package Package.IntegracionSimpson;
import Package.Distribucion.*;
public class IntegralSimpson extends AbsIntegralSimpson{

	@Override
	public double calcular(double X1, double X2, double dof) {
		// TODO Auto-generated method stub
		AbsNormal Distribucion =new Normal();
		double DistribucionT[]=Distribucion.calcular(X1, X2, 100);
		Distribucion Fx = new T_student();
		double Integral=Fx.calcular(dof, X1);//Verificar X1=0????
		int NumSeg=100;
		for(int a=1;a<=NumSeg;a+=2) {
			Integral=Integral+(4*Fx.calcular(dof, DistribucionT[a]));
			//System.out.println(a*Avance);
			//System.out.println(4*Fx(DistribucionT[a])*Avance/3);
		}
		for(int a=2;a<NumSeg;a+=2) {
			Integral=Integral+(2*Fx.calcular(dof, DistribucionT[a]));
			//System.out.println(a*Avance);
			//System.out.println(2*Fx(DistribucionT[a])*Avance/3);
		}
		Integral=Integral+Fx.calcular(dof, DistribucionT[NumSeg]);
		double Avance=(X2-X1)/NumSeg;
		Integral=Integral*Avance/3;
		//System.out.println(Fx(DistribucionT[NumSeg])*Avance/3);
		return Integral;
	}

}
```

Package. main

Clase VentanaPrincipal

```
package Package.main;
import javax.swing.JOptionPane;
import Package.MTC.*;
import Package.MedidasDispersion.*;
import Package.Correlacion.*;
import Package.RegresionMultifactorial.*;
import Package.Distribucion.*;
import Package.IntegracionSimpson.*;

public class VentanaPrincipal {
	static int funcion=0,Nseg=0;
```

```
	static double X1=0,X2=0,Xk=0,dof=0;
	public static void main(String[] args) {
		double conjunto1[],conjunto2[],conjunto3[],conjunto4[];
		// TODO Auto-generated method stub
		JOptionPane.showMessageDialog(null, "Marco de funciones de calculo de operaciones estadisticas");
		MenuPrincipal();
		switch (funcion) {
			case 1://1.-Media Aritmética
				conjunto1=new double[NumerosConjunto(1)];
				//JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
				for(int i=0;i<conjunto1.length;i++) {
					conjunto1[i]=Nums(i+1,1);
					//JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
				}
				ContextoMTC contexto;
				contexto=new ContextoMTC(new MediaAritmetica());
				double MediaAritmetica=contexto.procesar(conjunto1);
				JOptionPane.showMessageDialog(null, "La media aritmetica es: "+MediaAritmetica);
//Comprobando
				break;
			case 2://2.-Media Geométrica
				conjunto1=new double[NumerosConjunto(1)];
				//JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
				for(int i=0;i<conjunto1.length;i++) {
					conjunto1[i]=Nums(i+1,1);
					//JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
				}
				ContextoMTC contexto2;
				contexto2=new ContextoMTC(new MediaArmonica());
				double MediaArmonica=contexto2.procesar(conjunto1);
				JOptionPane.showMessageDialog(null, "La media Armonica es: "+MediaArmonica);
//Comprobando
				break;
			case 3://3.-Media Armónica
				conjunto1=new double[NumerosConjunto(1)];
				//JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
				for(int i=0;i<conjunto1.length;i++) {
					conjunto1[i]=Nums(i+1,1);
					//JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
				}
				ContextoMTC contexto3;
				contexto3=new ContextoMTC(new MediaGeometrica());
				double MediaGeometrica=contexto3.procesar(conjunto1);
				JOptionPane.showMessageDialog(null, "La media aritmetica es: "+MediaGeometrica);
//Comprobando
				break;
			case 4://Moda
				conjunto1=new double[NumerosConjunto(1)];
				//JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
				for(int i=0;i<conjunto1.length;i++) {
					conjunto1[i]=Nums(i+1,1);
					//JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
				}
				ContextoMTC contexto4;
				contexto4=new ContextoMTC(new Moda());
				double Moda=contexto4.procesar(conjunto1);
				JOptionPane.showMessageDialog(null, "La moda es: "+Moda); //Comprobando
				break;
			case 5://Mediana
				conjunto1=new double[NumerosConjunto(1)];
				//JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
				for(int i=0;i<conjunto1.length;i++) {
					conjunto1[i]=Nums(i+1,1);
					//JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
				}
```

```
                                        ContextoMTC contexto5;
                                        contexto5=new ContextoMTC(new Mediana());
                                        double Mediana=contexto5.procesar(conjunto1);
                                        JOptionPane.showMessageDialog(null, "La mediana: "+Mediana); //Comprobando
                                        break;
                                case 6://"6.-Desviación Estándar\n"+
                                        conjunto1=new double[NumerosConjunto(1)];
                                        //JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto1[i]=Nums(i+1,1);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        ContextoMD contexto6;
                                        contexto6=new ContextoMD(new DesviacionEstandar());
                                        double Desviacion=contexto6.procesar(conjunto1);
                                        JOptionPane.showMessageDialog(null, "La Desviacion Estandar es: "+Desviacion);
//Comprobando
                                        break;
                                case 7://"7.-Varianza\n"+
                                        conjunto1=new double[NumerosConjunto(1)];
                                        //JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto1[i]=Nums(i+1,1);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        ContextoMD contexto7;
                                        contexto7=new ContextoMD(new Varianza());
                                        double Varianza=contexto7.procesar(conjunto1);
                                        JOptionPane.showMessageDialog(null, "La varianza es: "+Varianza); //Comprobando
                                        break;
                                case 8://"8.-Rango\n"+
                                        conjunto1=new double[NumerosConjunto(1)];
                                        //JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto1[i]=Nums(i+1,1);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        ContextoMD contexto8;
                                        contexto8=new ContextoMD(new Rango());
                                        double Rango=contexto8.procesar(conjunto1);
                                        JOptionPane.showMessageDialog(null, "El rango es: "+Rango); //Comprobando
                                        break;
                                case 9://"9.-Correlación\n"+
                                        conjunto1=new double[NumerosConjunto(1)];
                                        conjunto2=new double[conjunto1.length];
                                        //JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto1[i]=Nums(i+1,1);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto2[i]=Nums(i+1,2);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        AbsCorrelacion Corr=new Correlacion();
                                        JOptionPane.showMessageDialog(null, "La correlacion es: "+Corr.calcular(conjunto1,
conjunto2)); //Comprobando
                                        break;
                                case 10://"10.-Distribución Normal\n"+
                                        X1 =Double.parseDouble(VerificaEntrada("Ingresa el valor X1:"));
                                        X2 =Double.parseDouble(VerificaEntrada("Ingresa el valor X2:"));
                                        Nseg =Integer.parseInt(VerificaEntrada("Ingresa el Numero de Segmentos"));
                                        AbsNormal Normal =new Normal();
                                        Normal.calcular(X1, X2, Nseg);
                                        JOptionPane.showMessageDialog(null, "La distribucion Normal es: "+Normal.calcular(X1,
X2, Nseg)[0]); //Definir que mostrar en la Dist Normal
```

```
                                        break;
                                case 11://"11.-Distribución T student\n"+
                                        X1 =Double.parseDouble(VerificaEntrada("Ingresa el valor Xk:"));
                                        dof =Double.parseDouble(VerificaEntrada("Ingresa los grados de libertad"));
                                        ContextoDistribucion contexto9;
                                        contexto9=new ContextoDistribucion(new T_student());
                                        double Tstudent=contexto9.procesar(X1, dof);
                                        JOptionPane.showMessageDialog(null, "La integral T student es: " + Tstudent);
//Comprobando
                                        break;
                                case 12://"12.-Regresión Multifactorial\n"+
                                        //conjunto1=new double[6];//Test
                                        conjunto1=new double[NumerosConjunto(1)];
                                        conjunto2=new double[conjunto1.length];
                                        conjunto3=new double[conjunto1.length];
                                        conjunto4=new double[conjunto1.length];
                                        //JOptionPane.showMessageDialog(null, "Cantidad total numeros "+ conjunto1.length);
//Comprobando

                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto1[i]=Nums(i+1,1);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto2[i]=Nums(i+1,2);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto3[i]=Nums(i+1,3);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        for(int i=0;i<conjunto1.length;i++) {
                                                conjunto4[i]=Nums(i+1,4);
                                                //JOptionPane.showMessageDialog(null, "Numero
"+String.valueOf(conjunto1[i])+" guardado"); //Comprobando
                                        }
                                        double wk =Double.parseDouble(VerificaEntrada("Ingresa el valor wk"));
                                        double xk =Double.parseDouble(VerificaEntrada("Ingresa el valor xk"));
                                        double yk =Double.parseDouble(VerificaEntrada("Ingresa el valor yk"));

                                        /* // Testing
                                        conjunto1[0]=345;
                                        conjunto1[1]=168;
                                        conjunto1[2]=94;
                                        conjunto1[3]=187;
                                        conjunto1[4]=621;
                                        conjunto1[5]=255;
                                        conjunto2[0]=65;
                                        conjunto2[1]=18;
                                        conjunto2[2]=0;
                                        conjunto2[3]=185;
                                        conjunto2[4]=87;
                                        conjunto2[5]=0;
                                        conjunto3[0]=23;
                                        conjunto3[1]=18;
                                        conjunto3[2]=0;
                                        conjunto3[3]=98;
                                        conjunto3[4]=10;
                                        conjunto3[5]=0;
                                        conjunto4[0]=31.4;
                                        conjunto4[1]=14.6;
                                        conjunto4[2]=6.4;
                                        conjunto4[3]=28.3;
                                        conjunto4[4]=42.1;
                                        conjunto4[5]=15.3;
                                        double wk =185;
                                        double xk =150;
                                        double yk =45;
                                        */

                                        AbsRegresionMultifactorial Regresion=new RegresionMultifactorial();
```

```
                                        JOptionPane.showMessageDialog(null, "La correlacion es: "+Regresion.calcular(conjunto1,
conjunto2, conjunto3, conjunto4, xk, yk, wk)); //Comprobando

                                        break;
                                case 13://"13.-Integral Simpson\n"+
                                        X1 =Double.parseDouble(VerificaEntrada("Ingresa el valor X1"));
                                        X2 =Double.parseDouble(VerificaEntrada("Ingresa el valor X2"));
                                        dof =Double.parseDouble(VerificaEntrada("Ingresa los grados de libertad"));
                                        AbsIntegralSimpson Integral = new IntegralSimpson();
                                        JOptionPane.showMessageDialog(null, "El valor calculado de la integral es:
"+Integral.calcular(X1, X2, dof)); //Comprobando
                                        break;
                                case 14://"14.-Distribución Chi Cuadrada\n"));
                                        X1 =Double.parseDouble(VerificaEntrada("Ingresa el valor Xk:"));
                                        dof =Double.parseDouble(VerificaEntrada("Ingresa los grados de libertad"));
                                        break;
                                default:
                                        MenuPrincipal();
                                        break;

                        }

        }
        public static String VerificaEntrada (String titulo) {
                String Entrada = "";
                Boolean VEntrada = false;
                while (VEntrada != true) {
                        Entrada = JOptionPane.showInputDialog(titulo);
                        VEntrada = Entrada.matches("[+-]?\\d*(\\.\\d+)?");
                }
                return Entrada;
        }
        public static void MenuPrincipal() {
                funcion=Integer.parseInt (VerificaEntrada("Selecciona la funcion de calculo a realizar:\n"+
                                        "1.-Media Aritmética\n"+
                                        "2.-Media Geométrica\n"+
                                        "3.-Media Armónica\n"+
                                        "4.-Moda\n"+
                                        "5.-Mediana\n"+
                                        "6.-Desviación Estándar\n"+
                                        "7.-Varianza\n"+
                                        "8.-Rango\n"+
                                        "9.-Correlación\n"+
                                        "10.-Distribución Normal\n"+
                                        "11.-Distribución T student\n"+
                                        "12.-Regresión Multifactorial\n"+
                                        "13.-Integral Simpson\n"+
                                        "14.-Distribución Chi Cuadrada\n"));
        }
        public static int NumerosConjunto(int X) {
                return Integer.parseInt (VerificaEntrada("Ingresa la cantidad de numeros del conjunto "+ X));

        }
        public static double Nums(int X, int Y) {
                return Double.parseDouble(VerificaEntrada("Ingresa el numero "+X+" del conjunto "+ Y));
        }
}
```

Package. MedidasDispersion

Clase ContextoMD

```
package Package.MedidasDispersion;

public class ContextoMD {
        private MD strategy;

        public ContextoMD(MD strategy) {
                //super();
                this.strategy = strategy;
        }
        public double procesar(double[] Numeros) {
     return strategy.Calcular(Numeros);
  }
}
```

Clase DesviacionEstandar

```
package Package.MedidasDispersion;
```

```
public class DesviacionEstandar implements MD {

	@Override
	public double Calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		MD varianza=new Varianza();
		double Desviacion=Math.sqrt(varianza.Calcular(Numeros));
		return Desviacion;
	}

}
```

Clase MD

```
package Package.MedidasDispersion;

interface MD {
	abstract double Calcular (double Numeros[]);
}
```

Clase Rango

```
package Package.MedidasDispersion;

public class Rango implements MD {

	@Override
	public double Calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double maximo=Numeros[0],minimo=Numeros[0];
		for (int i = 0; i < Numeros.length; i++){
			if (maximo < Numeros[i])
				maximo = Numeros[i];
			if (minimo > Numeros[i])
				minimo = Numeros[i];
		}
		return maximo-minimo;
	}

}
```

Clase Varianza

```
package Package.MedidasDispersion;
import Package.MTC.*;
public class Varianza implements MD {

	@Override
	public double Calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		MTC Media =new MediaAritmetica();

		double MediaT=Media.calcular(Numeros);
		double Varianza=0, Temp=0;
		int Total=Numeros.length;
		for(int M=0;M<Total; M++) {
			Temp=Temp+Math.pow((Numeros[M]-MediaT),2);
		}
		Varianza=Temp/(Total-1);
		return Varianza;
	}

}
```

Package. MTC

Clase ContextoMTC

```
package Package.MTC;

public class ContextoMTC {
	private MTC strategy;

	public ContextoMTC(MTC strategy) {
		//super();
		this.strategy = strategy;
	}
	public double procesar(double[] Numeros) {
      return strategy.calcular(Numeros);
   }
}
```

Maestría en Ciencias Computacionales.
Ingeniería de Software

Clase MediaAritmetica

```
package Package.MTC;

public class MediaAritmetica implements MTC {

	@Override
	public double calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double Media = 0;
		int Total = Numeros.length;
		for (int M = 0; M < Total; M++) {
			Media = Numeros[M] + Media;
		}
		Media = Media / Total;
		return Media;
	}


}
```

Clase MediaArmonica

```
package Package.MTC;

public class MediaArmonica implements MTC {

	@Override
	public double calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double MediaArm = 0;
		int Total = Numeros.length;
		for (int M = 0; M < Total; M++) {
			MediaArm = (1/Numeros[M] )+ MediaArm;
		}

		MediaArm =Total/MediaArm;
		return MediaArm;
	}

}
```

Clase MediaGeometrica

```
package Package.MTC;

public class MediaGeometrica implements MTC {

	@Override
	public double calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double MediaGeo = 1;
		int Total = Numeros.length;
		for (int M = 0; M < Total; M++) {
			MediaGeo = Numeros[M] * MediaGeo;
		}
		MediaGeo =Math.pow(MediaGeo, (double) 1 / Total);
		return MediaGeo;
	}

}
```

Clase Mediana

```
package Package.MTC;

public class Mediana implements MTC {

	@Override
	public double calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double Mediana=0;
		int Centro=Numeros.length/2;
		if(Numeros.length%2==1) {
			Mediana=Numeros[Centro];
			return Mediana;
		} else {
			return (Numeros[Centro-1]+Numeros[Centro])/2;

		}
	}
```

```
}
```

Clase Moda

```
package Package.MTC;

public class Moda implements MTC {

	@Override
	public double calcular(double[] Numeros) {
		// TODO Auto-generated method stub
		double Moda = 0;
		int MaxRep=0;
		for(int M=0;M<Numeros.length; M++) {
			int Rep=0;
			for(int N=0;N<Numeros.length;N++) {
				if(Numeros[M]==Numeros[N]) {
					Rep++;
				}
				if(Rep>MaxRep) {
				Moda=Numeros[M];
				MaxRep=Rep;
				}
			}
		}
		if (MaxRep>1) {
			return Moda;
		}else {
			return 0;
		}

	}

}
```

Clase MTC

```
package Package.MTC;

public interface MTC {
	public abstract double calcular(double Numeros[]);
}
```

Package. MTC

Clase AbsFuncionGamma

```
package Package.OperacionesBasicas;

public abstract class AbsFuncionGamma {
	public abstract double calcular(double dof);
}
```

Clase AbsSumatoria

```
package Package.OperacionesBasicas;

public abstract class AbsSumatoria{
	public abstract double calcular(double Numeros[]);
}
```

Clase FuncionGamma

```
package Package.OperacionesBasicas;

public class FuncionGamma extends AbsFuncionGamma {

	@Override
	public double calcular(double dof) {
		// TODO Auto-generated method stub
		double Gamma=dof-1;
		if(Gamma==0&&dof<2) {
			Gamma=1;
		}
		if(Gamma==1&&dof<2) {
			Gamma=2;
		}
		if(Gamma==.5&&dof<2) {
			Gamma=Math.sqrt(Math.PI);
		}

		for(int a=2;a<dof;a++) {
			if((dof-a)==.5) {
```

```
                                        Gamma=Math.sqrt(Math.PI)*Gamma*.5;
                                }else {
                                        Gamma=(dof-a)*Gamma;
                                }
                        }
                        return Gamma;
                }

}
```

Clase AbsSumatoriaProducto

```
package Package.OperacionesBasicas;

public abstract class AbsSumatoriaProducto {
        abstract public double calcular(double Numeros[], double Numeros2[]);
}
```

Clase Sumatoria

```
package Package.OperacionesBasicas;

public class Sumatoria extends AbsSumatoria{
        @Override
        public double calcular(double[] Numeros) {
                // TODO Auto-generated method stub
                double SUM = 0;
                for (int M = 0; M < Numeros.length; M++) {
                        SUM = Numeros[M] + SUM;
                }
                return SUM;
        }

}
```

Clase SumatoriaProducto

```
package Package.OperacionesBasicas;

public class SumatoriaProducto extends AbsSumatoriaProducto{

        @Override
        public double calcular(double[] Numeros, double[] Numeros2) {
                // TODO Auto-generated method stub
                double SUMSUM = 0;
                for (int M = 0; M < Numeros.length; M++) {
                        SUMSUM = (Numeros[M] * Numeros2[M]) + SUMSUM;
                }
                return SUMSUM;
        }

}
```

Package. RegresionMultifactorial

Clase AbsRegresionMultifactorial

```
package Package.RegresionMultifactorial;

public abstract class AbsRegresionMultifactorial {
        public abstract String calcular(double Numeros[],double Numeros2[],double Numeros3[],double Numeros4[],double
xk,double yk,double wk);
}
```

Clase RegresionMultifactorial

```
package Package.RegresionMultifactorial;
import Package.OperacionesBasicas.AbsSumatoria;
import Package.OperacionesBasicas.AbsSumatoriaProducto;
import Package.OperacionesBasicas.Sumatoria;
import Package.OperacionesBasicas.SumatoriaProducto;

public class RegresionMultifactorial extends AbsRegresionMultifactorial {
        double[][] MatrizGauss = new double[4][4];
        double r[]=new double[4];
        double Betas[]=new double[4];
        double Zk;

        @Override
        public String calcular(double[] Numeros, double[] Numeros2, double[ ] Numeros3, double[] Numeros4, double xk,
                        double yk, double wk) {
                // TODO Auto-generated method stub
                MatrizGauss(Numeros, Numeros2, Numeros3, Numeros4);
                System.out.println(Numeros[0]);
```

```
		System.out.println(Numeros2[1]);
		System.out.println(Numeros3[2]);
		System.out.println(Numeros4[3]);
		System.out.println(Numeros[4]);
		System.out.println(Numeros[5]);
		System.out.println(Numeros2[0]);
		System.out.println(Numeros2[1]);
		System.out.println(Betas[0]);
		Gauss();
		System.out.println(Betas[0]);
		PrediccionY(xk,yk,wk);
		System.out.println(Betas[0]);
		return toString();
	}

	public void MatrizGauss(double[] Numeros, double[] Numeros2, double[] Numeros3, double[] Numeros4) {
		int N = Numeros.length;
		AbsSumatoria Sumatoria = new Sumatoria();
		AbsSumatoriaProducto SumatoriaPrducto = new SumatoriaProducto();
		MatrizGauss[0][0] = N;

		MatrizGauss[0][1] = Sumatoria.calcular(Numeros);
		MatrizGauss[0][2] = Sumatoria.calcular(Numeros2);
		MatrizGauss[0][3] = Sumatoria.calcular(Numeros3);
		r[0] = Sumatoria.calcular(Numeros4);
		MatrizGauss[1][0] = Sumatoria.calcular(Numeros);
		MatrizGauss[1][1] = SumatoriaPrducto.calcular(Numeros, Numeros);
		MatrizGauss[1][2] = SumatoriaPrducto.calcular(Numeros, Numeros2);
		MatrizGauss[1][3] = SumatoriaPrducto.calcular(Numeros, Numeros3);
		r[1] = SumatoriaPrducto.calcular(Numeros, Numeros4);
		MatrizGauss[2][0] = Sumatoria.calcular(Numeros2);
		MatrizGauss[2][1] = SumatoriaPrducto.calcular(Numeros, Numeros2);
		MatrizGauss[2][2] = SumatoriaPrducto.calcular(Numeros2, Numeros2);
		MatrizGauss[2][3] = SumatoriaPrducto.calcular(Numeros2, Numeros3);
		r[2] = SumatoriaPrducto.calcular(Numeros2, Numeros4);
		MatrizGauss[3][0] = Sumatoria.calcular(Numeros3);
		MatrizGauss[3][1] = SumatoriaPrducto.calcular(Numeros, Numeros3);
		MatrizGauss[3][2] = SumatoriaPrducto.calcular(Numeros2, Numeros3);
		MatrizGauss[3][3] = SumatoriaPrducto.calcular(Numeros3, Numeros3);
		r[3] = SumatoriaPrducto.calcular(Numeros3, Numeros4);
		//
	}

	public void Gauss() {
		int n = 4, i = 0, j = 0, s = 0;
		double d = 0;
		for (i = 0; i < n; i++) {
			for (j = i; j < n; j++) {
				if (i == j) {
					d = MatrizGauss[i][j];
					for (s = 0; s < n; s++) {
						MatrizGauss[i][s] = ((MatrizGauss[i][s]) / d);
					}
					r[i] = ((r[i]) / d);
				} else {
					d = MatrizGauss[j][i];
					for (s = 0; s < n; s++) {
						MatrizGauss[j][s] = MatrizGauss[j][s] - (d * MatrizGauss[i][s]);
					}
					r[j] = r[j] - (d * r[i]);
				}
			}
		}
		for (i = n - 1; i >= 0; i--) {
			double y = r[i];
			for (j = n - 1; j >= i; j--) {
				y = y - Betas[j] * MatrizGauss[i][j];
			}
			Betas[i] = y;
		}
	}


	public  void PrediccionY(double xk, double yk, double wk) {
		 Zk = Betas[0] + Betas[1] * wk + Betas[2] * wk + Betas[3] * yk;
```

```
		}
		// Devolviendo el estado del objeto
			public String toString() {
				String Resultado = "El resultado es: ";
				for (int a = 0; a < 4; a++) {
					Resultado +=("Beta " + a + " es: " + Betas[a]);
				}
				Resultado += (" La proyeccion Zk es: "+ Zk);
				return Resultado;
			}
}
```