1. Datos Generales de la asignatura

Nombre de la asignatura:	Programación orientada a objetos
Clave de la asignatura:	IAD-2424
SATCA[1]:	2-3-5
Carrera:	Ingeniería en inteligencia artificial

2. Presentación

Caracterización de la asignatura

Esta asignatura destaca por su enfoque en introducir al estudiante al paradigma de programación orientada a objetos, ampliamente utilizado en la actualidad. Comienza con la identificación y comprensión de los objetos, explorando sus características principales y sus acciones asociadas. Luego, abarca el diseño de diagramas de clase y la implementación de soluciones prácticas a problemas reales, integrando conceptos fundamentales como abstracción, herencia y polimorfismo. Además, se profundiza en conceptos como manejo de excepciones, encapsulamiento, constructores y sobrecarga de métodos. Esta asignatura es altamente adaptable, siendo relevante para diversas ramas de la ingeniería y aplicable en una variedad de lenguajes de programación orientados a objetos, como Java, C# y Python.

Por otra parte, la asignatura fortalece el perfil de egreso desarrollando la habilidad de analizar, diseñar e implementar soluciones y modelos orientados a objetos, siguiendo estándares de calidad. El objetivo es crear proyectos que generen beneficios tangibles para la sociedad.

La asignatura de programación orientada a objetos se organiza en cinco temas:

El primer tema se centra en la identificación de objetos tanto tangibles como intangibles, junto con el reconocimiento de sus características y acciones. Luego, se exploran los conceptos fundamentales del paradigma orientado a objetos, como la abstracción, el encapsulamiento, la herencia y el polimorfismo, y se analiza su papel en el desarrollo de sistemas. Finalmente, se emplea una herramienta de software para crear diagramas de clase que representen gráficamente cada elemento, asegurándose de incluir todas las notaciones necesarias para comprender las relaciones y la multiplicidad entre las clases.

En el tema dos, se introduce a los estudiantes en la definición de clases en un lenguaje de programación, donde se crean clases con sus atributos y métodos. Se aborda el concepto de encapsulamiento para ocultar el estado interno y la funcionalidad de un objeto. Además, se explora la definición de constructores y la sobrecarga de métodos para la inicialización de valores. Por último, se analiza el manejo de excepciones para controlar posibles errores, como entradas no válidas o acceso a variables nulas, así como conversiones entre tipos de datos.

En el tema tres, se profundiza en la definición, tipos y clasificación de la herencia, explorando cómo las clases derivadas pueden heredar atributos y métodos de las clases bases para evitar la repetición de código. Se destaca la utilidad de este concepto en la reutilización y extensión de

[1] Sistema de Asignación y Transferencia de Créditos Académicos

funcionalidades en un programa. Además, se estudian las clases abstractas, resaltando sus diferencias con respecto a las clases tradicionales y cómo se integran en la solución de problemas mediante la definición de métodos abstractos que deben ser implementados por las clases derivadas.

En el tema cuatro, se explora el polimorfismo como un componente esencial del paradigma orientado a objetos. Se inicia con la comprensión de los conceptos fundamentales y se avanza hacia el desarrollo de interfaces, detallando su implementación y la técnica de sobreescritura de métodos. Además, se profundiza en el uso de variables polimórficas, destacando su capacidad para referenciar diferentes tipos de objetos de manera dinámica. Se comparan y contrastan las variables polimórficas con las clases abstractas, resaltando sus diferencias y aplicaciones específicas. Por último, se vincula el concepto de herencia con el desarrollo de interfaces para mostrar cómo estas pueden extender la funcionalidad de las clases base de manera flexible y modular.

En el tema cinco, se aborda la persistencia de datos de objetos en archivos de texto y binarios. Se comienza explorando los tipos de datos requeridos para escribir información en archivos, seguido por el análisis de los métodos para recuperar y visualizar los datos en la interfaz de usuario. Esta habilidad permite a los estudiantes realizar pruebas de almacenamiento de información, garantizando que los datos utilizados en el programa estén disponibles incluso después de que este finalice, facilitando así la continuidad y la consistencia de la aplicación.

Esta asignatura sienta las bases para asignaturas futuras relacionadas con el diseño y desarrollo de sistemas de software. La aplicación de los conceptos de modularidad en la programación orientada a objetos resulta fundamental para la gestión eficiente de algoritmos y estructuras de datos complejos en áreas como la Inteligencia Artificial. La herencia, por otro lado, permite la ampliación y especialización de las capacidades de los sistemas de IA. Además, los patrones de diseño basados en principios de programación orientada a objetos ofrecen soluciones estandarizadas y reutilizables para los desafíos comunes en el desarrollo de sistemas de IA.

Intención didáctica

El estudio de la programación orientada a objetos es fundamental en ingeniería porque proporciona un marco conceptual eficiente para el desarrollo de software. Permite la organización modular de sistemas complejos, facilita la reutilización de código y promueve la escalabilidad y mantenibilidad de las aplicaciones. Además, fomenta la abstracción y el encapsulamiento de datos y comportamientos, lo que conduce a un diseño más claro y estructurado. Asimismo, el enfoque orientado a objetos ayuda a los ingenieros a pensar de manera más abstracta y a resolver problemas de manera más efectiva, lo que resulta en un código más limpio, modular y fácil de entender y mantener.

El profesor de esta asignatura debe demostrar su experiencia y conocimiento en el desarrollo de aplicaciones orientadas a objetos en al menos un lenguaje de programación. Es importante que el docente pueda explicar de manera clara y concisa los conceptos abstractos y complejos relacionados con la programación orientada a objetos, y proporcionar ejemplos y ejercicios que ayuden a los estudiantes a comprender y aplicar estos conceptos en situaciones reales. Asimismo, debe estar al tanto de las tendencias y avances en el campo de la programación orientada a objetos para ofrecer una educación actualizada y relevante.

Por otra parte, se proponen ejercicios específicos que permiten al estudiante identificar los aspectos esenciales de cada tema, conduciendo gradualmente hacia una comprensión más profunda y generalizada.

El estudiante tendrá la oportunidad de desarrollar habilidades para modelar situaciones reales utilizando las herramientas y conceptos aprendidos en el curso.

Es esencial que el estudiante reconozca el valor de las actividades realizadas, desarrollando hábitos de estudio y trabajo que fomenten la curiosidad, la puntualidad, el entusiasmo, la perseverancia y la autonomía.

El estudio programación orientada a objetos contribuye al desarrollo de habilidades analíticas y de síntesis, capacidad para identificar y resolver problemas, capacidad de aplicar los conocimientos en la práctica, habilidades de investigación, solución de problemas y toma de decisiones.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Tecnológico Nacional de México del 4 al 6 de marzo de 2024	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán. Tecnológico de Estudios Superiores de Ixtapaluca.	Propuesta sintética de la carrera de Ingeniería en Inteligencia Artificial
Tecnológico Nacional de México del 22 al 26 de abril de 2024	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán, Tecnológico de Estudios Superiores de Ixtapaluca.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Inteligencia Artificial
Tecnológico Nacional de México del 27 al 31 de mayo de 2024	Representantes de los Institutos Tecnológicos de: Celaya, La Paz, Matehuala, Mérida, Minatitlán.	Consolidación curricular de la carrera de Ingeniería en Inteligencia Artificial

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Conoce y aplica el paradigma de programación orientada a objetos para el desarrollo de soluciones de problemas prácticos y situaciones del mundo real.

5. Competencias previas

- Implementa algoritmos para abordar y resolver distintos tipos de problemas.
- Utiliza un editor de texto para escribir y compilar programas de forma eficiente.
- Emplea variables, tipos de datos, estructuras de control y módulos de un lenguaje de programación para la solución de distintos problemas.

6. Temario

No.	Temas	Subtemas
1	Introducción al paradigma de la programación orientada a objetos	1.1. Definición. 1.1.1. Clases y objetos 1.2. Principios. 1.2.1. Abstracción. 1.2.2. Encapsulamientos. 1.2.3. Herencia. 1.2.4. Polimorfismo. 1.3. Diagrama de clasesUML. 1.3.1. Notaciones.
2	Clases y objetos	2.1. Definición de clases. 2.2. Declaración de clases. 2.2.1. Atributos. 2.2.2. Métodos. 2.2.3. Encapsulamiento. 2.2.4. Constructores. 2.2.5. Sobrecarga de métodos. 2.3. Creación de objetos. 2.4. Referencia al objeto actual. 2.5. Manejo de excepciones.
3	Herencia	3.1. Definición, 3.2. Clasificación. 3.3. Clase base y clase derivada. 3.4. Referencia a clase base 3.5. Atributos y método heredados 3.6. Implementación de clase derivada. 3.7. Clases abstractas 3.8. Métodos Abstractos.

4	Polimorfismo	4.1. Definición. 4.2. Interfaces. 4.2.1. Declaración. 4.2.2. Implementación. 4.2.3. Sobreescritura de métodos. 4.2.4. Diferencia con clases abstractas. 4.3. Variables polimórficas. 4.4. Reutilización de código. 4.5. Herencia de interfaces.
5	Flujos y archivos	5.1. Definición. 5.2. Clasificación. 5.2.1. Archivos de texto. 5.2.2. Archivos binarios. 5.3. Manejo de archivos. 5.3.1. Persistencia.

7. Actividades de aprendizaje de los temas

1. Introducción al paradigma de la programación orientada a objetos	
Competencias	Actividades de aprendizaje
Específica(s): • Comprende el concepto de la programación orientada a objetos y distingue las principales características de cada uno de sus pilares. • Diseña y desarrolla diagramas de clases completos, incluyendo todas las notaciones necesarias para una representación precisa y comprensible. *Genérica(s):* • Capacidad de adaptarse a nuevas situaciones. • Capacidad de aprender. • Capacidad de comunicar sus ideas. • Habilidad para trabajar en forma autónoma. • Habilidades de investigación.	• Investigar en múltiples fuentes los conceptos fundamentales de la programación orientada a objetos. • Elaborar un mapa mental que represente los pilares clave de la programación orientada a objetos, utilizando imágenes claras y fáciles de interpretar. • Crear una infografía que visualice varios objetos, resaltando tanto sus atributos como sus acciones de manera clara y concisa. • Realizar una investigación sobre diversas herramientas de software de código abierto disponibles para la creación de diagramas de clases. • Seleccionar una herramienta de software y utilizarla para construir diagramas de clases completos, empleando todas las componentes y notaciones necesarias.

2. Clases y objetos	
Competencias	Actividades de aprendizaje
Específica(s): • Define, crea e implementa clases y objetos en la programación, aplicando el paradigma orientado a objetos para resolver diversos problemas de manera efectiva. *Genérica(s):* • Búsqueda del logro. • Capacidad de aplicar los conocimientos en la práctica. • Capacidad de aprender • Capacidad de comunicar sus ideas. • Habilidad para trabajar en forma autónoma. • Habilidades de investigación • Habilidad para manejo de equipo de cómputo. • Solución de problemas. • Toma de decisiones.	• Investigar las diferencias entre una clase y un objeto para elaborar un informe que explique sus características y su relación en el contexto de la programación orientada a objetos. • Investigar los lenguajes de programación orientados a objetos y seleccionar uno adecuado para el desarrollo de clases, justificando la elección en función de sus características y capacidades. • Seleccionar un conjunto de objetos tangibles e intangibles y definir sus atributos y métodos para la creación de clases, promoviendo la comprensión de la relación entre objetos y clases. • Investigar el concepto de encapsulamiento y aplicarlo en la definición de atributos en las clases previamente creadas, asegurando la seguridad en el diseño de los objetos. • Implementar métodos y constructores sobrecargados en las clases para permitir la inicialización de objetos con diferentes conjuntos de parámetros, mejorando la modularidad y la eficiencia del código. • Realizar una investigación sobre las principales excepciones en el lenguaje de programación seleccionado y elaborar un mapa conceptual que presente de manera clara y concisa las relaciones entre cada tipo de excepción. • Aplicar el manejo de excepciones en el desarrollo de clases y objetos, implementando estrategias para prevenir y gestionar errores durante la ejecución del programa, garantizando la estabilidad del software desarrollado.

3. Herencia	
Competencias	Actividades de aprendizaje
Específica(s): Implementa los conceptos de herencia para diseñar clases base y derivadas, así como para emplear clases abstractas, con el propósito de desarrollar soluciones de software de forma eficiente y modular. *Genérica(s):* ● Búsqueda del logro. ● Capacidad de aplicar los conocimientos en la práctica. ● Capacidad de aprender ● Capacidad de comunicar sus ideas. ● Habilidad para trabajar en forma autónoma. ● Habilidades de investigación ● Habilidad para manejo de equipo de cómputo. ● Solución de problemas. ● Toma de decisiones.	● Crear un cuadro sinóptico que presente de manera concisa y ordenada los principales conceptos relacionados con la herencia en programación orientada a objetos. ● Elaborar una infografía que ilustre un ejemplo de una clase base y al menos tres clases derivadas, destacando sus atributos y métodos de manera clara y comprensible. ● Desarrollar una aplicación utilizando un lenguaje de programación orientado a objetos que integre el uso de una clase base y al menos tres clases derivadas, demostrando la aplicación práctica de los conceptos de herencia en el desarrollo de software. ● Diseñar y elaborar un diagrama de clases que incluya clases abstractas, mostrando de manera gráfica y organizada la relación entre las clases base, las clases derivadas y las clases abstractas en el contexto de un sistema de software. ● Crear un ejercicio práctico que implemente la herencia utilizando al menos una clase abstracta y sus métodos respectivos, con el objetivo de fortalecer la comprensión y aplicación de este concepto fundamental en la programación orientada a objetos.
4. Polimorfismo	
Competencias	Actividades de aprendizaje
Específica(s): Aplica el concepto de polimorfismo en la elaboración de programas a través de la utilización de interfaces, lo que posibilita la creación de soluciones con mayor flexibilidad y escalabilidad. *Genéricas:* ● Búsqueda del logro. ● Capacidad de aplicar los conocimientos en la práctica. ● Capacidad de aprender ● Capacidad de comunicar sus ideas. ● Habilidad para trabajar en forma autónoma. ● Habilidades de investigación	● Investigar el concepto de interfaces y realizar una comparación con clases abstractas para elaborar un cuadro comparativo. ● Definir una interfaz con sus métodos abstractos que se implementen en una clase. Por ejemplo, definir operaciones como cálculo de áreas, salarios, promedios, entre otros. ● Desarrollar ejemplos de sobreescritura de métodos. Por ejemplo, definir una interfaz "Reproductor Música" con métodos como "pausar ()", "continuar ()" y "detener ()". Luego, implementar subclases sobre diferentes reproductoras utilizando esta interfaz y adaptando la implementación de los métodos según las características específicas de cada reproductor.

• Habilidad para manejo de equipo de cómputo. • Solución de problemas. • Toma de decisiones.	• Codificar un ejercicio donde se trabaje con variables polimórficas en un ejercicio sobre figuras geométricas para calcular el área de diferentes figuras. • Crear una aplicación que utilice la herencia de interfaces para determinar los comportamientos que las clases podrán implementar.
5. Flujos y archivos	
Competencias	Actividades de aprendizaje
Específica(s): Reconoce los distintos tipos de archivos y aplica el concepto de persistencia de datos en el desarrollo de proyectos. *Genéricas:* • Búsqueda del logro. • Capacidad de aplicar los conocimientos en la práctica. • Capacidad de aprender • Capacidad de comunicar sus ideas. • Habilidad para trabajar en forma autónoma. • Habilidades de investigación • Habilidad para manejo de equipo de cómputo. • Solución de problemas. • Toma de decisiones.	• Realizar una investigación sobre los principales tipos de persistencia de datos en el desarrollo de software y crear un mapa conceptual que los represente. • Elaborar un cuadro comparativo que destaque las características de los archivos de texto y binarios utilizados como formas de persistencia de datos. • Desarrollar una aplicación que utilice archivos de texto como método de persistencia de datos. • Crear una aplicación que emplee archivos binarios como mecanismo de persistencia de datos.

8. Práctica(s)

Prácticas para el tema 1:

- Elaborar un diagrama de clases que represente las entidades de la institución. Para cada elemento, se deben incluir al menos cinco atributos y dos métodos, así como establecer las relaciones correspondientes entre ellos.
- Crear un diagrama de clases que refleje los elementos seleccionados por los alumnos, como videojuegos, deportes, grupos musicales, vehículos y actividades culturales, entre otros. Luego, en colaboración con todos los estudiantes, se asignan atributos, métodos y relaciones a cada elemento.

Prácticas para el tema 2:

- Elegir tres entidades tangibles y tres entidades intangibles para desarrollar sus clases correspondientes, incluyendo atributos, métodos, constructores y sobrecarga de métodos.
- Definir la clase "Galleta" y crear al menos ocho instancias con diversas características como sabores, tamaños, precios y formas.
- Implementar excepciones para gestionar posibles errores relacionados con la entrada de datos, su rango y su tipo durante la interacción por consola.

Prácticas para el tema 3:

- Identificar tres clases base que sirvan como modelos para la herencia de atributos y métodos, y aplicar este concepto en clases derivadas.
- Crear la clase base "Trabajador" con atributos como nombre, edad y género. Luego, desarrollar clases derivadas como "Chofer", "Panadero" y "Carpintero", que hereden los atributos de la clase "Trabajador".
- Definir una clase abstracta denominada "Habitación", la cual incluya un método abstracto para calcular el total a pagar por servicios. Esta clase debe considerar distintos tipos de habitaciones y servicios disponibles en el hotel.

Prácticas para el tema 4:

- Crear una interface para representar a un estudiante de cualquier carrera, permitiendo a los alumnos proponer los métodos que deberían ser implementados en las clases.
- Codificar una interfaz para gestionar datos en un arreglo, incluyendo métodos para consultar, insertar y obtener la cantidad de datos almacenados.
- Desarrollar una solución que combine herencia y polimorfismo, como una aplicación de pago de impuestos donde la herencia represente los diferentes tipos de impuestos con sus respectivos costos, y la interfaz permite calcular el subtotal, el descuento y el total a pagar.

Prácticas para el tema 5:

- Desarrollar un proyecto que incorpore herencia y polimorfismo, almacenando los datos en un archivo de texto plano para asegurar su persistencia después de que el programa se cierre. El docente puede adaptar una práctica del tema cuatro para esta actividad.

- Implementar un proyecto que combine herencia y polimorfismo, almacenando los datos en un archivo binario para garantizar su persistencia incluso al cerrar el programa. El docente puede adaptar una práctica del tema cuatro para esta actividad.

  Codificar un proyecto relacionado con la venta de boletos de una aerolínea, donde la clase abstracta sea "Pasajero" y las clases derivadas sean "Niños", "Adultos", "Maestros" y Personas con credencial “INAPAM". Se debe programar una interfaz para calcular el total a pagar considerando los descuentos aplicables. Finalmente, toda la información debe ser almacenada en un archivo de texto o binario para su conservación.

9. Proyecto de asignatura

El objetivo del proyecto que planteé el docente que imparta esta asignatura, es demostrar el desarrollo y alcance de la(s) competencia(s) de la asignatura, considerando las siguientes fases:

Fundamentación: marco referencial (teórico, conceptual, contextual, legal) en el cual se fundamenta el proyecto de acuerdo con un diagnóstico realizado, mismo que permite a los estudiantes lograr la comprensión de la realidad o situación objeto de estudio para definir un proceso de intervención o hacer el diseño de un modelo.

Planeación: con base en el diagnóstico en esta fase se realiza el diseño del proyecto por parte de los estudiantes con asesoría del docente; implica planificar un proceso: de intervención empresarial, social o comunitario, el diseño de un modelo, entre otros, según el tipo de proyecto, las actividades a realizar los recursos requeridos y el cronograma de trabajo.

Ejecución: consiste en el desarrollo de la planeación del proyecto realizada por parte de los estudiantes con asesoría del docente, es decir en la intervención (social, empresarial), o construcción del modelo propuesto según el tipo de proyecto, es la fase de mayor duración que implica el desempeño de las competencias genéricas y específicas a desarrollar.

Evaluación: es la fase final que aplica un juicio de valor en el contexto laboral-profesión, social e investigativo, ésta se debe realizar a través del reconocimiento de logros y aspectos a mejorar se estará promoviendo el concepto de "evaluación para la mejora continua", la metacognición, el desarrollo del pensamiento crítico y reflexivo en los estudiantes.

10. Evaluación por competencias

Para evaluar las actividades de aprendizaje, se recomienda solicitar una amplia gama de entregables, que incluyan desde mapas conceptuales o mentales hasta prototipos y estudios de casos. Es fundamental que los estudiantes demuestren sus habilidades mediante ejercicios prácticos que reflejen su comprensión y aplicación de los conceptos aprendidos a lo largo del curso.

Para la verificación del nivel de logro de las competencias, se pueden utilizar herramientas como listas de cotejo, matrices de valoración, guías de observación y rúbricas, que proporcionen una evaluación detallada y objetiva del desempeño de los estudiantes.

11. Fuentes de Información

1. Blanco Fernández, Y. (2020). Introducción a Programación Orientada a Objetos. España: Andavira.
2. Blasco, F. (2019). Programación Orientada a Objetos en JAVA. Malaga: RA-MA S.A. Editorial y Publicaciones.
3. F Lott, S., & Phillips, D. (2021). Python Object-Oriented Programming. Packt Publishing.
4. Hernández Bejarano, M., & Baquero Rey, L. E. (2023). Programación Orientada a Objetos en Java. Bogotá: Ediciones de la U.
5. Jiménez de Parga, C. (2021). UML. Arquitectura de aplicaciones en Java, C++ y Python. 2ª Edición. España: RA-MA S.A. Editorial y Publicaciones.
6. Jiménez Marín, A., & Pérez Montes, F. M. (2021). Programación. Madrid: Ediciones Paraninfo, S.A.
7. Joyanes Aguilar, L. (2014). PROGRAMACION EN C, C++, JAVA Y UML. España: MCGRAW HILL EDUCATION.
8. Kalb, I. (2022). Object-Oriented Python: Master OOP by Building Games and GUIs. San Francisco: No Starch Press.
9. SOMASHEKARA, M. T., GURU, D. S., & MANJUNATHA, K. S. (2017). Object Oriented Programming with Java. New Delhi: PRENTICE HALL.
10. Taher, R. (2019). Hands-On Object-Oriented Programming with C#. Inglaterra: Packt Publishing.
11. Veeramani, A., & Vardhan, H. (2023). El UML del aprendizaje profundo. Ediciones Nuestro Conocimiento.