**TITULO DE LABORATORIO**

**HERENCIA Y POLIMORFISMO**

**HANNA KATHERINE ABRIL GÓNGORA**

**KAROL ASLEY ORJUELA MAPE**

**UNIVERSIDAD MANUELA BELTRÁN**

**PROGRAMACIÓN ORIENTADA A OBJECTOS**

**DOCENTE**

**DIANA MARCELA TOQUICA RODRÍGUEZ**

**BOGOTÁ D.C. VIERNES 8 DEL 2024**

1. **PREGUNTAS ORIENTADORAS**
   1. ¿Cuáles son las características esenciales de la entidad que estás modelando en tu programa?

Sus características principales es que se maneja clases y herencia además de métodos y atributos, también una de las principales características es que hay una interacción con el usuario, se manejan excepciones y validaciones simples, y además de eso se realiza el importe de un módulo externo para propósito del programa.

**Entidad**: Cursos online

**Datos relevantes:**

• Nombre del curso

• Duración en meses

• Precio base

• Descripción del curso

• Nivel del curso (básico, intermedio, avanzado)

• Proyecto final (opcional)

• Plataforma (para C#)

• Lista de estudiantes inscritos

* 1. ¿Qué datos y comportamientos son relevantes para esta entidad y cuáles pueden ser omitidos?

**Comportamientos relevantes:**

• Calcular el precio final del curso con descuentos por duración

• Mostrar información completa del curso

• Inscribir a un estudiante en el curso

**Datos que pueden omitirse:**

• Tal vez el nivel del curso, ya que realmente no cambia mucho ese parámetro en el código.

* 1. ¿Qué datos de la entidad deben ser privados y cuáles pueden ser accesibles desde fuera de la clase?

**Datos privados:** En este caso se definen parámetros como la información principal de cada curso como privada.

**Datos accesibles:** Los parámetros de dar la información se establecen sin ningún encapsulamiento en específico.

* 1. ¿Existe una relación de "es un" entre dos o más clases en tu programa?

Python, Java, C#, JavaScript y HTML\_CSS son subclases de la clase Curso. Ya que estos heredan parámetros importantes de Curso.

* 1. ¿Qué atributos y comportamientos son compartidos por las clases relacionadas?

Las subclases comparten los atributos y comportamientos de la clase padre Curso, como el nombre, la duración, el precio, la descripción y la capacidad de inscribir estudiantes.

* 1. ¿Cómo puedes aprovechar el enlace dinámico para determinar el comportamiento de un objeto en tiempo de ejecución?

El enlace dinámico permite determinar el comportamiento específico de un objeto en tiempo de ejecución. Por ejemplo, si se crea un objeto de la clase Python, se pueden invocar métodos específicos de esa clase como calcular el precio con un descuento adicional para cursos de Python.

* 1. ¿Qué estrategias de diseño puedes utilizar para maximizar la extensibilidad y la adaptabilidad de tu código?

**Estrategias de diseño para la extensibilidad y adaptabilidad:**

• Herencia: La herencia permite crear subclases que heredan y extienden las características de la clase padre.

• Polimorfismo: El polimorfismo permite que diferentes objetos respondan al mismo mensaje de forma diferente, según su tipo específico.

• Encapsulamiento: El encapsulamiento permite ocultar los detalles de implementación de una clase y solo exponer una interfaz pública.

**Ejemplos de aplicación de las estrategias:**

• Se pueden crear nuevas subclases para cursos específicos, como Python para análisis de datos o Java para desarrollo web.

• Se puede implementar un método abstracto en la clase padre Curso para calcular el precio final, permitiendo que cada subclase lo implemente de forma específica.

• Se pueden usar métodos setters y getters para acceder a los datos privados de una clase de forma segura.

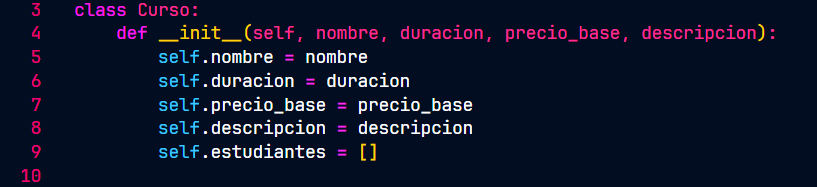
1. **EXPLICACIÓN DEL CÓDIGO EN PYTHON**
   * ‘import sys’: Esto importa el módulo sys, el propósito de haber importado este módulo es porque más adelante se va a implementar como una especie de break, pero en una función, este se utilizara para que el usuario pueda salir del programa.

****

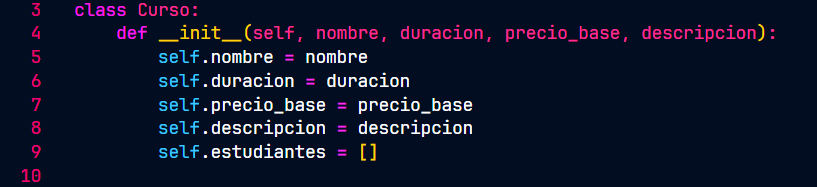
* + class Curso: Define una clase llamada Curso. La cual se va a utilizar como “molde” para las otras clases.

def \_\_init\_\_(self, nombre, duracion, precio\_base, descripcion): Define el método \_\_init\_\_, que es el constructor de la clase. Es llamado automáticamente cuando se crea un nuevo objeto de la clase Curso.

Toma cuatro parámetros: nombre, duracion, precio\_base y descripcion.

****

* + self.nombre = nombre: Asigna el valor del parámetro nombre, el cual es el nombre del curso / nombre de estudiante.
  + self.duracion = duracion: Asigna el valor del parámetro duración, este se refiere a la duración del curso en meses.
  + self.precio\_base = precio\_base: Asigna el valor del parámetro precio\_base al atributo precio\_base del objeto.
  + self.descripcion = descripcion: Asigna el valor del parámetro descripcion al atributo descripcion del objeto.
  + self.estudiantes = []: Inicializa el atributo estudiantes como una lista vacía. Este atributo almacenará los estudiantes inscritos en el curso.

****

* + def inscribir\_estudiante(self, estudiante): Define un método llamado inscribir\_estudiante que toma un parámetro estudiante y lo añade a la lista de estudiantes inscritos en el curso, que se inscribieron con anterioridad.

****

* + def dar\_info(self): Define un método llamado dar\_info que devuelve información detallada sobre el curso, incluyendo su nombre del curso, duración, precio de este y descripción general del curso.

**Texto

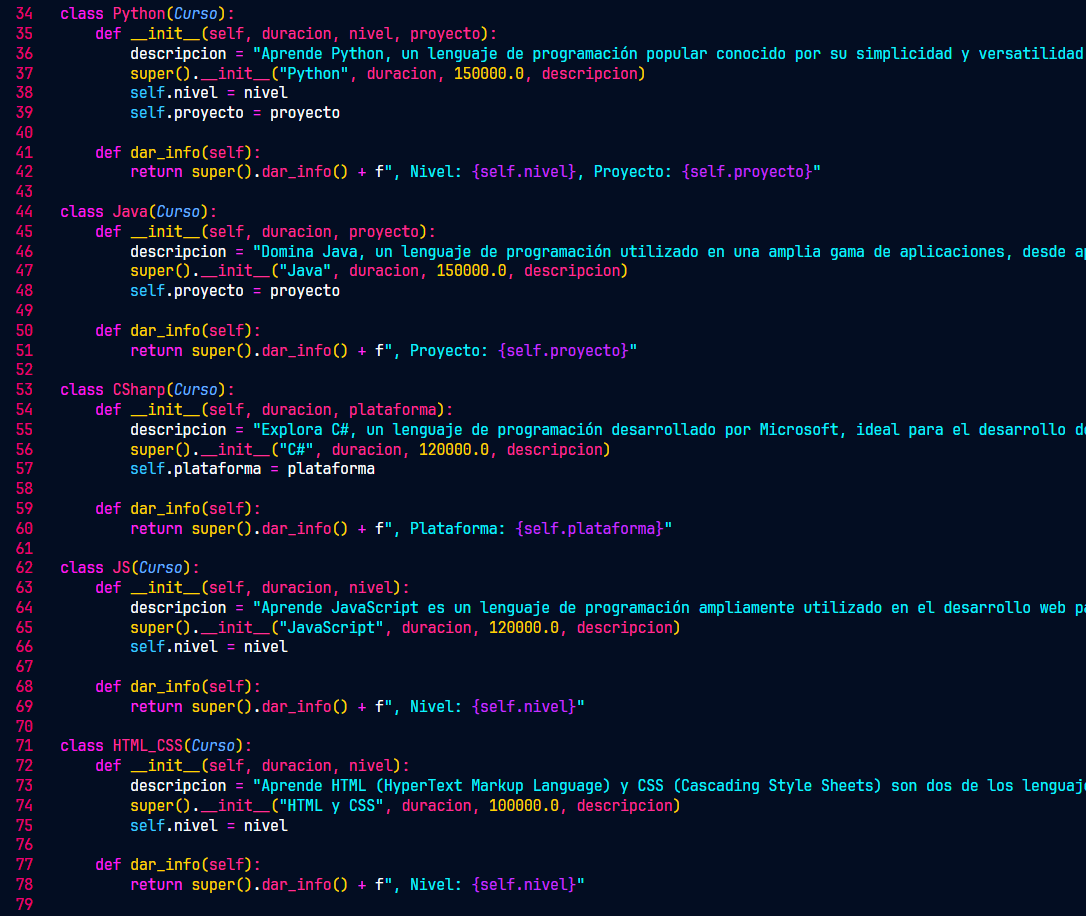
Descripción generada automáticamente**

* + def precio(self): Define un método llamado precio que calcula el precio total del curso, teniendo en cuenta la duración del curso en meses y posibles descuentos basados en la duración de este, únicamente se tienen en cuenta descuentos por 6 o 12 meses.

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

* + Luego se definen varias subclases que heredan de la clase Curso: Python, Java, CSharp, JS, HTML\_CSS y Estudiante. Cada una tiene su propio constructor y métodos específicos, con información propia de este.

****

* + def elegir\_curso(): Define una función llamada elegir\_curso que permite al usuario seleccionar un curso de programación de los listado y proporciona detalles sobre este curso, duración, descripción, precio entre otros.

****

* + def inter\_usu(): Define una función llamada inter\_usu que interactúa con el usuario para seleccionar un curso y proceder con la inscripción del curso que el usuario haya seleccionado.

**Texto

Descripción generada automáticamente**

* + if \_\_name\_\_ == "\_\_main\_\_": Esta línea comprueba si el script se está ejecutando como un programa principal.

inter\_usu(): Llama a la función inter\_usu() para iniciar la interacción con el usuario y comenzar el proceso de inscripción en el curso.

****

1. **EXPLICACION DEL CODIGO UML**

3.1. @startuml cursos\_online: Inicia un diagrama UML con el nombre

“cursos\_online”.

3.2. class Curso {: Define una clase llamada “Curso” con los siguientes atributos y

métodos:

* nombre: str: Un atributo para almacenar el nombre del curso como una cadena de texto.
* duracion: int: Un atributo para almacenar la duración del curso como un entero.
* precio.base: float: Un atributo para almacenar el precio base del curso como un número flotante.
* descripcion: str: Un atributo para almacenar la descripción del curso como una cadena de texto.
* estudiantes [] Estudiante: Una relación de asociación que indica que un curso puede tener múltiples estudiantes asociados.
* \_\_init\_\_(nombre: str, duracion: int, precio.base: float, descripcion: str): Un método constructor para inicializar los atributos del objeto Curso.
* inscribir\_estudiante(estudiante: Estudiante): Un método para inscribir a un estudiante en el curso.
* precio(): float: Un método que devuelve el precio del curso como un número flotante.

3.3. Las clases siguientes (Python, Java, CSharp, js, y HTML\_CSS) son subclases de la

clase “Curso”, lo que significa que heredan todos sus atributos y métodos. Cada una

tiene sus propios atributos adicionales y métodos específicos.

3.4. La clase Estudiante tiene dos atributos:

* nombre: Para guardar el nombre del estudiante.
* email: Para guardar el correo electrónico.

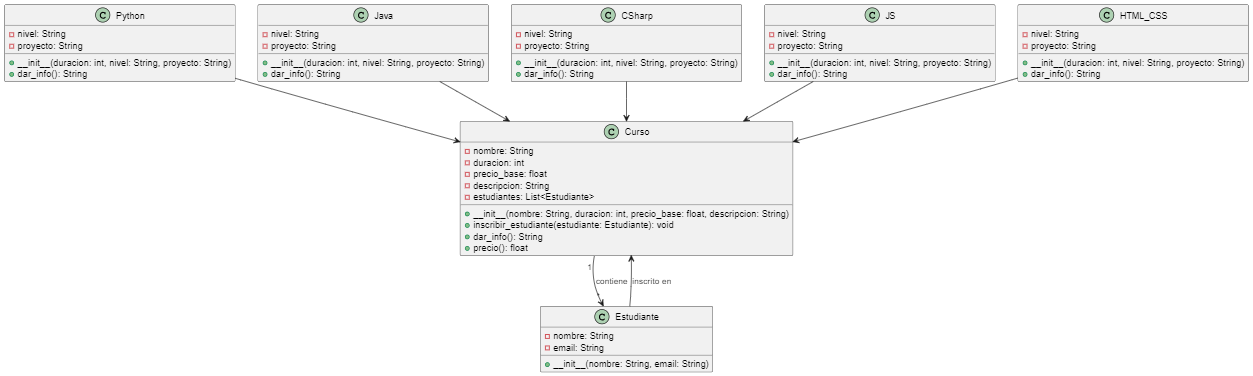
3.5. La última línea muestra una relación entre las clases “Curso” y “Estudiante”,

indicando que un estudiante está contenido en un curso.

**Texto

Descripción generada automáticamente**

1. **DIAGRAMA DEL CODIGO EN UML**

****

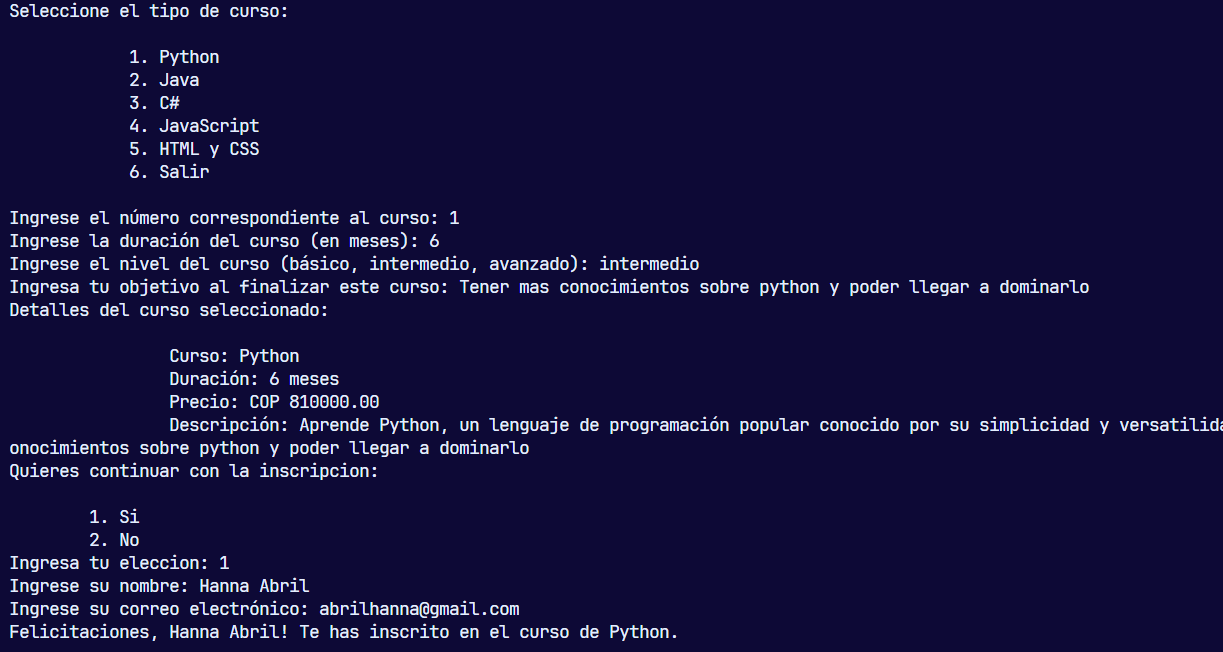
1. **EXPLICACION Y USO**

Este es un pequeño código que simula la página de inicio de una empresa de cursos online de programación, el usuario al ingresar tiene un menú en donde puede escoger el curso que desee, después de que haya realizado la elección se le pedirá cuanto tiempo desea realizar el curso, y después de eso se le dará toda la explicación necesaria sobre el curso, incluyendo una breve descripción de este el nivel del curso y el precio de este, además el precio puede llegar a cambiar dependiendo de la cantidad de meses, y también se realizara descuentos a partir de cierta cantidad de meses.

Es un código relativamente simple, pero en donde podemos implementar todos los pilares de POO el cual es el propósito de este laboratorio.

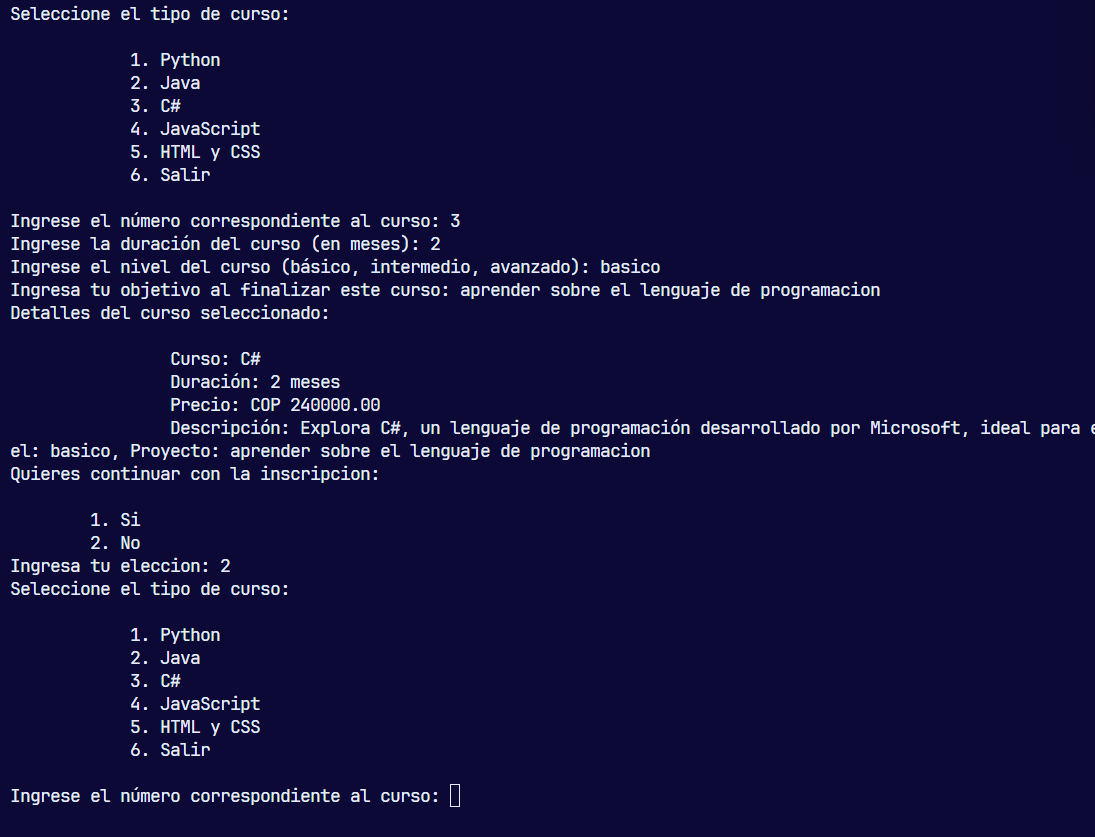
1. **DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO**

6.1. Prueba 1.



Mediante en esta prueba se evidencia el funcionamiento del código, y la información que se le da al usuario al momento de ingresar toda la información, además se muestra el proceso de inscripción al momento en que el usuario acepta continuar con la inscripción.

6.2. Prueba 2.



Mediante en esta prueba se evidencia el funcionamiento del código, y la información que se le da al usuario al momento de ingresar toda la información, además se muestra el proceso de inscripción al momento en que el usuario no desea continuar con la inscripción, por lo que lo devuelve al menú anterior.

1. **DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE**

La razón por la cual Decidimos crear una empresa de clases en línea fue para ayudar a esas personas que quieres a aprender a programar, pero no tienen tiempo para tomar un curso completo donde cada uno puede crear un horario que se adaptan a las necesidades de cada usuario

1. **PILARES DE POO IMPLEMENTADOS**

9.1. **Clases y Objetos:** Se definen varias clases como `*Curso*`, `*Python*`, `*Java*`, `*CSharp*`, `*JS*`, `*HTML\_CSS*`, y `*Estudiante*`. Cada una de estas clases representa una pequeña empresa de clases online y contiene atributos y métodos relacionados con esos conceptos, en este caso se decidió como supuesto que fuera una empresa de cursos online sobre programación, pero este sistema es únicamente como un menú introductorio de que cursos se tienen y que ofrece. Se crean objetos de estas clases en el código principal (`*inter\_usu*`) para modelar y manipular los datos.

9.2. **Encapsulación:** Los datos y el comportamiento relacionado se encapsulan dentro de las clases. Por ejemplo, en la clase `Curso`, los datos como `nombre`, `*duracion*`, `*precio\_base*`, `*descripcion*`, y `estudiantes` se mantienen encapsulados dentro de la clase. El acceso a estos datos se realiza mediante métodos como `*dar\_info*` y `*precio*`.

9.3. **Herencia:** Se utiliza la herencia para crear subclases como `*Python*`, `*Java*`, `*CSharp*`, `*JS*`, y `*HTML\_CSS*` que heredan comportamientos y atributos de la clase base `*Curso*` en donde se definieron anteriormente. Esto permite reutilizar código y definir comportamientos específicos para cada tipo de curso.

9.4. **Polimorfismo:** El polimorfismo se manifiesta en el método `*dar\_info*`. Aunque todas las subclases tienen su propia implementación de este método, cuando se llama a `*dar\_info*` en un objeto de tipo `Curso`, se ejecuta la implementación correspondiente al tipo específico de curso.

9.5. **Métodos y Atributos:** Se definen métodos como *`\_\_init\_\_`,* `*inscribir\_estudiante*`, `*dar\_info*`, y `precio` para encapsular el comportamiento de las clases. Los atributos como `nombre`, `*duracion*`, `*precio\_base*`, `*descripcion*`, `*estudiantes*`, `*nivel*`, y `*proyecto*` almacenan datos relacionados con los objetos de las clases.

1. **CONCLUCIONES**

esta práctica ha fortalecido mi comprensión de la programación orientada a objetos, brindándome la oportunidad de aplicar conceptos teóricos en un proyecto tangible. La combinación de teoría y práctica en este trabajo ha contribuido significativamente a mi desarrollo como estudiante de programación.

1. **ENLACE DE LA INFOGRAFIA – MODELO DE LA PAGINA**

<https://view.genial.ly/65ea5655708b2a0014d79ad6/interactive-content-infografia-robot>