	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): 5

Integrante(s): Mendoza Hernández Carlos Emiliano

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 2023-1

Fecha de entrega: 23 de septiembre del 2022

Observaciones:

CALIFICACIÓN: _____

Práctica 5.

Abstracción y encapsulamiento.

OBJETIVO

- Aplicar el concepto de abstracción para el diseño de clases que integran una solución, utilizando el encapsulamiento para proteger la información y ocultar la implementación

ACTIVIDADES

- Obtener las características y funcionalidades principales de un objeto dado.
- Utilizar diferentes niveles de acceso a las características y funcionalidades obtenidas.

INTRODUCCIÓN

La **abstracción** es la habilidad de ignorar los detalles de las partes para enfocar la atención en un nivel más alto de un problema.

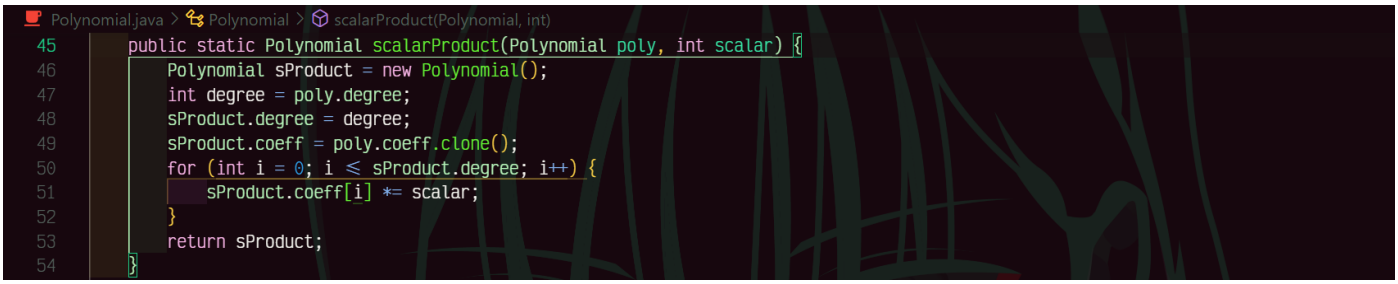
El **encapsulamiento** sucede cuando algo es envuelto en una capa protectora. Cuando el **encapsulamiento** se aplica a los objetos, significa que los datos del objeto están protegidos, "**ocultos**" dentro del objeto. Con los datos ocultos, ¿cómo puede el resto del programa acceder a ellos? (El acceso a los datos de un objeto se refiere a leerlos o modificarlos.) El resto del programa no puede acceder de manera directa a los datos de un objeto; lo tiene que hacer con ayuda de los métodos del objeto.

En el supuesto de que los métodos de un objeto estén bien escritos, los métodos aseguran que se pueda acceder a los datos de manera adecuada. Al hecho de empaquetar o proteger los datos o atributos con los métodos se denomina **encapsulamiento**.

Realiza las siguientes actividades después de leer y revisar en clase la *Práctica de Estudio 5: Abstracción y encapsulamiento.*

1. Dado el código que creaste la práctica anterior, deberás añadir las siguientes funcionalidades:
    a) Multiplicación de un polinomio por un escalar.
    b) Almacenar más de un polinomio en la estructura de datos de tu elección.
2. Describe en que parte de tu programa incluiste estas nuevas funcionalidades. Explica por qué tomaste esta decisión.

El método para multiplicar por un escalar se agregó dentro de la clase *Polynomial*. Elegí ponerla aquí dado que es un método de la clase *Polynomial* al igual que *add* y *substract*.



La funcionalidad para almacenar los polinomios se agregó a la clase *Main*. Me pareció más adecuado colocarla aquí porque creamos los polinomios desde la clase *Main*, mismo lugar desde donde se agregan al *ArrayList*.

3. Describe como es que el concepto de abstracción y encapsulamiento es usado en este diseño.

Abstracción:

Se empieza por el análisis de las clases que conforman el programa, luego se definen los atributos y métodos que contendrán las clases. Para este programa se “extraen” las características más importantes de un polinomio (grado y coeficientes) para conformar los atributos de la clase, así como se definen los métodos de la clase con respecto a las operaciones aritméticas y algebraicas de un polinomio.

Clases:

- *Polynomial*

  Atributos*:*
  - **Int *degree***
  - **Int *coeff*[]**

  Métodos:
  - ***Add***
  - ***Substract***
  - ***Evaluate***
  - ***ScalarProduct***
  - ***toString***
- *Main*

  Métodos:
  - ***Main***

Encapsulamiento:

Los niveles de acceso se definen de la siguiente manera:

- *Polynomial*

  Atributos*:*
  - **Int *degree* → Privado**
  - **Int *coeff*[] → Privado**

Métodos:

- ***Constructores*** → **Público**
- ***Add*** → **Público**
- ***Substract*** → **Público**
- ***Evaluate*** → **Público**
- ***ScalarProduct*** → **Público**
- ***toString*** → **Público**

De esta manera controlamos el acceso a los atributos. Sólo “son visibles” o es posible acceder a ellos mediante sus métodos o constructores porque pertenecen a la misma clase. Sin embargo, no podemos acceder a ellos desde otra clase como la clase *Main*. Así, es posible asegurarse de que solo podemos acceder a los atributos de *Polynomial* con los métodos diseñados para ello.

4. ¿Piensas que existe otra forma de diseñarlo? Explícalo.

Si, porque el resultado de analizar y abstraer un problema no es único. Si bien, en esencia, se resolvería el mismo problema con los mismos métodos y atributos, puede haber diferentes formas de implementar la solución. Por ejemplo, con respecto al programa de la práctica pasada, pensé en cambiar los métodos de operaciones algebraicas por métodos estáticos, lo cual resuelve el problema de la misma forma, pero cambia la comunicación entre los objetos dentro del programa.

CONCLUSIONES

En esta práctica se continuó desarrollando el programa de la práctica, lo que permitió la revisión de los siguientes conceptos:

- Reconocer las características y funcionalidades principales de la clase *Polynomial*. De esta manera, fue posible analizar la abstracción en el diseño de la clase *Polynomial.*
- Utilizar diferentes niveles de acceso, tanto para los atributos como para los métodos de la clase. Así, se utilizó el concepto de encapsulamiento y control de acceso para proteger y ocultar la información.

REFERENCIAS

- *Barnes David, Kölling Michael*
  Programación Orientada a Objetos con Java.
  Tercera Edición.
  Madrid
  Pearson Educación, 2007

- *Deitel Paul, Deitel Harvey*
  Cómo programar en Java.
  Séptima Edición.
  México
  Pearson Educación, 2008

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

- *Dean John, Dean Raymond*
  Introducción a la programación con Java.
  Primera Edición.
  México
  Mc Graw Hill, 2009

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