# 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**

1.1. ¿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

1.2. ¿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.

1.3.¿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.

1.4.¿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.

1.5.¿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.

1.6.¿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.

1.7. ¿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.

## 2. 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.

```
1   import sys
2
```

- 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.

```
3   class Curso:
4       def __init__(self, nombre, duracion, precio_base, 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.

```
3   class Curso:
4       def __init__(self, nombre, duracion, precio_base, descripcion):
5           self.nombre = nombre
6           self.duracion = duracion
7           self.precio_base = precio_base
8           self.descripcion = descripcion
9           self.estudiantes = []
10
```

- 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.

```
11      def inscribir_estudiante(self, estudiante):
12          self.estudiantes.append(estudiante)
```

- 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.

```
14      def dar_info(self):
15          return f"""
16                  Curso: {self.nombre}
17                  Duración: {self.duracion} meses
18                  Precio: COP {self.precio():.2f}
19                  Descripción: {self.descripcion}"""
20
```

- 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.

```
21      def precio(self):
22          if self.duracion ⩽ 0:
23              return "Por favor, ingresa un número válido de meses."
24
25          descuento = 0
26          if self.duracion ⩾ 12:
27              descuento = 0.2  # 20% de descuento para 12 meses o más
28          elif self.duracion ⩾ 6:
29              descuento = 0.1  # 10% de descuento para 6 meses o más
30
31          precio_total = self.precio_base * self.duracion * (1 - descuento)
32          return precio_total
33
```

- 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.

```python
class Python(Curso):
    def __init__(self, duracion, nivel, proyecto):
        descripcion = "Aprende Python, un lenguaje de programación popular conocido por su simplicidad y versatilidad
        super().__init__("Python", duracion, 150000.0, descripcion)
        self.nivel = nivel
        self.proyecto = proyecto

    def dar_info(self):
        return super().dar_info() + f", Nivel: {self.nivel}, Proyecto: {self.proyecto}"

class Java(Curso):
    def __init__(self, duracion, proyecto):
        descripcion = "Domina Java, un lenguaje de programación utilizado en una amplia gama de aplicaciones, desde a
        super().__init__("Java", duracion, 150000.0, descripcion)
        self.proyecto = proyecto

    def dar_info(self):
        return super().dar_info() + f", Proyecto: {self.proyecto}"

class CSharp(Curso):
    def __init__(self, duracion, plataforma):
        descripcion = "Explora C#, un lenguaje de programación desarrollado por Microsoft, ideal para el desarrollo d
        super().__init__("C#", duracion, 120000.0, descripcion)
        self.plataforma = plataforma

    def dar_info(self):
        return super().dar_info() + f", Plataforma: {self.plataforma}"

class JS(Curso):
    def __init__(self, duracion, nivel):
        descripcion = "Aprende JavaScript es un lenguaje de programación ampliamente utilizado en el desarrollo web p
        super().__init__("JavaScript", duracion, 120000.0, descripcion)
        self.nivel = nivel

    def dar_info(self):
        return super().dar_info() + f", Nivel: {self.nivel}"

class HTML_CSS(Curso):
    def __init__(self, duracion, nivel):
        descripcion = "Aprende HTML (HyperText Markup Language) y CSS (Cascading Style Sheets) son dos de los lenguaj
        super().__init__("HTML y CSS", duracion, 100000.0, descripcion)
        self.nivel = nivel

    def dar_info(self):
        return super().dar_info() + f", Nivel: {self.nivel}"

```

- 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.

```python
def elegir_curso():

    curso_valido = False

    while not curso_valido:
        print("Seleccione el tipo de curso:")
        print("""
            1. Python
            2. Java
            3. C#
            4. JavaScript
            5. HTML y CSS
            6. Salir
        """)
        opcion = input("Ingrese el número correspondiente al curso: ")
        duracion = int(input("Ingrese la duración del curso (en meses): "))

        if opcion == "1":
            nivel = input("Ingrese el nivel del curso (básico, intermedio, avanzado): ")
            proyecto= input("Ingresa tu objetivo al finalizar este curso: ")
            curso = Python(duracion, nivel, proyecto)
        elif opcion == "2":
            nivel = input("Ingrese el nivel del curso (básico, intermedio, avanzado): ")
            proyecto= input("Ingresa tu objetivo al finalizar este curso: ")
            curso = Java(duracion, nivel, proyecto)
        elif opcion == "3":
            nivel = input("Ingrese el nivel del curso (básico, intermedio, avanzado): ")
            proyecto= input("Ingresa tu objetivo al finalizar este curso: ")
            curso = CSharp(duracion, nivel, proyecto)
        elif opcion == "4":
            nivel = input("Ingrese el nivel del curso (básico, intermedio, avanzado): ")
            proyecto= input("Ingresa tu objetivo al finalizar este curso: ")
            curso = JS(duracion, nivel, proyecto)
        elif opcion == "5":
            nivel = input("Ingrese el nivel del curso (básico, intermedio, avanzado): ")
            proyecto= input("Ingresa tu objetivo al finalizar este curso: ")
            curso = HTML_CSS(duracion, nivel, proyecto)
        elif opcion== "6":
            print("Saliendo del sistema...")
            sys.exit()
        else:
            print("Opción no válida. Por favor, seleccione una opción válida.")
            return elegir_curso()

        return curso
```

- 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.

```
132  def inter_usu():
133      curso_elegido = elegir_curso()
134      print("Detalles del curso seleccionado:")
135      print(curso_elegido.dar_info())
136      print("Quieres continuar con la inscripcion: ")
137      print("""
138          1. Si
139          2. No""")
140
141      choice = input("Ingresa tu eleccion: ")
142
143      if choice == "1":
144          nombre_estudiante = input("Ingrese su nombre: ")
145          email_estudiante = input("Ingrese su correo electrónico: ")
146
147          estudiante = Estudiante(nombre_estudiante, email_estudiante)
148          curso_elegido.inscribir_estudiante(estudiante)
149
150          print(f"Felicitaciones, {nombre_estudiante}! Te has inscrito en el curso de {curso_elegido.nombre}.")
151
152      elif choice == "2":
153          return elegir_curso()
154
155      else:
156          print("Opción no válida. Por favor, seleccione una opción válida.")
157          return inter_usu()
158
```

- 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.

```
159  if __name__ == "__main__":
160      inter_usu()
161
```

3. **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.

```
@startuml cursos_online

class Curso {
    -nombre: str
    -duracion: int
    -precio_base: float
    -descripcion: str
    -estudiantes: List[Estudiante]
    +__init__(nombre: str, duracion: int, precio_base: float, descripcion: str)
    +inscribir_estudiante(estudiante: Estudiante)
    +dar_info(): str
    +precio(): float
}

class Python {
    -nivel: str
    -proyecto: str
    +__init__(duracion: int, nivel: str, proyecto: str)
    +dar_info(): str
}

class Java {
    -proyecto: str
    +__init__(duracion: int, proyecto: str)
    +dar_info(): str
}

class CSharp {
    -plataforma: str
    +__init__(duracion: int, plataforma: str)
    +dar_info(): str
}

class JS {
    -nivel: str
    +__init__(duracion: int, nivel: str)
    +dar_info(): str
}

class HTML_CSS {
    -nivel: str
    +__init__(duracion: int, nivel: str)
    +dar_info(): str
}

class Estudiante {
    -nombre: str
    -email: str
    +__init__(nombre: str, email: str)
}

Curso "1" *-- "*" Estudiante : contiene

@enduml

```

## 4. DIAGRAMA DEL CODIGO EN UML

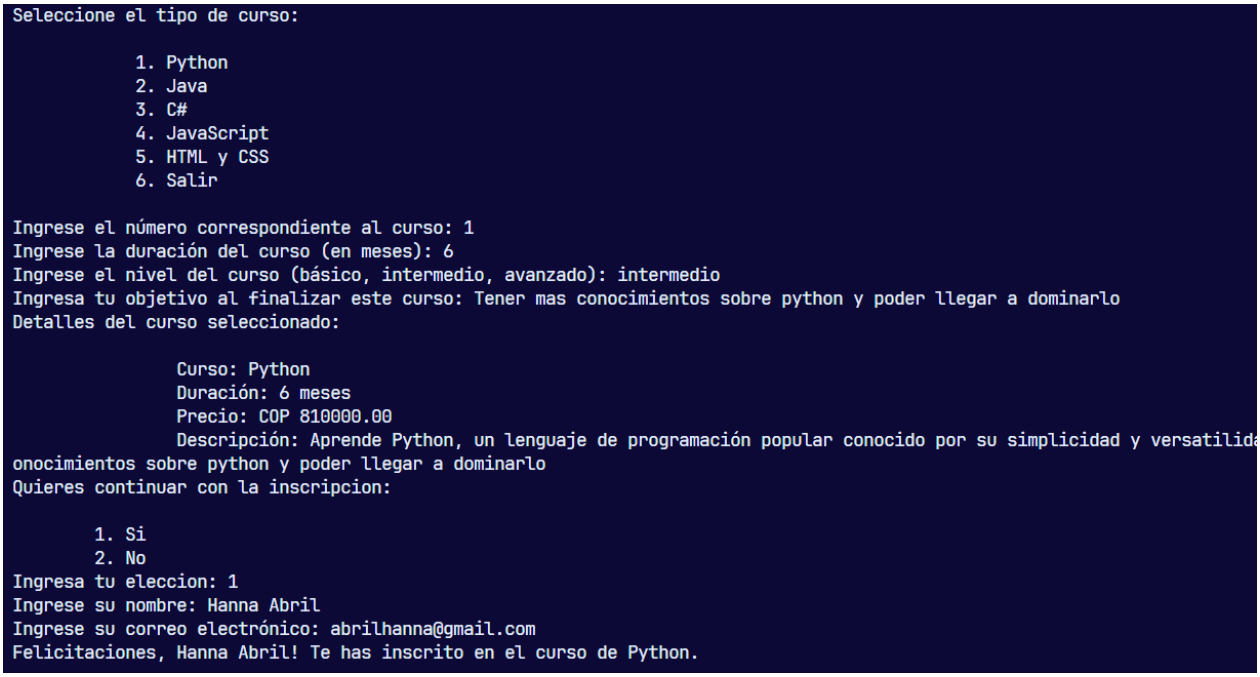

## 5. EXPLICACION Y USO

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.

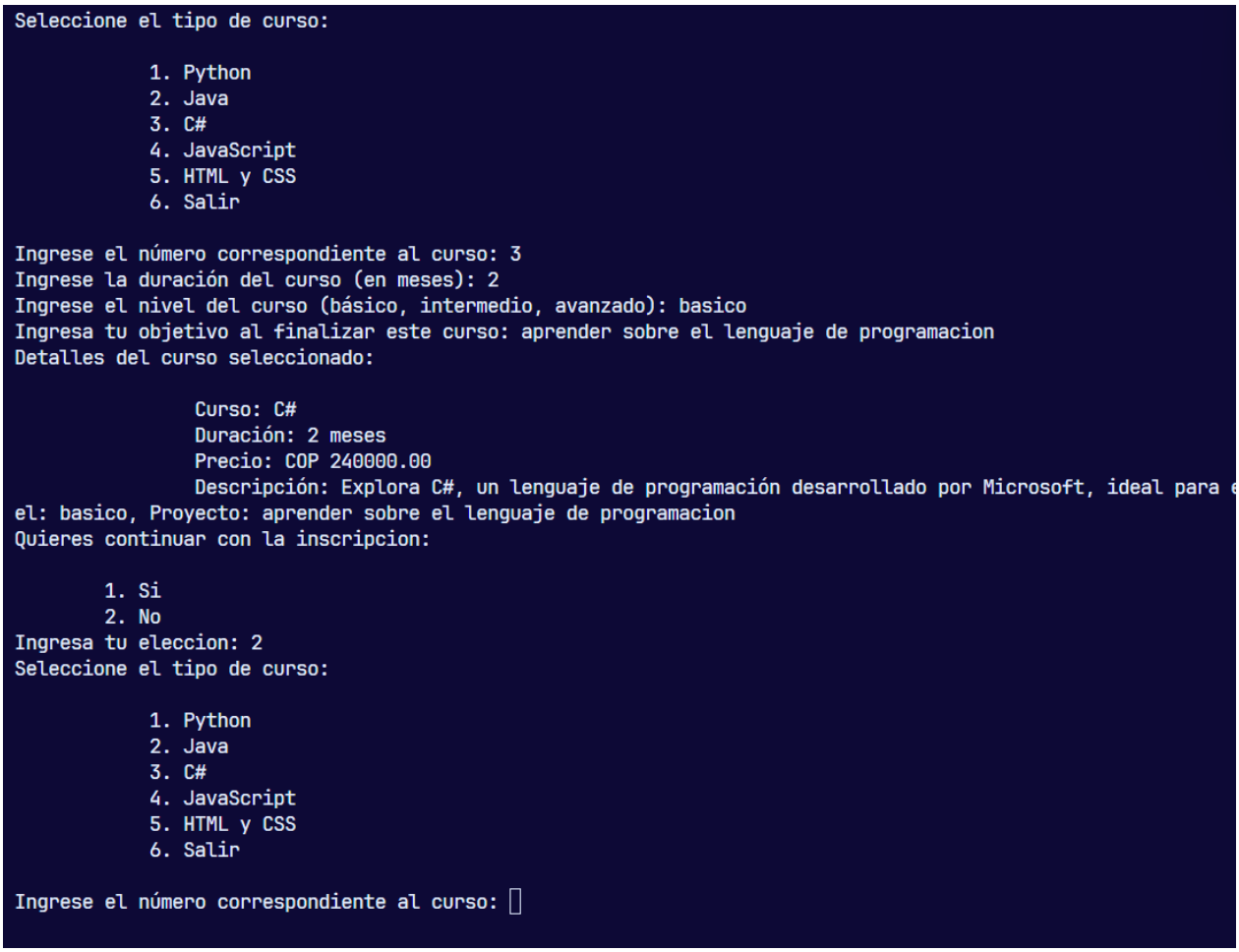
## 6. DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO

### 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.

## 7. 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

## 8. 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.

## 9. 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.

10. **ENLACE DE LA INFOGRAFIA – MODELO DE LA PAGINA**
    https://view.genial.ly/65ea5655708b2a0014d79ad6/interactive-content-infografia-robot