	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Dra. Rocío Alejandra Aldeco Pérez

Asignatura: Programación orientada a objetos -1323

Grupo: 6

No de Práctica(s): 4

Integrante(s): Mendoza Hernández Carlos Emiliano

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 2023-1

Fecha de entrega: 19 de septiembre del 2022

Observaciones:

CALIFICACIÓN: _____

Práctica 4.

Clases y objetos.

OBJETIVO

- Aplicar los conceptos básicos de la programación orientada a objetos en un lenguaje de programación

ACTIVIDADES

- Crear clases.
- Crear objetos o instancias.
- Invocar métodos.
- Utilizar constructores.

INTRODUCCIÓN

La programación orientada a objetos se basa en el hecho de que se debe dividir el programa, no en tareas, sino en modelos de objetos físicos o simulados. Si se escribe un programa en un lenguaje orientado a objetos, se está creando un modelo de alguna parte del mundo, es decir, se expresa un programa como un conjunto de objetos que colaboran entre ellos para realizar tareas.

Un **objeto** es, por tanto, la representación en un programa de un concepto, y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos

Los objetos pueden agruparse en categorías y una **clase** describe (de un modo abstracto) todos los objetos de un tipo o categoría determinada.

Realiza las siguientes actividades después de leer y revisar en clase la *Práctica de Estudio 4: Clases y objetos.*

En esta práctica deberás implementar la clase *Polynomial* para representar polinomios de una sola variable. Un polinomio está representado por dos campos:

- *degree* que representa el grado del polinomio.
- *coeff* que representa los coeficientes (el elemento de la posición k es el coeficiente de x^k).

Además, deberá contener los siguientes métodos:

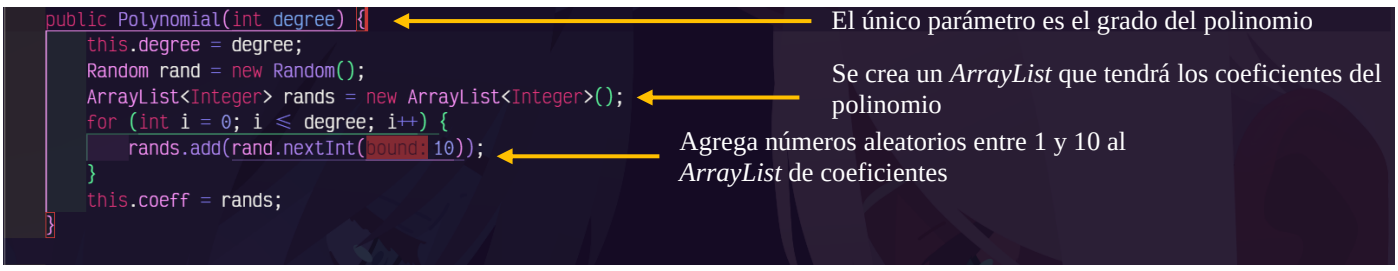
- Tener al menos tres constructores:
  - Un constructor vacío que hace que el polinomio sea cero.
  - Un constructor que dado su grado y un arreglo de coeficientes crea un polinomio.
- El método *evaluate*, que, dado un valor para x, devuelve el resultado de la evaluación de dicho polinomio.
- Los métodos *add* y *substract*, que devuelven el polinomio resultado de la suma y diferencia de los dos polinomios proporcionados como parámetros.
- El método *toString* que imprime un polinomio siguiendo el siguiente formato:

  x^n+x^n-1+…+x^3+x^2+x

  Presentando sólo los coeficientes que tienen algún valor en orden y asumiendo que la variable siempre es x. Por ejemplo: 3x^5-7x^2+x

1. Explica como opera el código de tus constructores no vacíos.

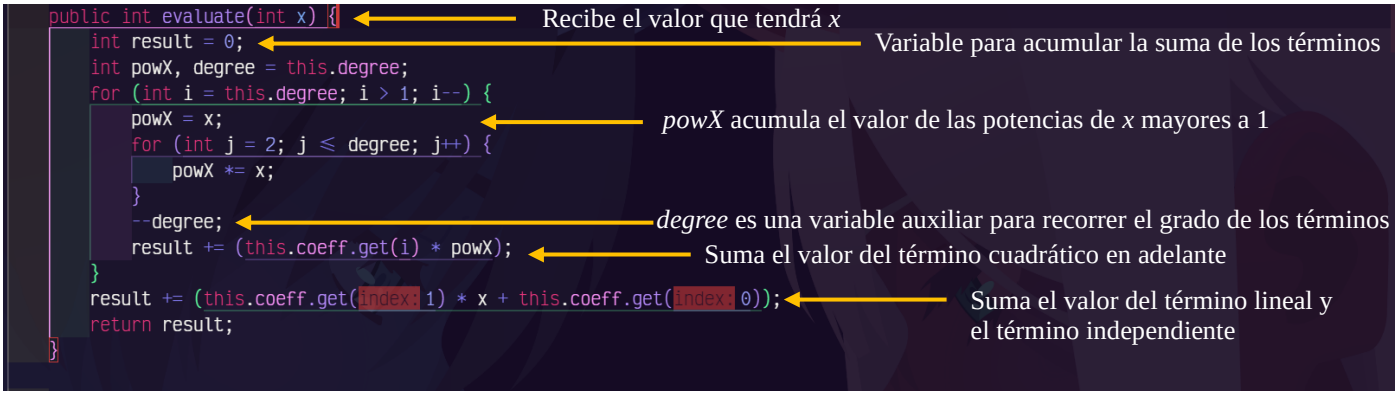
Constructor con un parámetro:



Constructor con dos parámetros:

2. Explica como operan las 3 operaciones que implementaste (evaluación, suma y resta).

Evaluación:



Prueba de escritorio para el polinomio $5x^2 + 6x - 4$ evaluado en $x = 3$.

this.coeff[2]	*this.coeff[1]*	*this.coeff[0]*	*this.degree*	*x*
5	6	- 4	2	3

```
result = 0
degree = 2

i = 2
    powX = 3
    j = 2
        powX = 3*3 = 9
    degree = 1
    result = 0 + this.coeff[2] * powX = 0 + 5 * 9 = 45
i = 1
result = 45 + this.coeff[1] * x + this.coeff[0] = 45 + 6 * 3 + (-4) = 59

return 59
```

Suma:

```java
public Polynomial add(Polynomial anotherPolynomial) {
    ArrayList<Integer> sumCoeff = new ArrayList<Integer>();
    int max = this.degree > anotherPolynomial.degree ? this.degree : anotherPolynomial.degree;
    int sumDegree = max;
    for (int i = 0; i <= max; i++) {
        sumCoeff.add(0);
    }
    if (this.degree > anotherPolynomial.degree) {
        for (int i = 0; i <= anotherPolynomial.degree; i++) {
            sumCoeff.remove(i);
            sumCoeff.add(i, this.coeff.get(i) + anotherPolynomial.coeff.get(i));
        }
        for (int i = anotherPolynomial.degree + 1; i <= this.degree; i++) {
            sumCoeff.remove(i);
            sumCoeff.add(i, this.coeff.get(i));
        }
    } else if (anotherPolynomial.degree > this.degree) {
        for (int i = 0; i <= this.degree; i++) {
            sumCoeff.remove(i);
            sumCoeff.add(i, this.coeff.get(i) + anotherPolynomial.coeff.get(i));
        }
        for (int i = this.degree + 1; i <= anotherPolynomial.degree; i++) {
            sumCoeff.remove(i);
            sumCoeff.add(i, anotherPolynomial.coeff.get(i));
        }
    } else {
        for (int i = 0; i <= sumDegree; i++) {
            sumCoeff.remove(i);
            sumCoeff.add(i, this.coeff.get(i) + anotherPolynomial.coeff.get(i));
        }
    }
    int index = max;
    while (sumCoeff.get(index--) == 0) {
        sumDegree--;
    }
    Polynomial sum = new Polynomial(sumDegree, sumCoeff);
    return sum;
}
```

El método devuelve otro polinomio. Para ello se crean variables que serán los atributos del nuevo polinomio

Se rellena con ceros el *ArrayList* para que, en caso de que un arreglo sea mayor que el otro, complete los coeficientes del menor con ceros.

Se contemplan 3 casos:
1) El polinomio que llama al método es de mayor grado: Agrega a *sumCoeff* la suma de los coeficientes de ambos polinomios desde el término independiente hasta el grado del polinomio menor. El resto de los términos los obtiene en directamente del polinomio con mayor grado.

2) El polinomio que llama al método es de menor grado: Su funcionamiento es análogo al método anterior, pero toma como polinomio de grado mayor al polinomio que fue pasado como parámetro.

3) Ambos polinomios son del mismo grado: Agrega a *sumCoeff* la suma de los coeficientes de los polinomios, término a término.

Valida que el grado que se pasa como parámetro al constructor sea el mayor término existente en el polinomio resultante de hacer la suma

Prueba de escritorio para los polinomios $2x^3 + 5x - 3$ y $2x^3 - 3x^2 + 4x$.

P1.coeff[3]	*P1.coeff[2]*	*P1.coeff[1]*	*P1.coeff[0]*	*P1.degree*
2	0	5	-3	3
P2.coeff[3]	***P2.coeff[3]***	***P2.coeff[3]***	***P2.coeff[3]***	***P2.degree***
2	-3	4	0	3

```
max = 3
sumDegree = 3
```

sumCoeff[3]	*sumCoeff[2]*	*sumCoeff[1]*	*sumCoeff[0]*
0	0	0	0

```
P1.degree > P2.degree ? false
P2.degree > P1.degree ? false
```

```
i = 0
    sumCoeff[0] = P1.coeff[0] + P2.coeff[0] = (-3) + 0 = -3
```

sumCoeff[3]	*sumCoeff[2]*	*sumCoeff[1]*	*sumCoeff[0]*
0	0	0	-3

```
i = 1
    sumCoeff[1] = P1.coeff[1] + P2.coeff[1] = 5 + 4 = 9
```

sumCoeff[3]	*sumCoeff[2]*	*sumCoeff[1]*	*sumCoeff[0]*
0	0	9	-3

```
i = 2
    sumCoeff[2] = P1.coeff[2] + P2.coeff[2] = 0 + (-3) = -3
```

sumCoeff[3]	*sumCoeff[2]*	*sumCoeff[1]*	*sumCoeff[0]*
0	-3	9	-3

```
i = 3
    sumCoeff[3] = P1.coeff[3] + P2.coeff[3] = 2 + 2 = 4
```

sumCoeff[3]	*sumCoeff[2]*	*sumCoeff[1]*	*sumCoeff[0]*
4	-3	9	-3

```
i = 4
index = 3
sumCoeff[3] == 0 ? false

Polynomial sum = new Polynomial(3, sumCoeff)
```

Polinomio obtenido:

coeff[3]	*coeff[2]*	*coeff[1]*	*coeff[0]*	*degree*
4	-3	9	-3	3

Polinomio: $4x^3 - 3x^2 + 9x - 3$

Grado: **3**

Resta:

```java
public Polynomial substract(Polynomial anotherPolynomial) {
    ArrayList<Integer> subCoeff = new ArrayList<Integer>();
    int max = this.degree > anotherPolynomial.degree ? this.degree : anotherPolynomial.degree;
    int subDegree = max;
    for (int i = 0; i <= max; i++) {
        subCoeff.add(0);
    }
    if (this.degree > anotherPolynomial.degree) {
        for (int i = 0; i <= anotherPolynomial.degree; i++) {
            subCoeff.remove(i);
            subCoeff.add(i, this.coeff.get(i) - anotherPolynomial.coeff.get(i));
        }
        for (int i = anotherPolynomial.degree + 1; i <= this.degree; i++) {
            subCoeff.remove(i);
            subCoeff.add(i, this.coeff.get(i) * -1);
        }
    } else if (anotherPolynomial.degree > this.degree) {
        for (int i = 0; i <= this.degree; i++) {
            subCoeff.remove(i);
            subCoeff.add(i, this.coeff.get(i) - anotherPolynomial.coeff.get(i));
        }
        for (int i = this.degree + 1; i <= anotherPolynomial.degree; i++) {
            subCoeff.remove(i);
            subCoeff.add(i, anotherPolynomial.coeff.get(i) * -1);
        }
    } else {
        for (int i = 0; i <= subDegree; i++) {
            subCoeff.remove(i);
            subCoeff.add(i, this.coeff.get(i) - anotherPolynomial.coeff.get(i));
        }
    }
    int index = max;
    while (subCoeff.get(index--) == 0) {
        subDegree--;
    }
    Polynomial sub = new Polynomial(subDegree, subCoeff);
    return sub;
}
```

El funcionamiento de este método es análogo al método de suma, pero con una diferencia. La diferencia es que resta los coeficientes de los términos.

Prueba de escritorio para los polinomios $2x^3 + 5x - 3$ y $2x^3 - 3x^2 + 4x$.

P1.coeff[3]	*P1.coeff[2]*	*P1.coeff[1]*	*P1.coeff[0]*	*P1.degree*
2	0	5	-3	3
P2.coeff[3]	***P2.coeff[3]***	***P2.coeff[3]***	***P2.coeff[3]***	***P2.degree***
2	-3	4	0	3

```
max = 3
subDegree = 3
```

subCoeff[3]	*subCoeff[2]*	*subCoeff[1]*	*subCoeff[0]*
0	0	0	0

```
P1.degree > P2.degree ? false
P2.degree > P1.degree ? false
i = 0
    subCoeff[0] = P1.coeff[0] - P2.coeff[0] = (-3) - 0 = -3
```

subCoeff[3]	*subCoeff[2]*	*subCoeff[1]*	*subCoeff[0]*
0	0	0	-3

```
i = 1
    subCoeff[1] = P1.coeff[1] - P2.coeff[1] = 5 - 4 = 1
```

subCoeff[3]	*subCoeff[2]*	*subCoeff[1]*	*subCoeff[0]*
0	0	1	-3

```
i = 2
    subCoeff[2] = P1.coeff[2] - P2.coeff[2] = 0 - (-3) = 3
```

subCoeff[3]	*subCoeff[2]*	*subCoeff[1]*	*subCoeff[0]*
0	3	1	-3

```
i = 3
    subCoeff[3] = P1.coeff[3] - P2.coeff[3] = 2 - 2 = 0
```

subCoeff[3]	*subCoeff[2]*	*subCoeff[1]*	*subCoeff[0]*
0	3	1	-3

```
i = 4
index = 3
subCoeff[3] == 0 ? true
    subDegree = 2
    index = 2
subCoeff[2] == 0 ? false

Polynomial sub = new Polynomial(2, subCoeff)
```

Polinomio obtenido:

coeff[3]	*coeff[2]*	*coeff[1]*	*coeff[0]*	*degree*
0	3	1	-3	2

Polinomio: $\mathbf{3x^2 + x - 3}$

Grado: **2**

3. Cuando estés seguro de que tu programa es correcto, súbelo a Alphagrader.
4. Finalmente, concluye.

CONCLUSIONES

En esta práctica se desarrolló un programa que permitió el desarrollo y práctica de las siguientes habilidades:

- Abstraer del planteamiento y la resolución del problema.
- Diseñar y codificar una clase (*Polynomial)* con sus respectivos atributos y métodos.
- Codificar e implementar la sobrecarga de constructores.
- Instanciar una clase (*Polynomial)*.
- Invocar los métodos de una instancia.
- Utilizar los constructores.

Con estos puntos, se aplicaron los conceptos básicos de la programación orientada a objetos en Java. Por lo tanto, se cumplieron los objetivos de la práctica.

REFERENCIAS

- *Sierra Katy, Bates Bert*
  SCJP Sun Certified Programmer for Java 6 Study Guide.
  McGrawHill

- *Martín, Antonio*
  Programador Certificado Java 2.
  Segunda Edición.
  México
  Alfaomega Grupo Editor, 2008

- *Joyanes, Luis*
  Fundamentos de programación. Algoritmos, estructuras de datos y objetos.
  Cuarta Edición.
  México
  McGrawHill, 2008

Yo, Carlos Emiliano Mendoza Hernández, hago mención que esta práctica fue de mi autoría.