UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

PRIMERA PRÁCTICA LÓGICA Y REPRESENTACIÓN III

POLINOMIOS REPRESENTADOS COMO LISTAS LIGADAS

MANUAL TÉCNICO

Por

CRISTIAN CAMILO TORRES ALZATE

2017

Para la realización de la práctica se usó el lenguaje de programación JAVA a través del entorno de desarrollo integrado NetBeans IDE 8.0.2.

En total, se crearon 7 clases para la elaboración de esta práctica

Lógicas:

- NodoDoble
- LDLRC (Lista doblemente ligada con registro cabeza)
- Termino
- Polinomio
- Main(main)

Visuales:

- Menu
- VistaPrincipal

Nota: Algunos algoritmos fueron tomados de los libros Algoritmia II y Algoritmia III de Roberto Flórez Rueda, para su mayor identificación llevarán un asterisco (*) al lado del nombre.

Clase NodoDoble* //Tomada en su totalidad del libro Algoritmia II (Roberto Flórez Rueda).

Privado:

Object **dato**
NodoDoble **Derecha**
NodoDoble **Izquierda**

Público:

NodoDoble ()*
Constructor vacío.

NodoDoble (Object d) *
Constructor para agregar dato al nodo.

NodoDoble **retornaLD** () *
Retorna el nodo que está a la derecha del nodo que invoca el método.

void **asignaLD** (NodoDoble derecha) *
Modifica la liga derecha del nodo que invoca el método.

NodoDoble **retornaLI** () *
Retorna el nodo que está a la izquierda del nodo que invoca el método.

void **asignaLI** (NodoDoble izquierda) *
Modifica la liga izquierda del nodo que invoca el método.

Object **retornaDato** () *
Retorna el dato que posee el nodo que invoca el método.

void **asignaDato** (Object Dato) *
Modifica el valor que hay en el campo dato del nodo que invoca el método.

Clase LDLRC* //Tomada parcialmente del libro Algoritmia II (Roberto Flórez Rueda).

Privado:

NodoDoble **cabeza**
NodoDoble **primero**
NodoDoble **ultimo**

Público:

LDLRC () *
Constructor vacío.

NodoDoble **retornaLD**() *
Retorna el nodo que está a la derecha del nodo que invoca el método.

NodoDoble **primerNodo** () *
Retorna el primer nodo de la lista.

NodoDoble **ultimoNodo** () *
Retorna el último nodo de la lista.

NodoDoble **nodoCabeza** () *
Retorna el nodo Cabeza de la lista.

NodoDoble **retornaLI** () *
Retorna el nodo que está a la izquierda del nodo que invoca el método.

Void **asignaLI** (NodoDoble izquierda) *
Modifica la liga izquierda del nodo que invoca el método.

Boolean **finRecorrido** (NodoDoble p) *
Retorna verdadero el NodoDoble p ya recorrió la lista.

Boolean **esVacia** () *
Retorna verdadero si la lista está vacía.

NodoDoble **anterior** (NodoDoble p) *
Retorna el nodo anterior a p en la lista que invoca el método.

Int **tamaño** ()
Retorna la cantidad de nodos que posee la lista.

Void **insertar** (Object o, NodoDoble s) *
Inserta un nodo que en su campo dato posee el objeto o después del nodo s de la lista que invoca el método.

Void **conectar** (NodoDoble x, NodoDoble s) *
Conecta el nodo x después del nodo s en la lista que invoca el método.

Void **borrar** (NodoDoble p) *
Borra el nodo p si está en la lista que invoca el método.

Void **desconectar** (NodoDoble p) *
Desconecta el nodo p de la lista que invoca el método.

String **recorreID** () *
Imprime un string con los datos de la lista desde el primero hasta el último.

String **recorreDI** () *¨
Imprime un string con los datos de la lista desde el último hasta el primero.

Void **invertir** ()
Invierte el orden de las conexiones de los nodos que la lista que invoca el método.

Void **duplicar** ()
Retorna una LDLRC igual a la lista que invoca el método.

Clase Termino* //Tomada parcialmente del libro Algoritmia III (Roberto Flórez Rueda).

Privado:

Double **base**
int **exponente**

Público:

Termino () *
Constructor vacío.

Termino (Double base, int exponente) *
Constructor que crea un termino con la base y el exponente entrados por parámetro.

Double **retornaBase** () *
Retorna un Double con la base del término.

Void **asignaBase** (Double base) *
Modifica la base del termino por el valor entrado por parámetro.

Int **retornaExponente** () *
Retorna un int con el exponente del término.

Void **asignaExponente** (int exponente) *
Modifica el exponente del término con el valor entrado por parámetro.

Double **evaluarTermino** (Double x)
Retorna un double con la evaluación del término con el Double x.

Clase Polinomio* //Tomada parcialmente del libro Algoritmia III (Roberto Flórez Rueda).

Público:

Polinomio () *
Constructor vacío.

Polinomio (double base)
Constructor de una constante que será el valor entrado por parámetro.

Polinomio (String s) *nota: se supone que el string es entrado de forma correcta.*
Constructor que inicializa el Polinomio con el String correspondiente al del parámetro s.

Void **insertarTermino** (Termino t)
Inserta un Nodo con el Termino t en el lugar correspondiente de forma descendente.

String **imprimirPolinomio** ()
Retorna un String que representa el polinomio de la lista.

Double **evaluarPolinomio** (Double x)
Retorna un doublé con el resultado de evaluar el polinomio con el valor x entrado por parámetro.

Polinomio **sumarPolinomios** (Polinomio p)
Una lista que representa al polinomio que es la suma del polinomio entrado por parámetro y el que invoca el método.

Polinomio **MultiplicarPolinomios** (Polinomio p)
Una lista que representa al polinomio que es la multiplicación del polinomio entrado por parámetro y el que invoca el método.

Integer **retornaGrado** ()
Retorna un Integer que es el grado del polinomio que invocó el método.

Polinomio **primeraDerivada** ()
Retorna un Polinomio que es la primera derivada del polinomio que invocó el método.

Polinomio **enesimaDerivada** (int n)
Una lista(polinomio) que representa la derivada enésima del polinomio que invocó el método.

Polinomio **integralIndefinida** ()
Una lista(polinomio) que representa la integral definida del polinomio que invocó el método.

Double **integralDefinida** (Double n1, Double n2)
Retorna un Double con el resultado de la Integral Definida en el intervalo [n1, n2].

Boolean **factorPolinomio** (Polinomio factor)
Retorna verdadero si el polinomio del parámetro es un factor del polinomio que invocó el método.

Void **eliminarMonomio** (Polinomio monomio)
Eliminar el monomio entrado por parámetro del polinomio que invocó el método si se encuentra en él.

Clase Main

Público:

Void **main** (String[] args)
Inicializa la gráfica de la clase **Menu.**