Actividad 04

CURSO 2016-2017

<Actividad04>

<FRANCISCO JAVIER MARQUÉS GAONA>

PROCESOS DE LA INGENIERÍA DEL SOFTWARE II

4º GRADO EN INGENIERÍA INFORMÁTICA

Departamento de Informática

Universidad de Almería

ÍNDICE

1. SELECCIÓN DE LA METODOLOGÍA DE TRABAJO

En el desarrollo de nuestro proyecto haremos uso de la metodología Scrum, esta metodología se trata de un proceso ágil que está basada en el uso de los denominados "sprints", en dichos sprints se fijarán las funcionalidades que el sistema debe cumplir siendo los equipos los que se autoorganizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad. La duración de estos "sprints" ronda entre las dos semanas y el mes de duración, en nuestro caso al tratarse de un proyecto no muy complicado con la fijación de un solo sprint podemos concretar todas las tareas que debemos de realizar en dicho espacio de tiempo.

2. SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

En cuanto al lenguaje seleccionado, se ha optado por la realización del proyecto en lenguaje Java, ya que se trata de un lenguaje bien conocido y sencillo de trabajar para los componentes del grupo. La utilización de Java se usa para la realización de las operaciones de sumar, multiplicar y dividir polinomios introducidos por pantalla.
Sin embargo para el cálculo de la estabilidad de un sistema dinámico se usa lenguaje Matlab, el por qué de este hecho es que Matlab al tratarse de un lenguaje más matemático nos proporciona herramientas más sencillas a la hora de calcular la matriz de Routh-Hurwitz, mientras que la programación de la matriz en java sería mucho más compleja y más si tratamos de trabajar con un ArrayList, en Matlab se introduce como parámetro un simple vector con los coeficientes del polinomio.
En cuanto a la estructura de datos utilizados, éstas se describirán más concretamente en el punto siguiente.

3. ESTRUCTURA DE DATOS

La estructura de datos utilizada son dos objetos, un objeto del tipo “Monomio” y otro objeto del tipo “Polinomio”. En cuanto al objeto del tipo “Monomio” tendrá como atributos el coeficiente y el exponente del monomio que se creará por pantalla. Al constructor de dicho objeto solo habrá que introducirle el coeficiente y exponente como parámetros.

```
1  package actividad04.OperacionesPolinomio;
2
3  public class Monomio  {
4
5      private int coeficiente;
6      private int exponente;
7
8      public Monomio(int coeficiente, int exponente) {
9          this.coeficiente = coeficiente;
10         this.exponente = exponente;
11     }
12
13     public int getCoeficiente() {
14         return coeficiente;
15     }
16
17     public int getExponente() {
18         return exponente;
19     }
20
21     public void cambiarSigno() {
22         coeficiente = -coeficiente;
23     }
24     public void setCoeficiente(int value) {
```

El objeto de tipo “Polinomio” contendrá un ArrayList<Monomio> que será el denominado polinomio formado por varios “Monomios”. Se contemplan dos constructores un constructor vacío que simplemente inicializa el polinomio y otro constructor en el que se pasa por parámetro un ArrayList<Monomio> útil para crear un polinomio en el que se pase por ejemplo la suma, multiplicación o división de dos polinomios en forma de Array. En este caso cabe destacar métodos como:

- ***Ordenar():*** ordena el polinomio situando a los monomios de mayor grado en primer lugar.

```
public void ordenar() {
    Collections.sort(polinomio, new Comparator<Monomio>() {
        @Override
        public int compare(Monomio m1, Monomio m2) {
            return new Integer(m2.getExponente()).compareTo(new Integer(m1.g
        }
    });
}
```

- ***Simplificar():*** simplifica el polinomio, es decir, en caso de que por ejemplo se introduzcan dos monomios del mismo grado, éstos se suman automáticamente o bien útil una vez realizadas las operaciones de sumar y multiplicar para expresar el polinomio de una forma más correcta

```
public void simplificar() {
    ArrayList<Integer> exponentes = new ArrayList<Integer>();
    ArrayList<Monomio> monomiosMismoExponente = new ArrayList<Monomio>();
    ArrayList<Monomio> simplificado = new ArrayList<Monomio>();
    for(Monomio m: polinomio) {
        exponentes.add(m.getExponente());
    }
    int max = 0;
    for(Integer i: exponentes) {
        if(i > max) {
            max = i;
        }
    }
    int i = 0;
    boolean encontrado = false;
    while(i <= max) {
        int suma = 0;
        for(Monomio m: polinomio) {
            if(m.getExponente() == i) {
                monomiosMismoExponente.add(m);
                encontrado = true;
            }
        }
        for(Monomio m2: monomiosMismoExponente) {
            suma = suma + m2.getCoeficiente();
        }
        monomiosMismoExponente.clear();
        if(encontrado == true) {
            Monomio mSimplificado = new Monomio(suma, i);
            simplificado.add(mSimplificado);
            encontrado = false;
        }
        i++;
```

- ***Sumar():*** operación que suma dos polinomios

```
public Polinomio sumar(Polinomio p) {
    ArrayList<Monomio> lista = new ArrayList<Monomio>();
    ArrayList<Integer> exponentes = new ArrayList<Integer>();
    for(Monomio m: polinomio) {
        for(Monomio m2: p.polinomio) {
            if(m.getExponente() == m2.getExponente()) {
                int c = m.getCoeficiente() + m2.getCoeficiente();
                Monomio m3 = new Monomio(c, m.getExponente());
                lista.add(m3);
                exponentes.add(m.getExponente());
            }
        }
    }
    for(Monomio m: polinomio) {
        if(!exponentes.contains(m.getExponente())) {
            lista.add(m);
        }
    }
    for(Monomio m: p.polinomio) {
        if(!exponentes.contains(m.getExponente())) {
            lista.add(m);
        }
    }
    Polinomio suma = new Polinomio(lista);
    suma.ordenar();
    return suma;
}
```

- ***Multiplicar():*** multiplicación de polinomios y los simplifica

```
public Polinomio multiplicar(Polinomio p) {
    ArrayList<Monomio> lista = new ArrayList<Monomio>();
    for(Monomio m: polinomio) {
        for(Monomio m2: p.polinomio) {
            int c = m.getCoeficiente()*m2.getCoeficiente();
            int e = m.getExponente() + m2.getExponente();
            Monomio m3 = new Monomio(c, e);
            lista.add(m3);
        }
    }
    Polinomio multiplicacion = new Polinomio(lista);
    multiplicacion.simplificar();
    return multiplicacion;
}
```

- ***Dividir():*** división entre polinomios que muestra el coeficiente y el resto de la división

```
public ArrayList<Polinomio> dividir(Polinomio p) {
    ArrayList<Polinomio> resultado = new ArrayList<Polinomio>();
    ArrayList<Monomio> dividendo = new ArrayList<Monomio>();
    ArrayList<Monomio> cociente = new ArrayList<Monomio>();
    ArrayList<Integer> exponentes = new ArrayList<Integer>();
    if(p.size() > 2 || p.get(0).getCoeficiente() != 1
            || p.get(0).getExponente() != 1 || p.get(1).getExponente() != 0) {
        throw new RuntimeException();
    }
    int grado = polinomio.get(0).getExponente();
    for(int i = grado; i >= 0; i--) {
        exponentes.add(i);
    }
    boolean encontrado = false;
    for(Integer i: exponentes) {
        for(Monomio mon: polinomio) {
            if(i == mon.getExponente()) {
                Monomio nuevo = new Monomio(mon.getCoeficiente(), mon.getExponente());
                dividendo.add(nuevo);
                encontrado = true;
            }
        }
        if(encontrado == false) {
            Monomio nuevo2 = new Monomio(0, i);
            dividendo.add(nuevo2);
        }
        encontrado = false;
    }

    Polinomio r = new Polinomio();
    int divisor = -p.get(1).getCoeficiente();
    Monomio m = new Monomio(polinomio.get(0).getCoeficiente(), polinomio.get(0).getExponente() - 1);
    cociente.add(m);
    int n = dividendo.get(0).getCoeficiente();
    int x = n*divisor;;
    int x2 = dividendo.get(0 + 1).getCoeficiente();
    int suma = x + x2;
    Monomio monomio = new Monomio(suma, dividendo.get(0 + 1).getExponente() - 1);
    cociente.add(monomio);
    for(int i = 1; i < dividendo.size() - 1; i++) {
        suma = suma*divisor + dividendo.get(i + 1).getCoeficiente();
        monomio = new Monomio(suma, dividendo.get(i + 1).getExponente() - 1);
        cociente.add(monomio);
    }
    Polinomio division = new Polinomio(cociente);
    Monomio resto = new Monomio(division.get(division.size() - 1).getCoeficiente(),
            division.get(division.size() - 1).getExponente() + 1);
    r.add(resto);
    division.remove(division.size() - 1);
    resultado.add(division);
    resultado.add(r);
    return resultado;
}
```

También se crear un objeto denominado “User” que será el método main donde el usuario introduce los datos requeridos y selecciona las operaciones (opciones) que desea realizar.
Finalmente respecto al ejercicio del teorema de Bolzano y la falsa posición se crean dos nuevas clases denominadas “FalsaPosicion” y “Funcion”. La clase “Función” será una clase abstracta que tendrá el método eval, que nos permitirá dado un punto (intervalo), calcular el valor de dicho punto (F(x)) en el polinomio. Por otra parte la clase “FalsaPosicion” tendrá un ArrayList que contiene todas las iteraciones que se realizan hasta aproximar el intervalo al error deseado y además tendrá el método calcularRaiz, que es el que procesa el valor de X_R tal y como se explica en el guión siguiendo la ecuación proporcionada.

4. REQUISITOS DEL SISTEMA

4.1 Requisitos funcionales del sistema

En esta sección se especificarán los requisitos funcionales del sistema, también denominados características del sistema u objetivos del sistema.

<id>01	*Introducir datos por pantalla*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguno*
Descripción	*El sistema deberá permitir al usuario introducir datos por pantalla, siendo estos coeficiente y exponente de un monomio.*
Requisitos hijos	• *Crear monomios* • *Crear polinomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*El grado del polinomio (exponente) no debe ser mayor a 100.*

<id>02	*Crear monomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Introducir datos por pantalla*
Descripción	*El sistema deberá crear monomios a partir de los datos introducidos por pantalla.*
Requisitos hijos	• *Operaciones con monomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>03	*Crear polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Introducir datos por pantalla* • *Crear monomios*
Descripción	*El sistema deberá crear un polinomio que estará formado por una será de monomios*
Requisitos hijos	• *Crear monomios* • *Crear polinomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>04	*Sumar monomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema deberá permitir la suma de monomios*
Requisitos hijos	*Ninguno*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>05	*Restar monomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema deberá permitir la resta de monomios*
Requisitos hijos	*Ninguno*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>06	*Multiplicar monomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema deberá permitir la multiplicación de monomios*
Requisitos hijos	• *Sumar exponentes de varios monomios* • *Multiplicar coeficientes varios monomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>07	*Operaciones con polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Sumar monomios* • *Restar monomios* • *Multiplicar monomios*
Descripción	*El sistema deberá permitir la realización de operaciones con polinomios, siendo estas operaciones las de sumar, multiplicar, dividir y simplificar polinomios*
Requisitos hijos	• *Sumar monomios* • *Restar monomios* • *Multiplicar monomios* • *Simplificar monomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>08	*Simplificar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Sumar monomios* • *Restar monomios*
Descripción	*El sistema deberá permitir la simplificación de polinomios, es decir agrupar los monomios de mismo grado en uno solo sumando o restando sus coeficientes.*
Requisitos hijos	• *Ordenar polinomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>09	*Ordenar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Simplificar polinomios*
Descripción	*El sistema deberá permitir la ordenación de un polinomio, para ordenar el polinomio se deberán de poner en primer lugar los monomios con un grado mayor.*
Requisitos hijos	*Ninguno*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>010	*Analizar la estabilidad de un polinomio*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema deberá permitir calcular la estabilidad de un polinomio dado*
Requisitos hijos	• *Comprobar polinomios triviales* • *Cambiar signo de los coeficientes de un polinomio* • *Crear matriz de Routh-Hurwitz*
[Importancia]	*Muy alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>011	*Comprobar polinomio trivial*
[Versión]	*29/03/2017*
[Dependencias]	• *Analizar la estabilidad de un polinomio*
Descripción	*El sistema deberá permitir conocer si un polinomio dado es un caso trivial o no*
Requisitos hijos	• *Comprobar coeficientes nulo* • *Comprobar signo de los coeficientes de todos los monomios de un polinomio dado*
[Importancia]	*Media*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>012	*Comprobar signo de los coeficientes de un polinomio dado*
[Versión]	*29/03/2017*
[Dependencias]	• *Comprobar polinomio trivial*
Descripción	*El sistema deberá permitir recorrer los coeficientes de los monomios de un polinomio dado para determinar sus signos o bien si son nulos.*
Requisitos hijos	*Ninguno*
[Importancia]	*Media*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>013	*Cambiar signo de los coeficientes de un polinomio*
[Versión]	*29/03/2017*
[Dependencias]	• *Analizar la estabilidad de un polinomio*
Descripción	*El sistema deberá permitir recorrer los coeficientes de los monomios de un polinomio dado para cambiar los signos de cada uno de ellos*
Requisitos hijos	*Ninguno*
[Importancia]	*Media*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>014	*Crear matriz de Routh-Hurwitz*
[Versión]	*29/03/2017*
[Dependencias]	• *Analizar la estabilidad de un polinomio*
Descripción	*El sistema deberá permitir crear la matriz de Routh-Hurwitz de un polinomio dado*
Requisitos hijos	• *Imprimir matriz de Routh-Hurwitz por pantalla y mostrar que tipo de estabilidad tiene*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

<id>015	*Imprimir por pantalla matriz de Routh-Hurwitz*
[Versión]	*29/03/2017*
[Dependencias]	• *Crear matriz de Routh-Hurwitz*
Descripción	*El sistema deberá permitir imprimir por pantalla la matriz de Routh-Hurwitz de un polinomio dado y analizar qué tipo de inestabilidad tiene*
Requisitos hijos	*Ninguno*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Finalizado*
Comentarios	*Ninguno*

4.2 Requisitos no funcionales del sistema

A continuación se muestran los requisitos no funcionales que se han identificad, además de clasificando éstos en los siguientes grupos: eficiencia, estabilidad y usabilidad del sistema.

- **4.2.1 Requisitos no funcionales de eficiencia**

<id>01	*Eficiencia en las operaciones con monomios y polinomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema deberá de realizar las operaciones de suma, resta, multiplicación, división, simplificación, etc en un tiempo razonable tanto para los monomios como los polinomios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*En proceso*
Comentarios	*El tiempo que tardan las operaciones no debe ser mayor que unos segundos*

<id>02	*Eficiencia en el cálculo de la estabilidad de un sistema dinámico*
[Versión]	*29/03/2017*
[Dependencias]	• *Eficiencia en las operaciones con monomios y polinomios*
Descripción	*El sistema deberá de realizar la operación de calcular el tipo de estabilidad de un sistema dinámico en un tiempo razonable, además de crear y mostrar la matriz de Routh-Hurwitz*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*En proceso*
Comentarios	*El tiempo que tarda dicha operación no debe ser mayor a unos minutos en los casos en los que el grado de la ecuación sea considerablemente grande*

- ### 4.2.2 Requisitos no funcionales de estabilidad

<id>03	*Estabilidad con ecuaciones de gran tamaño*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*Cuando el grado de las ecuaciones introducidas es bastante elevado el sistema deberá permanecer estable y funcional cuando se realizan operaciones con dichas ecuaciones o bien se calcula la matriz de Routh-Hurwitz evitando que se produzcan errores comunes como StackOverflow*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*En proceso*
Comentarios	*Ninguno*

- **4.2.3 Requisitos no funcionales de usabilidad**

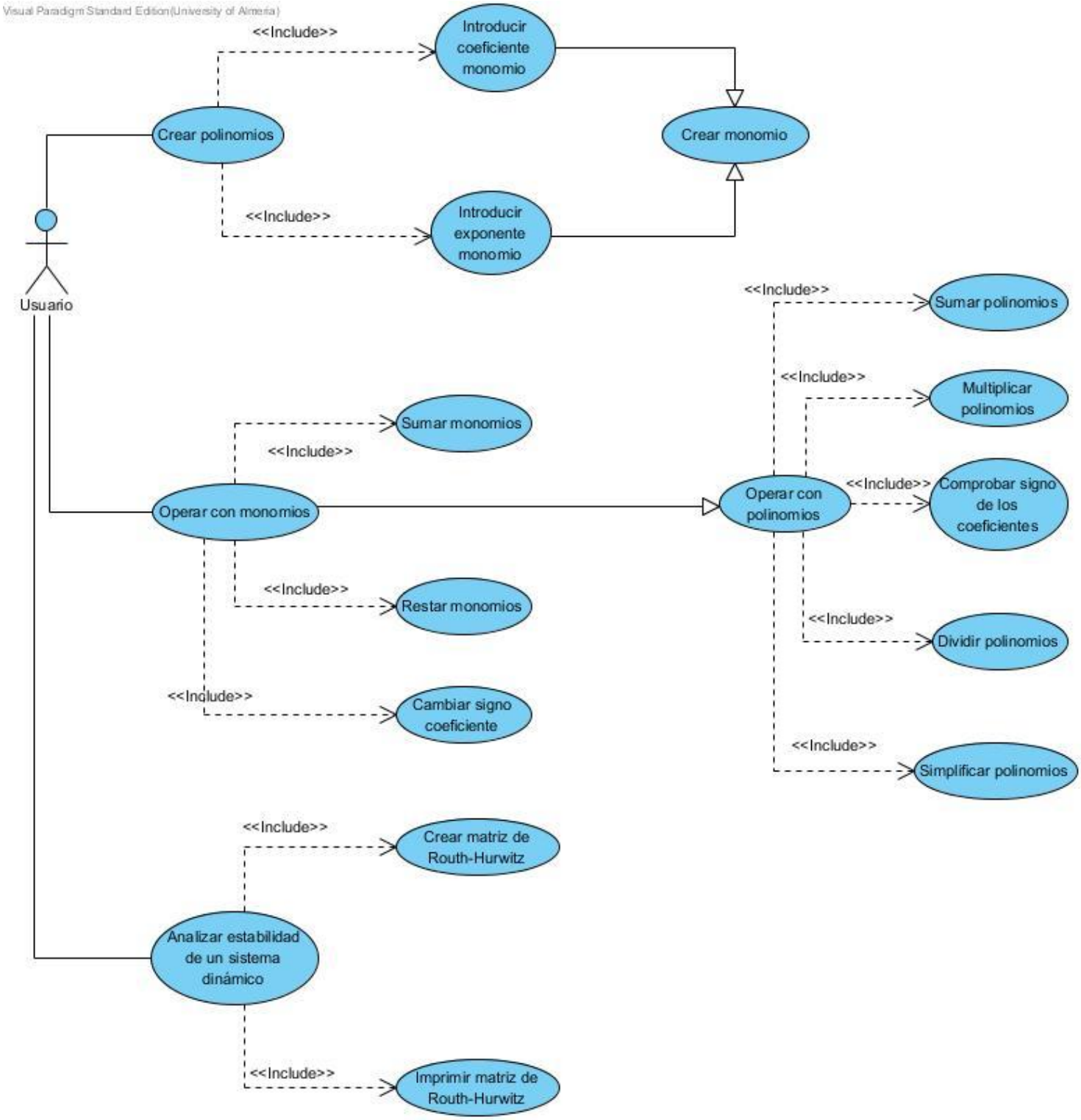
<id>04	*Sencillez en la interfaz*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*La interfaz de usuario debe ser sencilla y fácil de utilizar para todos los usuarios*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>05	*Usabilidad en sistemas*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*El sistema podrá usarse y ejecutarse en sistemas independientemente de su capacidad de proceso, sistema operativo, etc*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

5. ANÁLISIS DE DISEÑO

5.1 Diagramas de Casos de Uso del Sistema

Esta este apartado se muestran los diagramas de casos de uso del sistema que se han identificado.



5.2 Especificación de Actores del Sistema

Esta sección se muestran las especificaciones de los actores que se hayan identificado en los casos de uso, es decir, los diferentes tipos de usuarios y otros sistemas con los que deba interactuar el sistema a desarrollar. En nuestro caso al tratarse de un sistema muy sencillo solo existe un tipo de actor, el actor "Usuario".

<id>01	*Usuario*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Descripción	*Este actor representa al usuario final de la aplicación que será el que introducirá los distintos polinomios para realizar operaciones con ellos o bien calcular la estabilidad de un sistema dinámico*
Comentarios	*<comentarios adicionales sobre el actor del sistema>*

5.3 Especificación de Casos de Uso del Sistema

Esta sección debe contener las especificaciones de los casos de uso del sistema que se hayan identificado, especificados mediante las plantillas para casos de uso propuestas en Madeja. El nivel de detalle de la especificación de cada caso de uso deberá decidirse en función de su importancia y de las necesidades del proyecto. Por este motivo existen dos plantillas, la plantilla simplificada para casos de uso y la plantilla detallada, que se muestran a continuación.

<id>01	*Crear polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Crear monomio*
Precondición	*Ninguna*
Descripción	El sistema deberá crear un objeto del tipo Polinomio introduciendo monomios por pantalla (su coeficiente y exponente), para ello se llaman a otros casos de uso que son: • Introducir coeficiente monomio • Introducir exponente monomio • Crear monomio
Postcondición	*Si el polinomio creado no está ordenado, debe ordenarse*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>02	*Introducir coeficiente monomio*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Ninguna*
Descripción	El usuario podrá introducir por pantalla el coeficiente del monomio que se creará
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>03	*Introducir exponente monomio*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Ninguna*
Descripción	El usuario podrá introducir por pantalla el exponente del monomio que se creará
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>04	*Crear monomio*
[Versión]	*29/03/2017*
[Dependencias]	• *Introducir coeficiente monomio* • *Introducir exponente monomio*
Precondición	*Ninguna*
Descripción	El sistema deberá crear un objeto del tipo “Monomio”
Postcondición	*Una vez creado dicho objeto este se introducirá en el objeto de tipo “Polinomio”*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>05	*Operar con monomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Sumar monomios* • *Restar monomios* • *Cambiar signo coeficiente*
Precondición	*Ninguna*
Descripción	El sistema deberá realizará operaciones básicas con monomios, para ello se llaman a otros casos de uso que son: • Sumar monomios • Restar monomios • Cambiar signo coeficiente
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>06	*Sumar monomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Debe haber dos monomios creados*
Descripción	El sistema deberá sumar distintos monomios
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>07	*Restar monomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Debe haber dos monomios creados*
Descripción	El sistema deberá restar distintos monomios
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>08	*Cambiar signo monomio*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*El coeficiente del monomio no puede ser nulo*
Descripción	El sistema cambiará el signo del coeficiente del monomio
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>09	*Operar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	• *Operar con monomios*
Precondición	*Ninguna*
Descripción	El sistema realizará operaciones básicas con polinomios, para ello se llaman a otros casos de uso que son: • Sumar polinomios • Restar polinomios • Comprobar signo de los coeficientes • Dividir polinomios • Simplificar polinomios
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>010	*Sumar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Deberá haber al menos dos polinomios creados*
Descripción	El sistema sumará los distintos polinomios
Postcondición	*Se creará un nuevo polinomio que será la suma de ambos*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>011	*Multiplicar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Deberá haber al menos dos polinomios creados*
Descripción	El sistema multiplicará los distintos polinomios
Postcondición	*Se creará un nuevo polinomio que será la multiplicación de ambos*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>012	*Dividir polinomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Deberá haber al menos dos polinomios creados y al menos uno de los polinomios debe ser una raíz, es decir debe ser de grado uno*
Descripción	El sistema dividirá los distintos polinomios
Postcondición	*Se creará un nuevo polinomio que será la división de ambos y también se almacenará el resto de la división*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>013	*Simplificar polinomios*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Se debe de haber realizado una operación*
Descripción	La operación de simplificar polinomios se realizará cuando se ha realizado una operación sobre un polinomio para simplificarlo y ordenarlo. Esto implica sumar monomios del mismo grado y ordenar el polinomio de tal forma que se ordene de mayor a menor según los exponentes
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>014	*Cambiar signo de los coeficientes*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Debe haber al menos un polinomio creado*
Descripción	Se cambiarán el signo de todos los coeficientes de un polinomio
Postcondición	*Se creará un nuevo polinomio*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>015	*Analizar estabilidad*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Se comprobará antes de analizar la estabilidad de un polinomio que no se trata de un caso trivial*
Descripción	Se analizará la estabilidad del polinomio según el teorema de Routh-Hurwitz, para ello se llamarán a otros casos de uso que son: • Crear matriz de Routh-Hurwitz
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

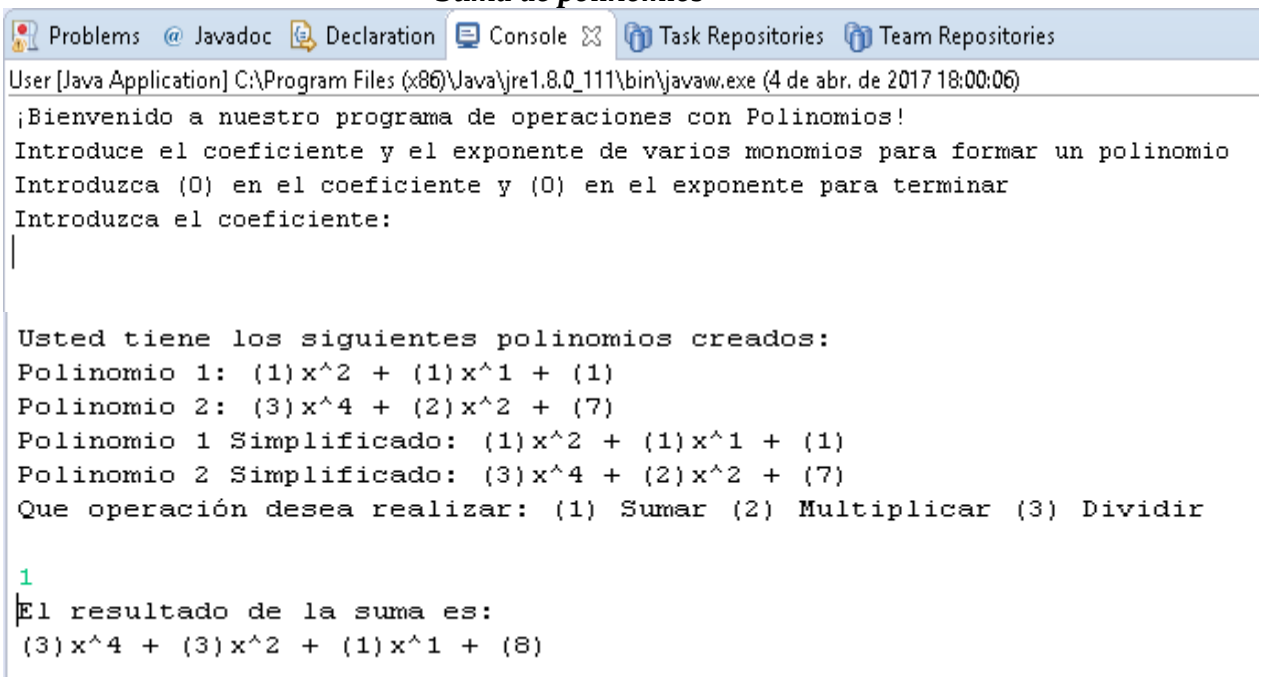
<id>016	*Crear Matriz Routh-Hurwitz*
[Versión]	*29/03/2017*
[Dependencias]	*Ninguna*
Precondición	*Ninguna*
Descripción	Se creará la matriz siguiendo el teorema de Routh-Hurwitz
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

<id>017	*Imprimir Matriz Routh-Hurwitz*
[Versión]	*29/03/2017*
[Dependencias]	• *Crear Matriz Routh-Hurwitz*
Precondición	*Ninguna*
Descripción	Se mostrará por pantalla la matriz de Routh-Hurwitz creada y así mismo se identificará que tipo de estabilidad tiene dicho sistema dinámico
Postcondición	*Ninguna*
[Importancia]	*Alta*
[Prioridad]	*Alta*
[Estado]	*Terminado*
Comentarios	*Ninguno*

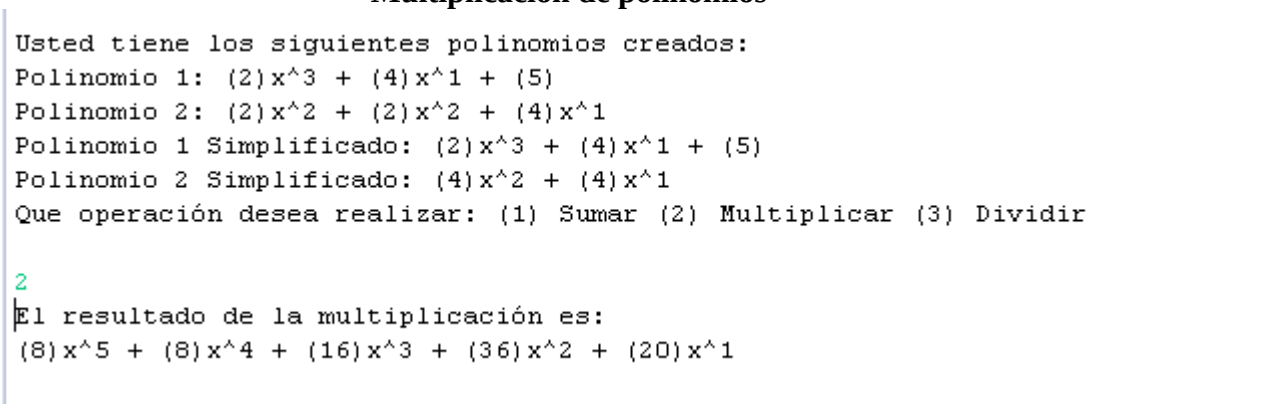
6. FUNCIONAMIENTO DEL PROGRAMA

6.1 Probando funcionalidad operaciones de polinomios

Suma de polinomios



Multiplicación de polinomios



División de polinomios

```
Usted tiene los siguientes polinomios creados:
Polinomio 1: (-7)x^5 + (5)x^3 + (-3)x^2 + (-6)
Polinomio 2: (2)x^2 + (4)x^1
Polinomio 1 Simplificado: (-7)x^5 + (5)x^3 + (-3)x^2 + (-6)
Polinomio 2 Simplificado: (2)x^2 + (4)x^1
Que operación desea realizar: (1) Sumar (2) Multiplicar (3) Dividir

3
Necesita un polinomio divisor de la forma x + t
Introduzca un polinomio raíz (x + t)
Introduzca el coeficiente (debe ser igual a 1):
```

```
Necesita un polinomio divisor de la forma x + t
Introduzca un polinomio raíz (x + t)
Introduzca el coeficiente (debe ser igual a 1):
1
Introduzca el exponente (debe ser igual a 1):
1
Introduzca el coeficiente (t):
4
Introduzca el exponente (debe ser igual a 0):
0
Divisor creado: (1)x^1 + (4)
Selecciona el polinomio que desea dividir
Polinomio 1: (-7)x^5 + (5)x^3 + (-3)x^2 + (-6)
Polinomio 2: (2)x^2 + (4)x^1
```

```
1
Dividendo (en ruffini): (-7)x^5 + (0)x^4 + (5)x^3 + (-3)x^2 + (0)x^1 + (-6)
Divisor: (1)x^1 + (4)
El resultado es:
Cociente: (-7)x^4 + (28)x^3 + (-107)x^2 + (425)x^1 + (-1700)
Resto: (6794)
```

6.2 Probando funcionalidad Estabilidad de un Sistema dinámico

- ***Caso 1: Sistema estable => no hay raíces en semiplano derecho***

```
>> matrizRH([1 10 35 50 124])

 Matriz de Routh-Hurwitz:

matriz =

    1.0000   35.0000  124.0000
   10.0000   50.0000         0
   30.0000  124.0000         0
    8.6667         0         0
  124.0000         0         0


 N° de Raíces  en el semiplano derecho = 0
```

- ***Caso 2: Degeneración en el cálculo***

```
 Matriz de Routh-Hurwitz:

matriz =

     1     2     3
     3     6     3
     0     2     0
  -Inf   NaN     0
   NaN   NaN     0
   NaN   NaN     0


N° de Raíces  en el semiplano derecho = 0
```

- ***Sistema inestable => dos raíces positivas (hay dos cambios de signo en primera columna)***

```
 Matriz de Routh-Hurwitz:

matriz =

    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
    0.7500    1.5000         0
   -3.0000    6.0000         0
    3.0000         0         0
    6.0000         0         0


 N° de Raíces  en el semiplano derecho = 2
```

- ***Caso 4: Sistema críticamente estable => Fila cero => No hay raíces positivas***

```
 Matriz de Routh-Hurwitz:

matriz =

     1    11    18
     2    18     0
     2    18     0
     4     0     0
    18     0     0


 N° de Raíces  en el semiplano derecho = 0
```

6.3 Probando funcionalidad teorema de Bolzano y falsa posición

```
Ha creado el polinomio: (1)x^2 + (-4)x^1 + (1)
Polinomio simplificado: ==>(1)x^2 + (-4)x^1 + (1)

Introduce el los valores del intervalo para calcular lar raíz
Introduzca Xizq:
-2
Introduzca Xder:
0
Intervalo incorrecto, introduzca otro intervalo
Introduzca Xizq:
-1
Introduzca Xder:
0
Intervalo incorrecto, introduzca otro intervalo
Introduzca Xizq:
0
Introduzca Xder:
1
Calculando la raíz en el intervalo: [0,1]
Raíz: 0.26794931956827783
Iteraciones
[Xizq                          Xder                          X                             F(x),
0.0            1.0             0.3333333333333333            -0.2222222222222221,
0.0            0.3333333333333333            0.27272727272727276           -0.016528925619834878,
0.0            0.27272727272727276           0.2682926829268293            -0.001189767995241064 5,
0.0            0.2682926829268293            0.2679738562091503            -8.543722499898543E-5,
0.0            0.2679738562091503            0.2679509632224168            -6.134197846119349E-6,
0.0            0.2679509632224168            0.26794931956827783           -4.4041600832933625E-7]
```