 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 1 de 14




UNIVERSIDAD
NACIONAL
DE COLOMBIA

Parcial 3
Programación orientada a objetos (POO)

Universidad Nacional de Colombia

Estudiante:
Brayan Andrés Guerrero Cortés
CC. 1011201494

07 de marzo de 2025
Programación de computadores
Grupo 18
Profesor: Gustavo Adolfo Mojica Perdigón

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 2 de 14

CONTROL DEL DOCUMENTO				
NOMBRES Y APELLIDOS		CC	FIRMA	FECHA
ELABORA	Brayan Guerrero Cortés	1011201494		07-03-2025

TABLA DE CONTENIDO

1. INTRODUCCIÓN

- 1.1 Presentación del paradigma de programación orientada a objetos (POO)
- 1.2 Importancia y relevancia actual en el desarrollo de software

2. PARTE TEORICA

- 2.1 Definición y fundamentos
- 2.2 Historia y evolución
- 2.3 Casos de uso
- 2.4 Lenguajes De programación asociados

3. PARTE PRÁCTICA


- 3.1 Implementación ejemplo práctico
- 3.2 Desafíos y consideraciones
- 3.3 Comparación con otros paradigmas

4. CONCLUSIONES

- 4.1 Resumen puntos clave
- 4.2 Reflexión
- 4.3 Opinión aplicabilidad en la ingeniería agrícola

5. REPOSITORIO

Brayux/Paradigma-Programaci-n-Orientada-a-objetos

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 3 de 14

1. INTRODUCCIÓN


1.1 Presentación del paradigma de programación orientada a objetos (POO)

En el desarrollo de software, la forma en que se organiza y estructura un código puede marcar la diferencia entre un proyecto exitoso y uno que se convierte en un dolor de cabeza. Aquí es donde entra en juego la Programación Orientada a Objetos (POO), un paradigma que revolucionó la forma en que pensamos y se construye software. En lugar de tratar el código como una serie de instrucciones secuenciales, la Programación orientada a objetos nos lleva a pensar y razonar en términos de objetos, entidades que representan elementos del mundo real o conceptos abstractos. Estos objetos no solo contienen datos, sino también las acciones que pueden realizar, lo que los convierte en bloques de construcción poderosos y variables.

A diferencia de la programación estructurada, que se enfoca en funciones y procedimientos independientes, la Programación orientada a objetos agrupa datos y comportamientos en un solo lugar, creando un sistema más intuitivo y fácil de manejar. Esto no solo facilita la reutilización del código, sino que también permite gestionar sistemas complejos de manera más eficiente. Por eso, la Programación orientada a objetos se ha convertido en una de las técnicas más populares en el desarrollo de software moderno, utilizada en todo tipo de aplicaciones, desde videojuegos hasta sistemas bancarios en los que confiamos nuestro dinero.

1.2 Importancia y relevancia actual en el desarrollo de software

La Programación orientada a objetos es una herramienta que se utiliza en una amplia variedad de aplicaciones en la industria del software. A continuación se presentan algunos ejemplos de cómo la POO está presente en nuestro día a día: Una de ellas son las aplicaciones empresariales, Frameworks como Java EE y .NET utilizan la POO para crear sistemas de gestión empresarial, plataformas de comercio electrónico y aplicaciones de facturación. Así como en el sector de los videojuegos, Motores gráficos como Unity y Unreal Engine se basan en la POO para representar personajes, escenarios y físicas del juego mediante objetos que interactúan entre sí. Además de la Inteligencia artificial, en lenguajes como Python, la POO es fundamental para implementar modelos de aprendizaje automático y “redes neuronales”. No obstante también se implementa en

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 4 de 14

el desarrollo web, Lenguajes como JavaScript, PHP y Ruby utilizan la POO para construir aplicaciones dinámicas y modulares que interactúan con bases de datos y manejan sesiones de usuario y para finalizar, un sector altamente demandado en la actualidad, las aplicaciones móviles, tanto Android como iOS emplean la Poo en sus entornos de desarrollo, como Swift y Kotlin, para crear aplicaciones escalables y fáciles de mantener.

En la Programación Orientada a Objetos, todo gira en torno a los objetos. Un objeto es como una caja que contiene dos cosas: datos (llamados atributos) y comportamientos (llamados métodos). Estos objetos se crean a partir de clases, que son como moldes o plantillas que definen cómo deben ser y qué pueden hacer. Por ejemplo, si queremos representar un cultivo en un programa, podríamos crear una clase llamada "Cultivo" con atributos como *tipo*, *variedad* y *etapaDeCrecimiento*, y métodos como *regar()*, *fertilizar()* y *cosechar()*. A partir de esta clase, podemos crear tantos objetos "cultivo" como queramos, cada uno con sus propias características, como un cultivo de maíz, uno de trigo o uno de tomate.


Imagina que estás diseñando un sistema para gestionar una granja. Podrías tener una clase "Cultivo" que te permita crear objetos para cada tipo de planta que siembras, cada uno con sus propios datos (como la cantidad de agua que necesita o el tipo de suelo preferido) y comportamientos (como cuándo regarlo o cuándo está listo para la cosecha). Esto no solo te ayuda a organizar la información de manera clara, sino que también te permite reutilizar el código para diferentes cultivos, ahorrando tiempo y esfuerzo.

Pero la POO no se trata solo de objetos y clases. Su centro de funcionamiento radica en cuatro principios fundamentales que la hacen que sea original: Abstracción, encapsulamiento, herencia y polimorfismo.

2. PARTE TEORICA

2.1 Definición, fundamentos, ventajas y desventajas

Por ejemplo, realicemos la suposición que estoy desarrollando un sistema para gestionar una granja. En lugar de tratar todo como una serie de funciones y variables sueltas, la POO permite crear objetos como "Cultivo", "Tractor" o "Sistema_De_Riego". Cada uno de estos objetos tendría

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 5 de 14

sus propios datos (como el tipo de cultivo, la marca del tractor o el caudal de agua del riego) y comportamientos (como regar, arar o encender).


La POO se sostiene sobre cuatro principios fundamentales que la hacen única y poderosa:

1. Abstracción. La abstracción es la manera de simplificar la complejidad. Consiste en modelar solo los aspectos más relevantes de un sistema, ignorando los detalles innecesarios. Por ejemplo, si estás creando una clase "Cultivo", no necesitas incluir detalles como la composición química exacta del suelo o la genética de la planta. En su lugar, te enfocas en atributos clave como *tipo*, *etapa_De_Crecimiento* y *necesidades_Agua*, y en métodos como *regar()* o *cosechar()*. La abstracción permite que los programadores se concentren en lo importante, sin perderse en detalles que no aportan valor al problema que están resolviendo.

2. Encapsulamiento. El encapsulamiento es como poner un escudo protector alrededor de los datos de un objeto. Consiste en ocultar los detalles internos de un objeto y exponer solo una interfaz controlada para interactuar con él. Esto se logra mediante modificadores de acceso como *privado*, *protegido* y *público*. Por ejemplo, en una clase "Sistema_De_Riego", el atributo *caudal_Agua* podría ser *privado*, y solo se podría modificar a través de un método público como *ajustar_Caudal()*. Esto no solo protege los datos de modificaciones no autorizadas, sino que también hace que el código sea más seguro y fácil de mantener.

3. Herencia. La herencia es un mecanismo que permite crear nuevas clases basadas en clases existentes, reutilizando código y estableciendo relaciones jerárquicas. Por ejemplo, se podría tener una clase "Maquinaria_Agrícola" con atributos como *marca* y *modelo*, y métodos como *encender()* y *apagar()*. A partir de esta clase, se podría crear subclases como "Tractor" o "Cosechadora", que heredan los atributos y métodos de "Maquinaria_Agrícola" pero también pueden añadir características específicas, como *arar()* o *cosechar()*. La herencia además de ahorrar tiempo al evitar la duplicación de código, también facilita la creación de sistemas modulares y escalables.

4. Polimorfismo. El polimorfismo es la capacidad de que objetos de diferentes clases respondan de manera distinta a un mismo mensaje o método. Esto permite diseñar sistemas más flexibles y


 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 6 de 14

adaptables. Por ejemplo, se podría tener un método `realizar_mantenimiento()` que funcione de manera diferente para un objeto "Tractor" (revisando el motor) y un objeto "Sistema_riego" (limpiando los filtros). El polimorfismo es especialmente útil en sistemas complejos, donde un mismo código puede comportarse de distintas maneras según el contexto.

La Programación Orientada a Objetos ofrece numerosas ventajas, como una mayor organización al agrupar datos y comportamientos en objetos, lo que hace que el código sea más estructurado y fácil de entender; la reutilización de código gracias a la herencia y la composición, permitiendo usar el mismo código en diferentes partes de un proyecto o en proyectos futuros; la facilidad de mantenimiento, ya que el encapsulamiento y la modularidad simplifican la corrección de errores y la adición de nuevas funcionalidades sin afectar otras partes del sistema; y el modelado del mundo real, que permite representar problemas de manera intuitiva, facilitando la comunicación entre desarrolladores y otros profesionales, como ingenieros agrícolas. Sin embargo, la POO también tiene desventajas, como una curva de aprendizaje más pronunciada, ya que conceptos como la herencia y el polimorfismo pueden ser difíciles de entender al principio, especialmente para programadores principiantes; la posible sobrecarga de abstracción, donde en proyectos pequeños o simples la POO puede añadir complejidad innecesaria; y cuestiones de rendimiento, ya que en algunos casos la POO puede ser menos eficiente en términos de memoria y velocidad comparada con otros paradigmas, como la programación procedural.

2.2 Historia y evolución

Da origen en la década de 1960 con el desarrollo del lenguaje Simula, el primero en introducir los conceptos de clases y objetos, sentando las bases de este paradigma. En los años 80, ganó popularidad con lenguajes como Smalltalk, que llevó la programación orientada a objetos a un nuevo nivel al enfatizar la interacción entre objetos, y C++, que combinó la programación estructurada con la orientada a objetos, permitiendo a los desarrolladores crear sistemas más complejos y eficientes. A finales de los 90 y principios de los 2000, lenguajes como Java y C+ consolidaron la POO como un estándar en la industria, gracias a su portabilidad, robustez y facilidad de uso, lo que permitió su adopción masiva en aplicaciones empresariales, desarrollo web y móvil. Con el tiempo, la POO ha evolucionado para incluir conceptos avanzados como interfaces, que permiten definir contratos para las clases; clases abstractas, que facilitan la

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 7 de 14

creación de estructuras jerárquicas más flexibles; y la programación genérica, que permite escribir código más reutilizable y adaptable a diferentes tipos de datos. Esta evolución ha hecho que esta siga siendo relevante en el desarrollo de software moderno, adaptándose a las necesidades cambiantes de la industria y manteniéndose como uno de los paradigmas más utilizados en la actualidad.

2.3 Casos de uso


En el ámbito del desarrollo de aplicaciones empresariales, la POO es fundamental para crear sistemas complejos como los ERP (Enterprise Resource Planning) y CRM (Customer Relationship Management), que requieren modularidad, escalabilidad y facilidad de mantenimiento. En el mundo de los videojuegos, motores gráficos como Unity y Unreal Engine utilizan la POO para modelar personajes, escenarios y comportamientos, permitiendo a los desarrolladores crear experiencias interactivas y dinámicas. Además, la POO es la base de muchos sistemas operativos modernos, como Windows y macOS, donde se utiliza para gestionar recursos, procesos y interfaces de usuario de manera eficiente.

Algunos ejemplos concretos de software desarrollado con POO incluyen aplicaciones creadas con Java, que es ampliamente utilizado en el desarrollo de aplicaciones Android y sistemas bancarios debido a su portabilidad y seguridad. Python, por otro lado, es un lenguaje clave en áreas como la inteligencia artificial y el análisis de datos, donde la Programación orientada a objetos permite organizar y manipular grandes volúmenes de información de manera eficiente. En el ámbito de los videojuegos y los sistemas embebidos, C++ es una opción popular debido a su alto rendimiento y control sobre los recursos del sistema.

2.4 Lenguajes De programación asociados

Java, C++, Python, C+, Ruby, JavaScript

A continuación se muestra un código ilustrativo en Python:

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 8 de 14

```
class Cultivo:
    def __init__(self, nombre, tipo):
        self.nombre = nombre
        self.tipo = tipo

    def regar(self):
        print(f"Regando el cultivo de {self.nombre}")

# Crear un objeto
mi_cultivo = Cultivo("Maíz", "Cereal")
mi_cultivo.regar()

Regando el cultivo de Maíz
```


Este código en Python define una clase llamada Cultivo, que representa un tipo de cultivo con atributos como nombre y tipo. La clase tiene un método especial llamado `__init__`, que actúa como un constructor y se ejecuta automáticamente al crear un objeto, inicializando sus atributos con los valores proporcionados. Además, se define el método `regar()`, el cual imprime un mensaje indicando que se está regando el cultivo, accediendo al atributo nombre mediante `self.nombre`. Luego, se crea un objeto llamado `mi_cultivo` basado en la clase Cultivo, al que se le asignan los valores "Maíz" como nombre y "Cereal" como tipo. Finalmente, al llamar al método `regar()` del objeto, se imprime el mensaje "Regando el cultivo de Maíz", demostrando así el uso de la Programación Orientada a Objetos (POO) en Python mediante la creación de clases, objetos y métodos.

3. PARTE PRÁCTICA

3.1 Implementación ejemplo práctico

A continuación, se desarrollará un Sistema de Gestión de Cultivos Agrícolas utilizando el paradigma de Programación Orientada a Objetos. Este sistema permitirá registrar cultivos, asignar tareas como riego y fertilización, y monitorear su estado. El objetivo es demostrar cómo la POO puede aplicarse para resolver problemas del mundo real, en este caso, en el ámbito agrícola. El sistema está diseñado para gestionar cultivos de manera eficiente. Cada cultivo se representa como un **objeto** que contiene información como su nombre, tipo y estado. Además, el sistema permite realizar acciones específicas, como regar o fertilizar, a través de **métodos** asociados a cada cultivo.

El proyecto se divide en las siguientes funcionalidades principales:

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 9 de 14

1. **Registro de cultivos:** Crear y almacenar información sobre los cultivos.
2. **Asignación de tareas:** Realizar acciones como riego y fertilización.
3. **Monitoreo del estado:** Verificar el estado actual de los cultivos.


A continuación, se presenta un ejemplo básico del sistema implementado en Java. Este código define una clase **Cultivo** con atributos como *nombre* y *tipo*, y métodos como *regar()*. Luego, se crea un objeto de tipo **Cultivo** y se ejecuta una acción sobre él (abrir el siguiente repositorio y descargar el código en java).

Brayux/Paradigma-Programación-Orientada-a-objetos

```

J code.java
1  // Definición de la clase Cultivo
2  class Cultivo {
3      // Atributos de la clase
4      String nombre;
5      String tipo;
6
7      // Constructor para inicializar los atributos
8      public Cultivo(String nombre, String tipo) {
9          this.nombre = nombre;
10         this.tipo = tipo;
11     }
12
13     // Método para regar el cultivo
14     public void regar() {
15         System.out.println("Regando " + this.nombre);
16     }
17 }
18
19 // Clase principal para ejecutar el programa
20 public class Main {
21     public static void main(String[] args) {
22         // Crear un objeto de tipo Cultivo
23         Cultivo maiz = new Cultivo("Maíz", "Cereal");
24
25         // Ejecutar el método regar() del objeto maiz
26         maiz.regar();
27     }
28 }

```

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 10 de 14

Explicación del Código:

1. Clase Cultivo:

- Define los atributos *nombre* y *tipo*.
- Incluye un constructor para inicializar los atributos al crear un objeto.
- Contiene un método `regar()` que imprime un mensaje indicando que el cultivo está siendo regado.

2. Clase Main:

- Es el punto de entrada del programa.
- Crea un objeto **Cultivo** llamado `maiz` con el nombre "Maíz" y el tipo "Cereal".
- Llama al método `regar()` del objeto `maiz`, lo que imprime: "Regando Maíz".

3.2 Desafíos y consideraciones

- **Diseño de clases complejas:**


A medida que el sistema crece, el diseño de las clases puede volverse más complicado. Por ejemplo, si se añaden más atributos (como `fechaDeSiembra` o `necesidadesDeAgua`) o métodos (como `fertilizar()` o `cosechar()`), es importante mantener una estructura clara y organizada.

- **Gestión de relaciones entre objetos:**

En sistemas más avanzados, las relaciones entre objetos pueden volverse complejas. Por ejemplo, un cultivo podría estar asociado a un objeto *Suelo* o *Clima*, lo que requeriría el uso de composición o herencia.

- **Rendimiento y escalabilidad:**

La POO puede ser menos eficiente en términos de memoria y velocidad en comparación con la programación procedural, especialmente en sistemas con muchos objetos y relaciones complejas. Sin embargo, su ventaja en organización y mantenimiento suele compensar este inconveniente.

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 11 de 14

3.3 Comparación con otros paradigmas

- **Programación imperativa:**

Es más simple y directa para proyectos pequeños, ya que se basa en funciones y procedimientos. Sin embargo, para sistemas grandes y complejos, como un sistema de gestión agrícola, la POO ofrece una mejor organización y modularidad.

- **Programación Funcional:**

Se enfoca en funciones puras y la evitación de estado mutable. Es útil para tareas de procesamiento de datos, como el análisis de rendimiento de cultivos, pero puede resultar menos intuitiva para modelar entidades del mundo real, como cultivos o tareas agrícolas.


4. CONCLUSIONES

4.1 Resumen puntos clave

La Programación Orientada a Objetos se consolida como un paradigma poderoso y versátil que ha revolucionado la forma en que se desarrolla software. Su capacidad para organizar el código en objetos y clases facilita la creación de sistemas modulares, escalables y fáciles de mantener, lo que la convierte en una herramienta indispensable para los desarrolladores y programadores en este ámbito. Los principios fundamentales de la programación orientada a objetos son la abstracción, encapsulamiento, herencia y polimorfismo, donde no solo permiten modelar problemas del mundo real de manera intuitiva, sino que también promueven la reutilización de código, reducen la redundancia y mejoran la claridad del software. Estos principios han sido clave para el desarrollo moderno, permitiendo la creación de aplicaciones complejas que van desde sistemas operativos hasta plataformas de inteligencia artificial.

4.2 Reflexión

La importancia de la POO en la industria del software es innegable. En un mundo donde las aplicaciones son cada vez más complejas y demandantes, la POO ofrece un enfoque estructurado que permite gestionar esta complejidad de manera eficiente. Es especialmente relevante en aplicaciones de gran escala, como sistemas empresariales, plataformas de comercio electrónico, videojuegos y sistemas de inteligencia artificial, donde la organización y el mantenimiento del código son críticos. Además, su aplicabilidad no se limita al software tradicional; en campos como la ingeniería agrícola, la POO ha demostrado ser una herramienta

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 12 de 14

valiosa para modelar sistemas agrícolas, gestionar recursos de manera eficiente y automatizar procesos. Por ejemplo, se pueden desarrollar sistemas de riego inteligente, modelos predictivos para cultivos o plataformas de gestión de maquinaria agrícola, todo gracias a la capacidad de la POO para representar entidades y procesos de manera clara y eficiente.


4.3 Opinión aplicabilidad en la ingeniería agrícola

A medida que surgen nuevas tendencias tecnológicas, como la automatización industrial y la agricultura de precisión, la Programación Orientada a Objetos (POO) se posiciona como un marco sólido y flexible para desarrollar soluciones innovadoras que no solo mejoran la eficiencia, sino que también promueven la sostenibilidad. Estas tendencias están transformando industrias enteras, y la POO, con su capacidad para modelar sistemas complejos de manera organizada y modular, es una herramienta clave para aprovechar al máximo estas tecnologías emergentes.

En el campo de la automatización industrial, la POO permite modelar máquinas, líneas de producción y procesos como objetos que interactúan entre sí. Esto no solo simplifica el diseño y la implementación de sistemas automatizados, sino que también facilita su mantenimiento y escalabilidad. En la agricultura, esto podría aplicarse a maquinaria autónoma, como tractores y cosechadoras, que pueden ser programados para realizar tareas específicas con precisión, reduciendo el desperdicio de recursos y aumentando la eficiencia operativa.

La agricultura de precisión, por su parte, es un área donde la POO brilla especialmente. Este enfoque utiliza tecnologías como GPS, drones y análisis de datos para optimizar el uso de recursos como agua, fertilizantes y pesticidas. La POO permite crear modelos de software que representan cultivos, suelos, condiciones climáticas y otros factores relevantes como objetos interconectados. Por ejemplo, se podría desarrollar un sistema que analice datos históricos y en tiempo real para predecir el momento óptimo para sembrar, regar o cosechar, ajustando automáticamente las operaciones agrícolas para maximizar el rendimiento y minimizar el impacto ambiental.

En concreto, la POO puede ser clave para crear sistemas que optimicen el uso de recursos como el agua y los fertilizantes. Por ejemplo, un sistema de riego inteligente basado en POO podría incluir objetos como "Suelo", "Cultivo" y "Sistema_riego", cada uno con atributos y métodos que permitan tomar decisiones basadas en datos. El objeto "Suelo" podría tener atributos

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 13 de 14


como humedad_actual y capacidad_retención, mientras que el objeto "Sistema_riego" podría incluir métodos como calcular_cantidad_agua() y activar_riego(). Esto permitiría ajustar el riego de manera precisa, evitando el desperdicio de agua y asegurando que los cultivos reciban exactamente lo que necesitan.

Además, la POO puede ayudar a reducir el impacto ambiental al permitir el desarrollo de sistemas que minimicen el uso de químicos y recursos no renovables. Por ejemplo, un sistema de gestión de fertilizantes basado en POO podría utilizar objetos como "Análisis_de_suelo" y "Recomendación_de_fertilizante" para determinar la cantidad exacta de nutrientes que necesita un cultivo, evitando la sobrefertilización y la contaminación de los suelos y cuerpos de agua cercanos.

Finalmente, la POO puede contribuir a aumentar la productividad de los cultivos al permitir la creación de sistemas integrados que combinen datos de múltiples fuentes (como sensores, satélites y bases de datos históricas) para proporcionar recomendaciones precisas y personalizadas. Por ejemplo, un sistema de gestión agrícola basado en POO podría incluir objetos como "Pronóstico_Del_Tiempo", "Historial_De_Cultivos" y "Recomendación_De_Siembra", trabajando juntos para sugerir las mejores prácticas agrícolas en función de las condiciones actuales y futuras.

5. REPOSITORIO

Brayux/Paradigma-Programaci-n-Orientada-a-objetos

 UNIVERSIDAD NACIONAL DE COLOMBIA SEDE BOGOTÁ FACULTAD DE INGENIERÍA	PC - 18: Parcial 3			
	Código: 2015734	Versión 1	Fecha: 07-03-2025	Página 14 de 14

BIBLIOGRAFÍA Y REFERENCIAS

Programación Orientada a Objetos (POO)

Autor desconocido. (s.f.). *Programación Orientada a Objetos: Conceptos y principios fundamentales.*

Historia y evolución de la POO

Autor desconocido. (s.f.). *Evolución de la Programación Orientada a Objetos: Desde Simula hasta la actualidad.*

Aplicaciones de la POO en la agricultura

Autor desconocido. (s.f.). *Uso de la POO en la agricultura de precisión y automatización.*

Ventajas y desventajas de la POO

Autor desconocido. (s.f.). *Ventajas y desventajas de la Programación Orientada a Objetos.*

POO y nuevas tendencias tecnológicas (IoT, automatización, agricultura de precisión)

Autor desconocido. (s.f.). *La POO en el contexto de las nuevas tendencias tecnológicas.*