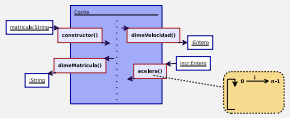
Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen

Programación 1



Práctica 4

UNIVERSIDAD POLITECNICA DE VALENCIA

Escola Politècnica Superior de Gandia

DSIC

Departament de Sistemes Informàtics i Computació

Sesiones

- Grupos martes: dic-4, dic-11, dic-18
- Grupos viernes: dic-7, dic-14, dic-21

Práctica 4

Objetivos

- Diseño de una clase.
- Implementación de una clase dado su diseño.
- Ingeniería inversa del código de una clase.
- Uso de sentencias condicionales y bucles.
- Uso de arrays.
- Utilización de clases ya existentes.

¡ Atención !

▷ Se recuerda que las prácticas deben prepararse antes de acudir al aula informática. Esto incluye leer el código proporcionado y probarlo.

▷ La realización de las prácticas es un trabajo individual y original. En caso de plagio se excluirá al alumno de la asignatura. Por tanto es preferible presentar el trabajo realizado por uno mismo aunque éste tenga errores.

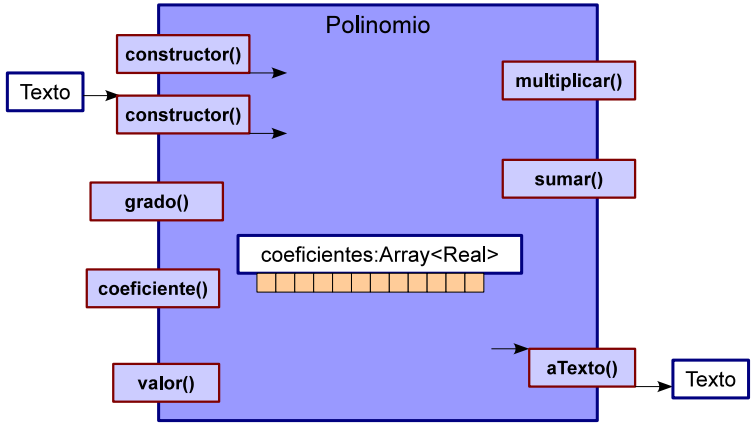
1

Diseño e implementación de la clase Polinomio

Se pretende disponer de una clase Polinomio para poder realizar cálculos y gráficas con polinomios:

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$$

El diseño incompleto de la clase Polinomio es



donde falta por diseñar los métodos:

- grado(): que devuelve el grado del polinomio
- coeficiente(): que, dado i, devuelve a_i (el coeficiente de x^i).
- valor(): que evalúa el polinomio para un valor concreto de la x, devolviendo el resultado de la evaluación.
- multiplicar(): que devuelve un nuevo polinomio, resultado de multiplicar éste por un real.
- sumar(): que suma dos polinomios y devuelve el resultado.

Se dan ya diseñados e implementados los siguientes métodos:

- constructor por defecto. Al utilizarlo:

```
Polinomio p1 = new Polinomio ();
```

  hace que $p_1(x) = 0$
- constructor que recibe un texto. Al utilizarlo:

```
Polinomio p2 = new Polinomio ("1.2 3.4 5.6 7.8");
```

  hace que $p_2(x) = 1.2 + 3.4x + 5.6x^2 + 7.8x^3$
- aTexto() que devuelve un string con los coeficientes del polinomio de menor a mayor exponente separados por un espacio en blanco. Por ejemplo, para el anterior p2, aTexto() devuelve "1.2 3.4 5.6 7.8".

En la parte privada hay un array de números reales llamado coeficientes que, como su nombre indica, guarda los coeficientes del polinomio. En concreto, en la casila i guarda a_i. Es decir, para el anterior p2,

p2.coeficientes contiene

0	1	2	3
[1.2]	[3.4]	[5.6]	[7.8]

1.1 Trabajo a realizar

En papel, hay que completar el diseño de la clase Polinomio() y escribir llamadas de ejemplo a todos sus métodos. Hay que pensar y escribir el algoritmo del método valor().

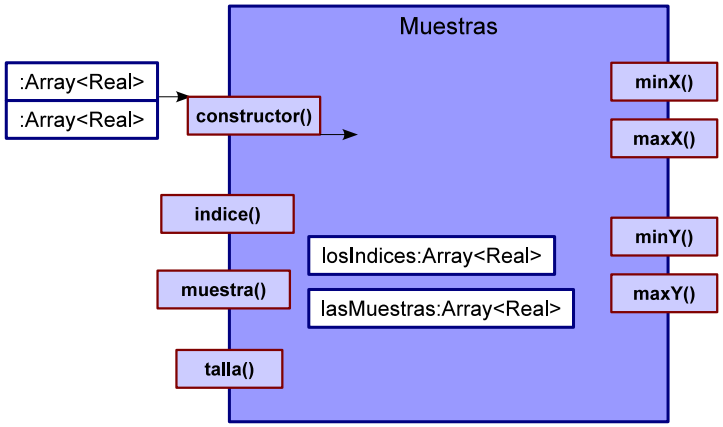
A continuación, en el ordenador, hay que implementar los métodos de la clase Polinomio que se indica más abajo. (Estos métodos han de probarse en E1.java).

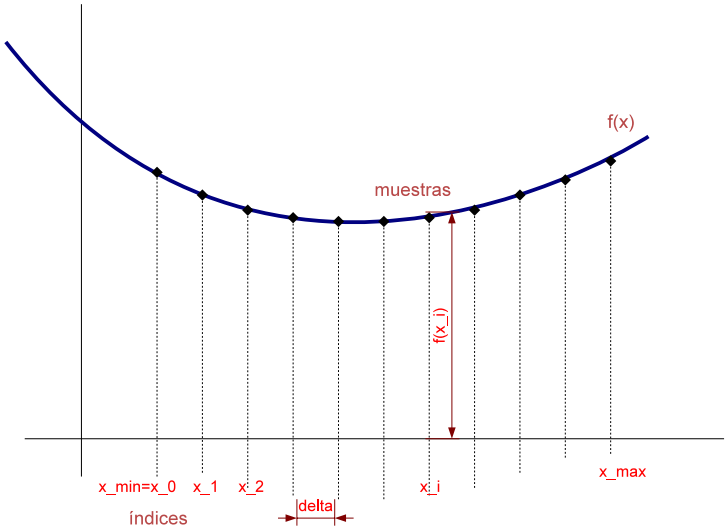
La implementación hay realizarla de la siguiente forma:

1. Implementar grado().
2. Implementar coeficiente().
3. Probar los métodos anteriores.
4. Implementar valor().
5. Probar el método valor().

2 Diseño de la clase Muestras

La clase Muestras se da completamente implementada.





Sirve para guardar muestras tomadas de una función $f(x)$, así como sus correspondientes índices. Por eso contiene dos arrays de reales losIndices y lasMuestras.

Así pues, si el array losIndices contiene

$$x_0, x_1, x_2, x_3, x_4, ..., x_n$$

(se asume que $(\forall i | 0 \leq i < n : x_i < x_{i+1})$) entonces, el array lasMuestras guarda

$$f(x_0), f(x_1), f(x_2), f(x_3), f(x_4), ..., f(x_n)$$

Para crear un objeto Muestras, el constructor recibe justamente dos arrays de reales (del mismo tamaño). El resto de métodos son de acceso y de utilidad sobre las muestras e índices guardados.

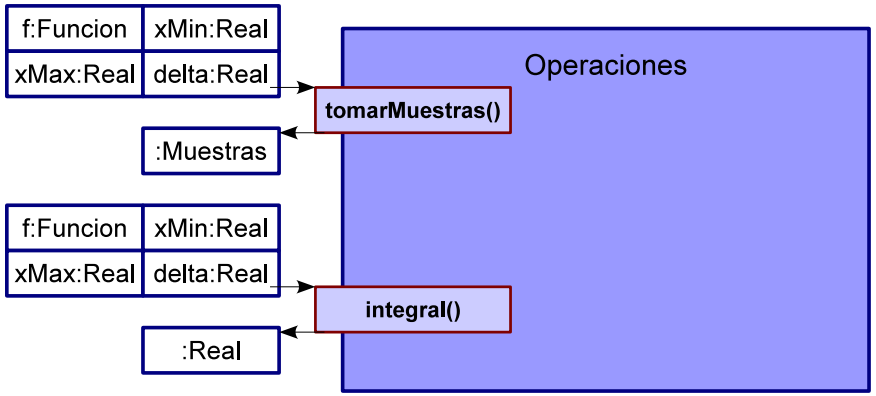
2.1 Trabajo a realizar (en papel)

1. Realiza la ingeniería inversa de la implentación de Muestras. Es decir, haz el dibujo de su diseño a partir del código proporcionado.
2. Escribe llamadas a los métodos de un objeto de la clase Muestras.

3 Implementación de la clase Operaciones

La clase Operaciones contiene dos funciones[1] de utilidad: tomarMuestras() e integral().

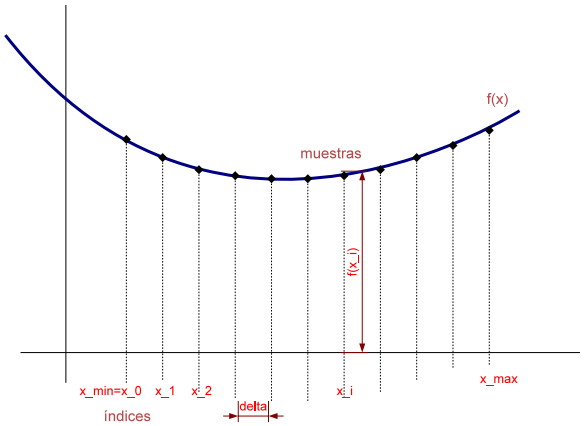
Este es su diseño:



y estos son sus perfiles:

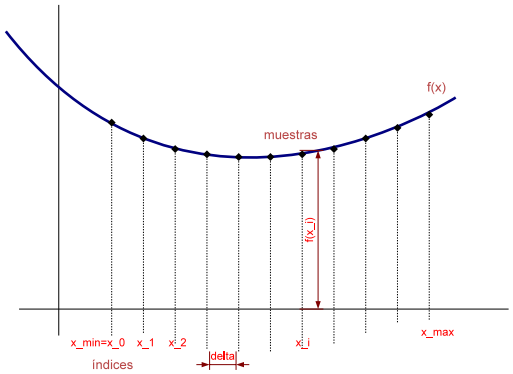
```
public class Operaciones {
 public static Muestras tomarMuestras (Funcion f, double xMin, double xMax, double delta) { } // ()

 public static double integral (Funcion f, double xMin, double xMax, double delta) { } // ()
} // class
```

[1] Métodos estáticos en Java.



- `tomarMuestras()` toma muestras de una función dada `f` en el rango $[x_{min}, x_{max}]$ en intervalos `delta` (Δx). Su algoritmo es el siguiente:

$indices$: array de reales
$valores$: array de reales
x_{min} ——— x (+Δx) ———→ $x_{max} - \Delta x$
  $indices \leftarrow indices \cup \{x\}$
  $valores \leftarrow valores \cup \{f(x)\}$
$muestras \leftarrow (indices, valores)$



- `integral()` calcula el valor de la integral definida[2] de la funcion `f` en el rango $[x_{min}, x_{max}]$

$$\int_{x_{min}}^{x_{max}} f(x)dx$$

utilizando el algoritmo

$$\sum_{x=x_{min}}^{x_{max}} f(x) \cdot \Delta x \quad (x \leftarrow x + \Delta x)$$

o, lo que es lo mismo:

$muestras \leftarrow tomarMuestras(f, x_{min}, x_{max}, \Delta x)$
$sum \leftarrow 0$
para $\forall m \in muestras$ **hacer**
  $sum \leftarrow sum + m$
fin para
$resultado \leftarrow sum \cdot \Delta x$

[2] Que coincide con el área si la función es positiva en todo el intervalo. Si se quiere calcular el área, hay que sumar las muestras en valor absoluto.

3.1 Trabajo a realizar

1. Escribe en papel llamadas a las funciones Operaciones.tomarMuestras() y Operaciones.integral() utilizando objetos Polinomio como función.
2. Implementa dichas funciones en la clase Operaciones.
3. Prueba las anteriores funciones:
   - Comprueba, por ejemplo, que la integral definida de
     - $f(x) = 3$ en el intervalo $[0, 1]$ es 3.
     - $f(x) = x$ en el intervalo $[0, 1]$ es $1/2$.
     - $f(x) = x^2$ en el intervalo $[0, 1]$ es $1/3$.
   - Toma muestras de un polinomio y dibújalas utilizando la clase de utilidad Grafica. Este código:

```
Polinomio p1 = new Polinomio ("0 0 0.1 -1.2 0.1");
Grafica graf = new Grafica ();
Muestras m = Operaciones.tomarMuestras (p1, -10.0, 10.0, 0.1);
graf.dibuja(m);
```

muestra la gráfica de $p(x) = 0.1x^2 - 1.2x^3 + 0.1x^4$ en $[-10, 10]$:

4

Trabajo voluntario valorable

Completar la clase Polinomio:

1. Implementar multiplicar().
2. Implementar sumar().

Diseñar e implementar otros métodos para la clase Polinomio:

1. derivada()
2. integral() (analítica)
3. producto de polinomios

5 Entregas

5.1 Trabajo realizado en papel

- A la entrada de la primera sesión se entregará:
  - El diseño de la clase Polinomio
  - Llamadas de ejemplo a los métodos de Polinomio.
  - El algoritmo del método Polinomio.valor().

  Durante la primera sesión hay que terminar la implementación sólo de los métodos: grado(), coeficiente() y valor() (los demás son voluntarios).
- A la entrada de la segunda sesión se entregará:
  - El diseño de la clase Muestras
  - Llamadas de ejemplo a los métodos de Muestras.
  - Llamadas de ejemplo a los métodos de Operaciones.
  - La preparación de la implementación de los métodos tomarMuestras() e integral().

  Durante la segunda sesión hay que terminar la implementación de los métodos: tomarMuestras() e integral().
- En la tercera sesión se corregiran posibles errores pendientes y se probará a realizar gráficas de polinomios utilizando la cláse Grafica.

5.2 Código completo de la práctica

El fichero p4.zip con el código completo se debe entregar electrónicamente en la propia tarea de poliformaT antes del 24 de diciembre de 2012.

27 noviembre 2012