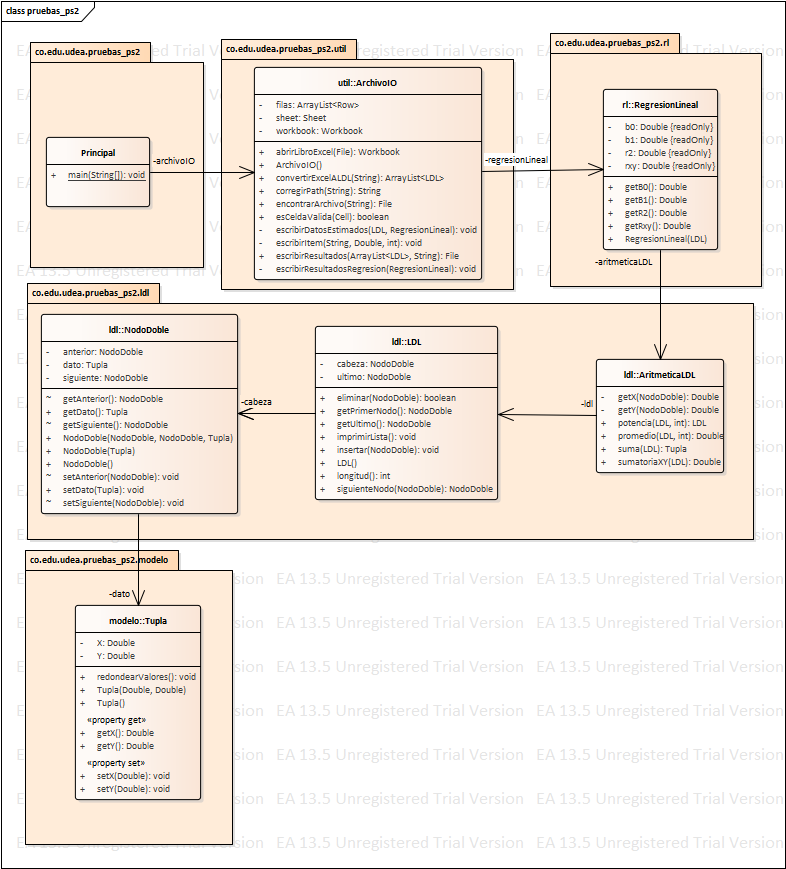
**Operational Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Yoiner Gómez, Frank Castrillón, Jhonatan Orozco | **Date** | 02/09/2017 |
| **Program** | Regresión lineal | **Program #** | 3 |
| **Instructor** | Juan Carlos Marin | **Language** | Java |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario Number** | 1 | **User Objective** | Tener los datos de entrada en un archivo excel que cumpla con el formato establecido (Ver README) | |
| **Scenario Objective** | | Calcular la regresión lineal a partir de un conjunto de datos con el fin de predecir nuevos valores en base a la ecuación encontrada. | | |
| **Source** | **Step** | **Action** | | **Comments** |
| Usuario | 1 | Ejecuta el programa | |  |
| Programa | 2 | Solicita la ruta del archivo con los datos. | | * Los datos deben cumplir con el formato establecido (Ver README) * El mensaje de solicitud se muestra en consola. |
| Usuario | 3 | Ingresa la ruta del archivo | | * La entrada de la ruta se realiza por consola. |
| Programa | 3.1.1 | Notifica que el programa no soporta el tipo de archivo ingresado | | Mensaje de error |
| Usuario | 3.1.2 | Debe ingresar una ruta de un archivo con extensión “.xls” | |  |
| Programa | 3.2.1 | Notifica que el archivo no puede abrirse | | Mensaje de error |
| Usuario | 3.2.2 | Debe cerrar el archivo que el programa está intentando leer. | |  |
| Programa | 3.3.1 | Notifica que el archivo no existe | | Mensaje de error |
| Usuario | 3.3.2 | Debe indicar un archivo que realmente esté almacenado en la ruta especificada. | |  |
| Programa | 3.4.1 | Notifica que el archivo está vacío | | Mensaje de error |
| Usuario | 3.4.2 | Debe ingresar valores al archivo | | Los datos deben cumplir con el formato establecido (Ver README) |
| Programa | 3.5.1 | Notifica que alguna celda no se puede leer | | Mensaje de error |
| Usuario | 3.5.2 | Debe comprobar que todas las celdas del archivo sean de tipo numérico | |  |
| Programa | 3.6.1 | Notifica que no hay X de prueba | | Mensaje de error |
| Usuario | 3.6.2 | Debe especificar por lo menos una columna con datos X de prueba para predecir nuevos valores | |  |
| Programa | 3.7.1 | Notifica que alguna tupla (X,Y) está incompleta | | Mensaje de error |
| Usuario | 3.7.2 | Debe comprobar que todas las tuplas (X,Y) posean un valor especificado | |  |
| Programa | 4 | Notifica la ruta del archivo donde fueron guardados los resultados | | La notificación se hace por consola. |
|  |  |  | |  |

**Functional Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | Yoiner Gómez, Frank Castrillón, Jhonatan Orozco | **Date** | 02/09/2017 |
| **Program** | Regresión lineal | **Program #** | 3 |
| **Instructor** | Juan Carlos Marin | **Language** | Java |



**Logic Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Frank castrillón, Yoiner Gómez, Jhonatan Orozco | Date | 04/09/2017 |
| Program | Regresión Lineal | Program # | 3 |
| Instructor | Juan Carlos Marín | Language | Java |

|  |  |
| --- | --- |
| **Design** | Operational Template |
| **References** | Funcional Template |
|  |  |

|  |
| --- |
| public class Main{  public void main(){  ArchivoIO archivoIO;  Lista <LDL> datos= archivoIO.convertirExcelALDL(nombreArchivo);  archivoIO.generarResultadosExcel(datos,nombreArchivo);  }  } |
| public class ArchivoIO{  Lista<LDL> convertirExcelALDL(String nombreArchivo){  LDL datosRegresion, datosEstimacion;  Tupla t1, t2;  Archivo f=abrirArchivoExcel(nombreArchivo);  while(f.existanFilas()){  while(f.existanColumnas()){  switch(indiceColumna){  case 0:  t1.setX(celda.getValue());  break;  case 1:  t1.setY(celda.getValue());  break;  case 2:  t2.setX(celda.getValue());  NodoDoble n= new NodoDoble(t1);  datosEstimacion.insertar(n);  break;    }  }  NodoDoble n= new NodoDoble(t1);  datosRegresion.insertar(n);  }  datos.insertar(datosRegresion);  datos.insertar(datosEstimacion);  return datos;    }    public void generarResultadosExcel(Listaz<LDL> datos,String nombreArchivo){  LDL datosRegresion=datos.get(0);  Regresion r=new Regresion(datosregresion);  escribirEnExcel("B0", r.getB0());  escribirEnExcel("B1", r.getB1());  escribirEnExcel("r²", r.getR2());  escribirEnExcel("Rxy", r.getRXY());    }  } |
| public class Regresion{    private double b0;  private double b1;  private double r2;  private double rxy;  public Regresion(LDL datosRegresion){  double xavg, yavg, denominador, numerador;  int n= datosRegresion.lenght();  AritmeticaLDL aritmetica= AritmeticaLDL();  Tupla promedio= aritmetica.promedio(datosRegresion);  Tupla sum2=aritmetica.suma(aritmetica.potencia(datosRegresion,2));  Tupla sum=aritmetica.suma(datosregresion);  xavg=promedio.getX();  yavg=promedio.getY();  sumx2=sum2.getX();  sumy2=sum2.getY();  sumx=sum.getX();  sumy=sum.getY();  numerador=aritmetica.sumatoriaXY(datosRegresion)-(n\*xavg\*yavg);  denominador= sumx2-(n\*xavg^2);  b1=numerador/denominador;  b0=yavg- b1\*xavg;  numerador=n\*aritmetica.sumatoriaXY(datosRegresion)- sumx\*sumy;  denominador=( (n\*sumx2)-sumx^2 ) )\*( (n\*sumy2)-sumy^2 ) );  rxy= numerador /sqrt(denominador);  r2=rxy\*rxy;    }  } |