Programa de Estudios de la Unidad de Aprendizaje: PROGRAMACIÓN ORIENTADA A OBJETOS

Clave:	5FP-FM1194	Créditos:	3.37

Ramas de Conocimiento

Ingeniería y Ciencias Físico Matemáticas	X	Ciencias Sociales Administrativas		Ciencias Médico Biológicas	

Área de Formación Curricular

Institucional		Científica, Humanística y Tecnológica Básica		Profesional	X

Tipo de Espacio

Aula		Taller		Laboratorio	X	Otros ambientes de Aprendizaje	

Modalidad

Escolarizada	X	No Escolarizada		Mixta	

Vigencia a Partir:	AGOSTO 2024

Proceso de Diseño y Autorización:			Día	Mes	Año
Elaborado por:	REP. ACAD. NMS	Fecha de Elaboración:	01	10	2023
Revisado por:	DEMS	Fecha de Revisión:	17	05	2024
Aprobado por:	CTCE-NMS	Fecha de Aprobación:	12	06	2024
Autorizado por:	CPA-CGC	Fecha de Autorización:	20	06	2024

Programa Académico:	TÉCNICO EN SISTEMAS DIGITALES

Nivel:	1°	2°	3°	4°	5°	6°

Unidades Académicas donde se Imparte:

TODAS LAS U.A.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	CET1

Tiempos Asignados:

Global:	54	Hrs/18 semanas/Semestre		
Aula:	-	Hrs/Semana	Total:	- Hrs/Semestre
Taller:	-	Hrs/Semana	Total:	- Hrs/Semestre
Laboratorio:	3	Hrs/Semana	Total:	54 Hrs/Semestre
Otros ambientes de aprendizaje:	-	Hrs/Semana	Total:	- Hrs/Semestre

Organización

Por Unidad de Aprendizaje:	X	Por Área:		Por Módulo:	

Firma y Sello de Autorización:

INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

M. EN E.N.A. MARÍA ISABEL ROJAS RUÍZ
Directora de Educación Media Superior

Programa Académico: **Técnico en Sistemas Digitales** **Unidad de Aprendizaje**: **Programación Orientada a Objetos**

FUNDAMENTACIÓN

La unidad de aprendizaje **Programación Orientada a Objetos** pertenece al área de formación profesional del Bachillerato Tecnológico Bivalente del Nivel Medio Superior del Instituto Politécnico Nacional, se ubica en el quinto nivel del Programa Académico Técnico en Sistemas Digitales y se imparte en la modalidad escolarizada, de manera optativa en la rama del conocimiento de Ingeniería y Ciencias Físico - Matemáticas.

La unidad de aprendizaje de Programación Orientada a Objetos promueve el análisis, desarrollo e implementación de programas básicos, mediante el empleo de un lenguaje de programación orientado a objetos que le permitan dar solución a problemas reales, tomando en cuenta el contexto de las dimensiones científica, técnica y tecnológica; de una forma social, responsable, reflexiva, metodológica y sustentable; que incentive la adquisición, desarrollo y aplicación del razonamiento abstracto, el pensamiento analítico, la creatividad, la innovación, el emprendimiento y diversas habilidades cognitivas

Esta Unidad de aprendizaje introduce al estudiante al campo conceptual, procedimental y actitudinal para dar solución a problemas de su entorno, aplicando la programación orientada a objetos, considerando tanto los principios y ejes del desarrollo humano sustentable como la perspectiva de género. La adquisición de estas destrezas y habilidades relacionadas con la implementación de programas orientados a objetos básicos favorecerán en el estudiante el desarrollo de una visión crítica y holística, cuya puesta en práctica, en forma autónoma, en el futuro le coadyuvará a responder en forma eficiente y eficaz a los retos que se le presenten cuando se incorpore a estudios superiores o al campo laboral.

Programación Orientada a Objetos es una Unidad de Aprendizaje enfocada al desarrollo de habilidades técnicas, cognitivas y socioemocionales inherentes al estudio, análisis e implementación de programas básicos, mediante el empleo de un lenguaje orientado a objetos, aplicando enfoques didácticos al trabajo colaborativo, la autonomía y ubicuidad a través de diversas herramientas, orientados a solucionar problemas elementales de la cotidianeidad.

La unidad de aprendizaje Programación Orientada a Objetos está fundamentada en el Modelo Educativo Institucional vigente, por lo que se emplearán metodologías didácticas activas como el Aprendizaje basado en Proyectos, Método de situaciones o de casos, Aprendizaje basado en problemas, Aula invertida, Trabajo colaborativo, Gamificación; esto con el propósito de que el estudiante desarrolle competencias del siglo XXI, como el trabajo colaborativo, trabajo en equipo, reto al cambio, autodirección, resolución de problemas cercanos a la realidad, autogestión del aprendizaje y resiliencia. Además, se emplearán herramientas tecnológicas que fomentarán la colaboración e interacción presenciales'.

El rol del docente será de mediador entre el estudiante y los contenidos didácticos a abordar, puesto que se centrará en la creación, organización, supervisión y mediación de los espacios de trabajo, incluidos los ciberespacios, atendiendo las necesidades técnicas, de conocimientos, apoyo logístico y metodológico en los procesos de aprendizaje individual y grupal, con el objetivo de generar ambientes que favorezcan la educación inclusiva, flexible, sustentable y con perspectiva de género.

El estudiante desarrollará un trabajo autónomo en diferentes ambientes de aprendizaje, organizará su trabajo de manera independiente y articulará saberes de diversos campos del conocimiento, que le posibilitarán construir y expresar su propio conocimiento en beneficio de la sociedad; también adquirirá habilidades tanto tecnológicas como personales que promoverán la comunicación asertiva, la creatividad, la negociación, la gestión del tiempo, la motivación, el liderazgo y la responsabilidad social vinculada a la protección del medio ambiente, la erradicación de toda manifestación de violencia de género, la inclusión y la accesibilidad.

Considerando que el trabajo en laboratorios es esencial para el proceso de aprendizaje, ya que permite al docente verificar el nivel del logro de las competencias a desarrollar, se requiere generar ambientes de trabajo que favorezcan un monitoreo especializado y personalizado a los estudiantes, debido a esto es necesario se trabaje con un docente titular y dos docentes adjuntos que lo apoyen en las actividades, logrando mayor calidad en el proceso enseñanza aprendizaje.

La evaluación se efectuará en el marco de la evaluación auténtica, por esto, comprenderá tres momentos: diagnóstica, formativa y sumativa. La evaluación diagnóstica se llevará a cabo mediante un cuestionario informatizado con evaluación y retroalimentación automatizadas, la finalidad es que el docente efectúe los ajustes didácticos pertinentes y que el estudiante conozca y, si es necesario, nivele sus conocimientos previos adquiridos en otras unidades de aprendizaje para que establezca conexiones significativas con la propuesta didáctica de la unidad de aprendizaje Programación Orientada a Objetos.

Un segundo momento de la evaluación hace referencia a la evaluación formativa, que se desarrollará a lo largo del proceso de enseñanza-aprendizaje mediante las secuencias didácticas y actividades de aprendizaje formativas que estimulen el aprendizaje activo y significativo del estudiante; este momento se enriquecerá con diversos tipos de evaluación, como la autoevaluación, coevaluación y la heteroevaluación, puesto que coadyuvarán a dar seguimiento al desarrollo de los saberes y habilidades en contexto. Cabe señalar que estas clases de evaluación serán reforzadas a través de la retroalimentación efectiva y oportuna. En el tercer momento de la evaluación, con fines de acreditación, se diseñarán situaciones integradoras que permitan recuperar el nivel de logro y conducir al estudiante a la metacognición en la unidad de aprendizaje Programación Orientada a Objetos, esto mediante evidencias de conocimiento, producto y desempeño en la comprensión sólida de los fundamentos del paradigma orientado a objetos, incluyendo la creación de clases y objetos, la herencia, el polimorfismo, la encapsulación, el manejo de excepciones y el modelado de sistemas, preparándolos para abordar problemas de programación de manera modular y efectiva, y dotándolos de las habilidades necesarias para desarrollar aplicaciones de software empleando las mejores prácticas de desarrollo y pensamiento abstracto y cuyos criterios, aspectos e indicadores serán conocidos por los estudiantes en forma previa. Las evidencias de evaluación formativa e integradora mostrarán el saber hacer de manera reflexiva de los estudiantes, utilizando el conocimiento que van adquiriendo durante el proceso didáctico para luego transferir ese aprendizaje a situaciones similares y diferentes, en contextos escolares, sociales y laborales.

Con base en la flexibilidad curricular y en el reconocimiento de aprendizajes múltiples, también podrá aplicarse una evaluación para verificar que el estudiante domina los saberes y propósitos de la unidad de aprendizaje de Programación Orientada a Objetos, previo a su inicio. De esa forma, este programa de estudios tiene una naturaleza normativa, puesto que establece los estándares para el desarrollo de conocimientos, habilidades prácticas del área de formación, habilidades socioemocionales, actitudes y valores.

Es importante remarcar que se requiere un docente titular para la unidad de aprendizaje de Programación Orientada a Objetos que defina los lineamientos y 2 docentes auxiliares que ayudaran a monitorear y reforzar el proceso académico de los estudiantes, de tal manera que la asesoría pueda ser más especializada.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

DESCRIPCIÓN DE LA UNIDAD DE APRENDIZAJE

Unidad de Aprendizaje: PROGRAMACION ORIENTADA A OBJETOS		
Propósito de la Unidad de Aprendizaje Desarrolla aplicaciones de software empleando la programación orientada a objetos (POO) para satisfacer las necesidades de los usuarios de forma eficiente y con responsabilidad social.		
Unidad 1: INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS		
Unidad de competencia	Aprendizajes esperados	Contenidos de aprendizaje
Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.	Compara las características de clase y objeto identificando los conceptos básicos de la POO para su uso correcto en la resolución de situaciones del mundo real.	**Conceptuales:** • Definición de Programación Orientada a Objetos (POO). • Características de la POO. • Concepto de clases y objetos. **Procedimentales:** • Reconoce los conceptos y las partes de clase y objeto, identificando sus diferencias en código predefinido. • Relaciona las características de clase y objeto para determinar su uso correcto dentro de programa orientado a objetos. ✓ Práctica N°1 "Instancia de objeto y métodos de entrada y salida de datos" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.
	Usa los principios de encapsulamiento en la programación orientada a objetos (POO), identificando los tipos de acceso a una clase, para asegurar la integridad y seguridad de los datos.	**Conceptuales:** • Modificadores de acceso (públicos, privados o protegidos). • Principios del encapsulamiento (Ocultamiento de la información, protección de la integridad de los datos, abstracción, facilidad de mantenimiento y de evolución del código). **Procedimentales:** • Compara las ventajas y las desventajas de los diferentes modificadores de acceso a la encapsulación de datos

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

		• Hace uso de principios del encapsulamiento para la implementación de clases y objetos correcto dentro de programa orientado a objetos. ✓ Práctica N°2 "Principios de Encapsulamiento en la POO" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.
	Estructura diagramas de clases básicos utilizando Lenguaje Unificado de Modelado (UML) para representar relaciones entre clases y objetos.	**Conceptuales:** • Definición del Lenguaje Unificado de Modelado (UML) • Diagramas de clases. **Procedimentales:** • Reconoce las características del Lenguaje Unificado de Modelado para la identificación de los componentes modulares del sistema. • Realiza diagramas de clases simples agrupando correctamente atributos y métodos de un objeto del mundo real para representar las relaciones entre estos dentro de su programa. ✓ Práctica N°3 "Modelado de clases con UML" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.
Unidad 2: CODIFICACIÓN		
Unidad de competencia	Aprendizajes esperados	Contenidos de aprendizaje
Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante	Construye una nueva clase estableciendo la relación entre superclases y subclases para mejorar la reutilización del código, el mantenimiento, la extensibilidad, la flexibilidad y la organización de este.	**Conceptuales:** • Herencia. • Relación de superclase y subclase. **Procedimentales:** • Identifica las características de una superclase y una subclase para comprender la relación entre estos.

el empleo de una comunicación asertiva.		• Reconoce el concepto y el uso de la herramienta de herencia para ilustrar la manera eficiente de reutilizar el código • Realiza una nueva clase mediante el uso de herencia para reutilizar el código de forma adecuada dentro de su programa. ✓ Práctica N°4 "Implementación de herencia en POO" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados
	Simplifica la codificación de un programa orientado a objetos mediante el uso del polimorfismo adecuando su comportamiento a las necesidades del código.	**Conceptuales:** • Creación de objetos. • Polimorfismo (Definición, características y ventajas). **Procedimentales:** • identifica las características, los beneficios y las desventajas de hacer uso de métodos polimórficos en el desarrollo de un código • Implementa métodos polimórficos a partir de una clase base para adaptarlos a diferentes procesos dentro de un programa. ✓ Práctica N°5 "Desarrollo colaborativo de código polimórfico". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.
Unidad 3: APLICACIÓN PRÁCTICA		
Unidad de competencia	Aprendizajes esperados	Contenidos de aprendizaje
Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos	Establece excepciones en la programación, mediante la implementación de bloques try-except, con el propósito de prevenir la interrupción abrupta del programa.	**Conceptuales:** • Definición de excepción • Excepción try-except. • Importancia en las aplicaciones. **Procedimentales:** • Reconoce el concepto excepción try-except para identificar las situaciones en las que se puede ejecutar.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje: Programación Orientada a Objetos**

prácticos, apoyándose del trabajo colaborativo.		• Resuelve errores comunes durante la ejecución del programa mediante el uso de excepciones para evitar la detención abrupta de esta. ✓ Práctica N°6 "Implementación de excepciones en POO". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados
	Selecciona cada uno de los elementos de la programación orientada a objetos (POO), de manera creativa para diseñar aplicaciones que resuelvan problemas reales de manera efectiva.	**Conceptuales:** • Modelado de aplicaciones mediante la POO. • Estilo del programador **Procedimentales:** • Estructura aplicaciones de software utilizando la POO para dar solución a problemas del mundo real. ✓ Práctica N°7 "Diseño de aplicación de software mediante POO". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

Programa Académico: Técnico en Sistemas Digitales

Unidad de Aprendizaje: Programación Orientada a Objetos

MATRIZ DE VINCULACIÓN

	Unidad de Competencia 1			Unidad de Competencia 2		Unidad de Competencia 3	
COMPETENCIAS PARA EL SIGLO XXI HABILIDADES BLANDAS Y SOCIOEMOCIONALES	**AE 1**	**AE 2**	**AE 3**	**AE 1**	**AE 2**	**AE 1**	**AE 2**
Pensamiento lógico y matemático.	X	X	X	X	X	X	X
Trabajo Colaborativo	X	X	X		X		X
Pensamiento crítico	X	X	X	X	X	X	X
Reflexiona sobre los aprendizajes desarrollados	X	X	X	X	X	X	X
Comunicación asertiva	X	X	X		X		X

Programa Académico: **Técnico en Sistemas Digitales** **Unidad de Aprendizaje**: **Programación Orientada a Objetos**

PERFIL DOCENTE

El docente que imparta la Unidad de Aprendizaje Programación Orientada a Objetos, contará con las habilidades en el manejo de los saberes disciplinares, profesionales, así como el dominio de los temas en el área de los sistemas digitales, uso de lenguajes de programación de alto nivel y entornos de desarrollo visuales, dominio de software aplicado a la programación orientada a objetos, uso de las Tecnologías de la Información y la Comunicación, dominio de las Tecnologías del Aprendizaje y Conocimiento, control de grupo, estrategias de aprendizaje que permitan al estudiante ser el centro de su proceso educativo, fomento de una comunicación asertiva, reflexiva y crítica, deberá mostrar una conducta basada en la normatividad del Instituto Politécnico Nacional y el Modelo Educativo vigente.

Colaborará de forma colegiada, en la construcción de un proyecto de formación integral dirigido a los estudiantes, el cual deberá considerar su contexto social y brindará de forma constante el apoyo técnico pedagógico a estudiantes y personal de la Unidad Académica que lo requiera.

Habilidades docentes en el desarrollo del Talento

En el campo de su especialización:

- Dominio del estilo de programación orientada a objetos.
- Manejo de Entornos de Desarrollo Integrado (IDE) para lenguajes orientados a objetos.
- Experiencia en el desarrollo de proyectos de forma colaborativa, para la solución de problemáticas del entorno social.
- Actualiza las habilidades digitales para desarrollarlas e implementarlas en el aula.
- Desarrollar procesos de enseñanza aprendizaje, utilizando métodos basados en administración de proyectos reales, para mejorar la calidad y pertinencia de la enseñanza.

En el campo pedagógico:

- Fomenta procesos de enseñanza que le permitan interpretar y resolver las necesidades de aprendizaje de los estudiantes, tomando en cuenta sus capacidades, habilidades, vocación e intereses.
- Planea las clases considerando las características diversas de los estudiantes, el contexto institucional y el trabajo colaborativo.
- Diseña planeaciones didácticas incorporando el uso de herramientas tecnológicas y recursos digitales
- Llevar a la práctica el proceso de E-A, de forma efectiva, creativa e innovadora, en el contexto institucional.
- Evalúa los aprendizajes tomando en cuenta los propósitos curriculares y particularidades de los estudiantes.
- Fomenta la participación activa de los estudiantes sin discriminación.
- Implementa metodologías activas para incentivar en los estudiantes el pensamiento crítico, ético, solidario, ecológico y sustentable.
- Propone actividades o retos de acuerdo con propósitos o competencias específicas.

En el campo de la investigación:

- Fortalece el trabajo académico a partir del aprovechamiento de los resultados y productos de los proyectos de investigación.
- Fomenta la investigación y desarrollo tecnológico, como estímulo para la actividad intelectual creadora.
- Está atento a los avances científicos y tecnológicos dentro del campo disciplinar.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje: Programación Orientada a Objetos**

Perfil Profesional (titular y auxiliar)

- Licenciado en Ing. en Computación, Ing. en Sistemas Computacionales, Ing. en Comunicaciones y Electrónica, Ing. en Robótica, Ing. en Telemática, Ing. en Control y Automatización, Lic. o Ing. en Informática, o áreas afines al campo de la programación; con experiencia mínima de 1 año en el área docente.
- Experiencia comprobable de 1 año en la iniciativa pública o privada aplicando los conocimientos de la Unidad de Aprendizaje.
- Nivel comprobable de conocimiento del idioma inglés.

Es importante remarcar que se requiere un docente titular para la unidad de aprendizaje de Programación Orientada a Objetos que defina los lineamientos y 2 docentes auxiliares que ayudaran a monitorear y reforzar el proceso académico de los estudiantes, de tal manera que la asesoría pueda ser más especializad

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

ESTRUCTURA DIDÁCTICA

Unidad Didáctica 1:	Introducción a la Programación Orientada a Objetos	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 1:	Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.		
Aprendizaje Esperado No 1:	Compara las características de clase y objeto identificando los conceptos básicos de la POO para su uso correcto en la resolución de situaciones del mundo real.	**Tiempo estimado para obtener el Aprendizaje Esperado:**	3 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Definición de Programación Orientada a Objetos (POO). • Características de la POO. • Concepto de clases y objetos.	• Reconoce los conceptos y las partes de clase y objeto, identificando sus diferencias en código predefinido. • Relaciona las características de clase y objeto para determinar su uso correcto dentro de programa orientado a objetos. ✓ Práctica N°1 "Instancia de objeto y métodos de entrada y salida de datos"	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Aula Invertida

En el aula:

El estudiante:

- Contesta el cuestionario de evaluación diagnóstica mediante herramientas digitales como: Microsoft Forms, formularios Google, Moodle, Classkick, etc., con la finalidad de identificar los conocimientos previos de los estudiantes con respecto a los conocimientos y habilidades de programación y algoritmia.
- Realiza una investigación sobre los conceptos básicos de la Programación Orientada a Objetos (POO) según los criterios establecidos por el docente.
- Recolecta información relevante sobre elementos del modelo de objetos, características y ventajas de la POO.
- Trabaja en parejas para elaborar un organizador gráfico que muestre los conceptos clave de la POO, así como sus características y ventajas.
- Participa en la dinámica de clase propuesta por el docente.
- Contribuye con ideas y perspectivas durante la discusión en grupo para fortalecer los aprendizajes adquiridos.
- Bajo la instrucción del docente, realizar programas básicos utilizando un Entorno de Desarrollo Integrado (IDE) como herramienta tecnológica en la POO.
- Practicar con la instancia de objetos y sus métodos de entrada y salida de datos siguiendo las indicaciones del docente.

El docente:

- Establece los criterios para la investigación sobre Programación Orientada a Objetos (POO), elementos del modelo de objetos, características y ventajas de la POO.
- Comunica los criterios a los estudiantes y los orienta en la búsqueda de información relevante.
- Explica los criterios de organización e integración de la información utilizando un organizador gráfico sobre Programación Orientada a Objetos.
- Establece la dinámica para compartir la información en clase, ya sea mediante lluvia de ideas, debate o juego de roles.
- Actúa como mediador del conocimiento durante la dinámica, facilitando la discusión y aclarando dudas.

INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

- Define los criterios para el manejo del Entorno de Desarrollo Integrado (IDE) como herramienta tecnológica en la POO.
- Proporciona instrucciones claras sobre el uso del IDE y sugerir prácticas recomendadas.

En el laboratorio:
- El docente presenta los criterios a utilizar para la realización de la Práctica N°1 "Instancia de objeto y métodos de entrada y salida de datos"

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas) y plataformas digitales.

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje con los conceptos básicos de la POO. • Cuadernillo de prácticas. • Presentaciones interactivas sobre los conceptos básicos de la POO.	• Organizador grafico: "Programación Orientada a Objetos"	**Instrumento de Evaluación:** • Rubrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • El organizador gráfico es visualmente atractivo y fácil de entender. • La información presentada es clara y fácil de seguir. • El organizador gráfico tiene una estructura lógica y coherente. • El texto y los elementos visuales son legibles y están bien distribuidos. • Se utilizan elementos creativos y efectivos para representar los conceptos de POO. Fondo: • Se incluyen los conceptos fundamentales de la POO. • Muestra claramente las relaciones entre los diferentes conceptos de POO y se presenta una jerarquía clara sobre cómo se relacionan entre sí. • Se proporcionan ejemplos para ilustrar cada concepto. • Se incluyen ejemplos de cómo se aplican los conceptos de POO en situaciones reales. • Los diferentes elementos del organizador gráfico están relacionados de manera coherente y consistente.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 1:	Introducción a la Programación Orientada a Objetos	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 1:	Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.		
Aprendizaje Esperado No 2:	Usa los principios de encapsulamiento en la programación orientada a objetos (POO), identificando los tipos de acceso a una clase, para asegurar la integridad y seguridad de los datos	**Tiempo estimado para obtener el Aprendizaje Esperado:**	9 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Modificadores de acceso (públicos, privados o protegidos). • Principios del encapsulamiento (Ocultamiento de la información, protección de la integridad de los datos, abstracción, facilidad de mantenimiento y de evolución del código).	• Compara las ventajas y las desventajas de los diferentes modificadores de acceso a la encapsulación de datos • Hace uso de principios del encapsulamiento para la implementación de clases y objetos ✓ Práctica N°2 "Principios de Encapsulamiento en la POO"	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Trabajo colaborativo

En el aula:

El estudiante:

- Analiza en parejas, los ejemplos proporcionados. Se les pide que discutan cómo afectan los diferentes modificadores de acceso a la encapsulación de datos en cada ejemplo y por qué es importante mantener la integridad y seguridad de los datos en un programa.
- Realiza una reflexión individual sobre los aprendizajes adquiridos durante la discusión en grupos. Se les pide que reflexionen sobre la importancia del pensamiento lógico, la capacidad de pensar críticamente y la comunicación asertiva en el desarrollo de software utilizando los principios de encapsulamiento.

El docente:

- Comienza la clase con una breve introducción sobre los principios de encapsulamiento en la POO, destacando la importancia de este concepto para asegurar la integridad y seguridad de los datos en un programa.
- Proporcionan ejemplos concretos de clases en un lenguaje orientado a objetos, donde se ilustran diferentes tipos de modificadores de acceso (públicos, privados o protegidos) para los atributos y métodos de las clases.
- Facilita la discusión en grupos, proporcionando orientación y resolviendo dudas según sea necesario. Se fomenta la participación activa de los estudiantes y se les motiva a expresar sus opiniones y puntos de vista.
- Estimula el pensamiento crítico haciendo preguntas desafiantes y alentando a los estudiantes a cuestionar y analizar los ejemplos proporcionados desde diferentes perspectivas.
- Promueve el trabajo colaborativo entre los estudiantes, animándolos a compartir ideas, colaborar en la resolución de problemas y aprender unos de otros.
- Fomenta la comunicación asertiva al proporcionar un entorno seguro y respetuoso donde los estudiantes se sientan cómodos expresando sus ideas y opiniones.

En el laboratorio:

- El docente presenta los criterios a utilizar para la realización de la Práctica N°2 "Principios de encapsulamiento en la POO"

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje. • Cuadernillo de prácticas. • Videos interactivos en la WEB • Bibliografía física o electrónica sugerida sobre POO • Hoja de trabajo para la reflexión individual.	• Hoja de trabajo con la reflexión individual.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • La reflexión está organizada de manera lógica y estructurada. • Los puntos presentados se relacionan entre sí de manera lógica. • La reflexión está escrita correctamente en términos de gramática, ortografía y puntuación. • La reflexión cubre los aspectos clave de la discusión y proporciona suficiente detalle para comprender los puntos de vista del estudiante. • Se incluyen todos los elementos requeridos, como introducción, desarrollo y conclusión. Fondo: • Muestra comprensión profunda de los temas discutidos durante el trabajo colaborativo. • Se centra en los puntos clave de la discusión y las lecciones aprendidas. • Presenta ideas originales o perspectivas creativas sobre el tema discutido. • Resume los puntos principales y extraer conclusiones relevantes. • Reflexiona sobre su propio proceso de aprendizaje y contribución al trabajo colaborativo.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 1:	Introducción a la Programación Orientada a Objetos	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 1:	Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.		
Aprendizaje Esperado No 3:	Estructura diagramas de clases básicos utilizando Lenguaje Unificado de Modelado (UML) para representar relaciones entre clases y objetos	**Tiempo estimado para obtener el Aprendizaje Esperado:**	6 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Definición del Lenguaje Unificado de Modelado (UML) • Diagramas de clases.	• Reconoce las características del Lenguaje Unificado de Modelado para la identificación de los componentes modulares del sistema. • Realiza diagramas de clases simples agrupando correctamente atributos y métodos de un objeto del mundo real para representar las relaciones entre estos dentro de su programa. ✓ Práctica N°3 "Modelado de clases con UML"	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Método del caso

El estudiante:
- Revisa material didáctico sobre UML y diagramas de clases antes de la clase.
- Explora ejemplos de diagramas de clases y casos de estudio del mundo real.
- Trabaja en parejas para crear diagramas de clases basados en casos de estudio proporcionados por el docente.
- Utilizan herramientas de modelado UML para diseñar las clases, identificar atributos y métodos, y definir las relaciones entre ellas.
- Implementa los programas basados en los diagramas de clases utilizando un entorno de desarrollo integrado.
- Cada pareja presenta su diagrama de clases y explica las decisiones de diseño tomadas.
- Se facilita una discusión en clase para analizar los diferentes enfoques y compartir experiencias.

El docente:
- Selecciona casos de estudio del mundo real y prepara material didáctico sobre UML y diagramas de clases.
- Proporciona una introducción clara y concisa sobre UML y los diagramas de clases.
- Fomenta la participación de los estudiantes en la discusión.
- Supervisa y orienta el trabajo de los estudiantes durante el desarrollo de los diagramas de clases y la implementación de los programas.
- Proporciona retroalimentación constructiva y resuelve dudas según sea necesario.
- Evalúa los diagramas de clases y los programas implementados según los criterios establecidos.
- Proporciona retroalimentación individualizada a cada pareja de estudiantes.

En el laboratorio:
- El docente presenta los criterios a utilizar para la realización de la Práctica N°3 "Modelado de clases con UML"

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje sobre UML y diagramas de clases. • Ejemplos de diagramas de clases. • Cuadernillo de prácticas. • Presentaciones interactivas sobre UML y diagramas de clases. • Bibliografía física o electrónica sugerida sobre POO. • Casos de estudio del mundo real para ejemplificar el uso de UML.	• Diagramas de clases.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • El diagrama utiliza un diseño visual que facilite la comprensión de la relación entre las clases. • Los elementos del diagrama (clases, atributos, métodos, relaciones) son legibles y están representados de manera clara. Se utilizan nombres significativos para las clases, atributos y métodos. • Las relaciones entre las clases están representadas correctamente y de acuerdo con las especificaciones del caso de estudio. Se utilizan los diferentes tipos de relaciones (asociación, agregación, composición, herencia) de manera adecuada. • Los atributos y métodos de cada clase están correctamente identificados y etiquetados en el diagrama. Se muestran de manera clara y precisa sus propiedades y funciones. • Se utilizan los símbolos y notaciones de UML de manera consistente y adecuada. Fondo: • El diagrama de clases refleja correctamente las entidades y relaciones del sistema según las especificaciones del caso de estudio. • Identifica correctamente las clases principales y las relaciones clave entre ellas. • Desarrolla el diagrama de manera coherente en todo el sistema. Se evitan redundancias y ambigüedades en la representación de las clases y sus relaciones. • Considera posibles cambios en los requisitos del sistema en el diseño del diagrama. • El estudiante demuestra habilidades de pensamiento crítico al tomar decisiones de diseño y resolver problemas encontrados durante la elaboración del diagrama.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 2:	Codificación	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 2:	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.		
Aprendizaje Esperado No 1:	Construye una nueva clase estableciendo la relación entre superclases y subclases para mejorar la reutilización del código, el mantenimiento, la extensibilidad, la flexibilidad y la organización de este.	**Tiempo estimado para obtener el Aprendizaje Esperado:**	6 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Herencia. • Relación de superclase y subclase.	• Identifica las características de una superclase y una subclase para comprender la relación entre estos. • Reconoce el concepto y el uso de la herramienta de herencia para ilustrar la manera eficiente de reutilizar el código • Realiza una nueva clase mediante el uso de herencia para reutilizar el código de forma adecuada dentro de su programa. ✓ Práctica N°4 "Implementación de herencia en POO"	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Método del caso

El estudiante:
- Investiga casos de estudio y ejemplos de aplicaciones de herencia en el mundo real.
- Analiza la estructura y la lógica de los programas que utilizan herencia para comprender su funcionamiento.
- Construye clases de superclase y subclase basadas en un caso de estudio o un problema dado.
- Codifica las clases utilizando herencia para reutilizar el código de la superclase en la subclase.
- Realiza pruebas exhaustivas para verificar el correcto funcionamiento de las clases implementadas.
- Depura y corrige errores encontrados durante las pruebas, si es necesario.

El docente:
- Presenta el concepto de herencia en Programación Orientada a Objetos (POO), explicando cómo se establecen las relaciones de superclase y subclase.
- Proporciona ejemplos simples y claros para ilustrar el concepto de herencia y su aplicación en la reutilización de código.
- Presenta casos de estudio o ejemplos del mundo real donde se aplique la herencia en la programación.
- Fomenta la participación de los estudiantes en la discusión sobre cómo la herencia puede simplificar el diseño y la implementación de sistemas de software.
- Guía a los estudiantes en la implementación práctica de clases utilizando herencia en un entorno de desarrollo integrado (IDE).
- Proporciona asistencia individualizada según sea necesario para resolver dudas y problemas durante la implementación.

En el laboratorio:
- El docente presenta los criterios a utilizar para la realización de la Práctica N°4 "Implementación de herencia en POO"

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje sobre herencia en POO. • Ejemplos de código que ilustren el concepto de herencia. • Cuadernillo de prácticas. • Presentaciones interactivas sobre herencia en POO. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado. • Casos de estudio del mundo real para ejemplificar el uso de herencia en POO.	• Programa con implementación de clases utilizando la herramienta de herencia.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • El código está bien estructurado y organizado. • Se utilizan nombres significativos para las clases, métodos y variables. • Se siguen las convenciones de codificación y estilo del lenguaje de programación utilizado. • El código está debidamente comentado para explicar su funcionamiento y propósito. • Se utilizan adecuadamente las relaciones de herencia entre las clases. • Se heredan correctamente los atributos y métodos de la superclase en la subclase. • Se implementan métodos adicionales en las subclases según sea necesario. • Se evita la redundancia de código mediante la reutilización efectiva de la funcionalidad de la superclase en las subclases. Fondo: • El programa implementado cumple con los requisitos funcionales establecidos. • El programa funciona correctamente y produce los resultados esperados. • Se utilizan estructuras de datos adecuadas y se evitan operaciones innecesarias. • El diseño del programa permite futuras extensiones y modificaciones de manera fácil y eficiente. • Se consideran posibles cambios en los requisitos del sistema al diseñar la estructura de clases y relaciones de herencia. • El estudiante demuestra habilidades creativas al implementar la herencia en el diseño del programa. • Se exploran soluciones innovadoras y se aplican enfoques creativos para resolver problemas específicos utilizando la herencia.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 2:	Codificación	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 2:	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.		
Aprendizaje Esperado No 2:	Simplifica la codificación de un programa orientado a objetos mediante el uso del polimorfismo adecuando su comportamiento a las necesidades del código.	**Tiempo estimado para obtener el Aprendizaje Esperado:**	9 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Creación de objetos. • Polimorfismo (Definición, características y ventajas).	• identifica las características, los beneficios y las desventajas de hacer uso de métodos polimórficos en el desarrollo de un código • Implementa métodos polimórficos a partir de una clase base para adaptarlos a diferentes procesos dentro de un programa. ✓ Práctica N°5 "Desarrollo colaborativo de código polimórfico".	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Trabajo colaborativo

El estudiante:
- Investiga sobre los conceptos de creación de objetos y polimorfismo en programación orientada a objetos. Se le proporcionan recursos didácticos como lecturas, videos o tutoriales.
- Analiza casos de estudio relacionados con la implementación de polimorfismo en programas de software. Se les asignan ejercicios prácticos para aplicar los conceptos aprendidos.
- Trabaja en parejas para implementar el polimorfismo en programas de software. Se le anima a explorar diferentes enfoques y técnicas para adaptar el comportamiento de los objetos.
- Participa en discusiones grupales moderadas por el docente sobre los conceptos de polimorfismo y su aplicación en la codificación de programas orientados a objetos. Se fomenta el intercambio de ideas y experiencias.

El docente:
- Selecciona y prepara recursos didácticos como lecturas, videos y ejemplos de código relacionados con el polimorfismo en POO.
- Guía al estudiante en la investigación previa sobre los conceptos de creación de objetos y polimorfismo, proporcionando orientación y respondiendo preguntas.
- Dirige discusiones grupales sobre los conceptos de polimorfismo, sus ventajas y desventajas, y su aplicación en diferentes escenarios de programación.
- Supervisa el progreso de la práctica del estudiante, brindando retroalimentación y apoyo individualizado según sea necesario.

En el laboratorio:
- El docente presenta los criterios a utilizar para la realización de la Práctica N°5 "Desarrollo colaborativo de código polimórfico".

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

PODER EJECUTIVO FEDERAL ESTADOS UNIDOS MEXICANOS MÉXICO CDMX
INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Presentaciones interactivas sobre conceptos de polimorfismo. • Apuntes de la unidad de aprendizaje sobre polimorfismo en POO. • Ejemplos de código que ilustren el concepto de polimorfismo. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado.	• Programa con implementación de polimorfismo	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • El código está bien estructurado y organizado de acuerdo con los estándares de desarrollo de software. • El código fuente es claro y legible. Se siguen convenciones de nomenclatura y estilo de codificación adecuadas. • Se incluyen comentarios claros y explicativos en el código para explicar el propósito y funcionamiento de las secciones importantes. • La presentación del código es profesional y bien organizada. Fondo: • Implementa correctamente los métodos polimórficos para adaptar diferentes procesos dentro del código. • Utiliza el polimorfismo de manera efectiva para mejorar la flexibilidad y la capacidad de adaptación del código. • El código cumple con los requisitos funcionales establecidos. • La implementación del polimorfismo mejora la funcionalidad y eficacia del código. • La implementación del polimorfismo no introduce cuellos de botella o problemas de rendimiento. • Los estudiantes demuestran una colaboración efectiva y un trabajo en equipo durante el desarrollo del código. • Se distribuyen equitativamente las tareas y se resuelven los conflictos de manera constructiva.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 3:	Aplicación práctica	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 3:	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.		
Aprendizaje Esperado No 1:	Establece excepciones en la programación, mediante la implementación de bloques try-except, con el propósito de prevenir la interrupción abrupta del programa.	**Tiempo estimado para obtener el Aprendizaje Esperado:**	9 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Definición de excepción. • Excepción try-except. • Importancia en las aplicaciones.	• Reconoce el concepto excepción try-except para identificar las situaciones en las que se puede ejecutar. • Resuelve errores comunes durante la ejecución del programa mediante el uso de excepciones para evitar la detención abrupta de esta. ✓ Práctica N°6 "Implementación de excepciones en POO".	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Método del caso

El estudiante:

- Realiza una investigación individual sobre el concepto de excepciones en programación y la importancia de los bloques try-except.
- En pareja, el estudiante analiza casos de programas con errores comunes durante la ejecución. Utiliza el pensamiento crítico para identificar la causa de los errores y cómo podrían haberse evitado con el uso de bloques try-except.
- Trabaja en pareja para resolver problemas prácticos relacionados con la implementación de bloques try-except. Utiliza el razonamiento lógico para diseñar soluciones efectivas y eficientes.
- Implementa bloques try-except en programas de ejemplo proporcionados por el docente. Utiliza el pensamiento crítico para anticipar posibles errores y determinar las mejores estrategias para manejarlos.

El docente:

- Introduce el concepto de excepciones y explicará la importancia de los bloques try-except en la programación. Fomenta el pensamiento crítico planteando preguntas reflexivas sobre la necesidad de manejar errores en los programas.
- Presenta casos de programas con errores comunes durante la ejecución y guía a los estudiantes para analizarlos críticamente, identificar los errores y proponer soluciones utilizando bloques try-except.
- Demuestra cómo implementar bloques try-except en programas de ejemplo y proporciona ejemplos prácticos para que los estudiantes practiquen. Fomenta el razonamiento lógico al explicar la lógica detrás de cada bloque try-except.
- Facilita discusiones en grupo para que los estudiantes compartan sus ideas y enfoques para resolver problemas relacionados con excepciones en programación. Guía la discusión hacia el pensamiento crítico, desafiando a los estudiantes a cuestionar y justificar sus decisiones.
- Proporciona retroalimentación individualizada a los estudiantes sobre sus soluciones y enfoques para manejar excepciones en sus programas. Evalúa el pensamiento crítico y el razonamiento lógico de los estudiantes a través de la calidad de sus respuestas y soluciones propuestas.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

En el laboratorio:

- El docente presenta los criterios a utilizar para la realización de la Práctica N°6 "Implementación de excepciones en POO".

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Material audiovisual, como videos explicativos o tutoriales en línea, que ilustren el uso de bloques try-except en la programación. • Apuntes de la unidad de aprendizaje sobre manejo de excepciones en POO. • Ejemplos de código con y sin bloques try-except para análisis y práctica. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado. • Casos de estudio y ejercicios prácticos que presenten situaciones reales donde el manejo de excepciones es crucial.	• Informe de análisis y soluciones propuestas • Presentación oral	**Instrumento de Evaluación:** • Rúbrica (Informe) • Guía de observación (Presentación oral) **Criterios de Evaluación:** *Informe:* Forma: • Entrega en tiempo y con limpieza. • Introducción clara que presenta los casos de estudio y su propósito. • Las secciones están organizadas de manera lógica y coherente. • Sigue las normas de formato establecidas, incluyendo márgenes, tamaño de letra y espaciado. • Se incluyen citas y referencias adecuadas según el estilo de escritura requerido. • Está libre de errores gramaticales y ortográficos. • Los casos de estudio y sus análisis están presentados de manera clara y comprensible. Fondo: • Se describen con precisión los errores comunes durante la ejecución de los programas y las soluciones propuestas utilizando bloques try-except. • Se identifican y explican claramente los errores comunes y las razones detrás de su ocurrencia. • Las soluciones propuestas utilizando bloques try-except son coherentes y viables. • Se justifica adecuadamente la elección de bloques try-except específicos para manejar cada tipo de excepción. • Se muestra la capacidad de analizar críticamente los casos de estudio presentados y proponer soluciones efectivas utilizando bloques try-except. *Presentación Oral:*

Programa Académico: **Técnico en Sistemas Digitales** **Unidad de Aprendizaje**: **Programación Orientada a Objetos**

		Forma: • Estructura clara con una introducción, desarrollo y conclusión bien definidos. • Los recursos visuales utilizados son efectivos para apoyar la presentación y mejorar la comprensión del tema. • El diseño de los recursos visuales es limpio, legible y estéticamente atractivo. • La expresión oral es clara, fluida y fácil de entender. • El volumen y la entonación son apropiados para mantener el interés del público. Fondo: • Se ofrece un análisis crítico y detallado de los casos de estudio presentados. • Se demuestra una comprensión profunda del manejo de excepciones en la programación durante la presentación. • Se explican claramente los conceptos y principios relacionados con el tema. • Se muestra la capacidad de analizar críticamente los casos de estudio presentados y proponer soluciones efectivas utilizando bloques try-except. • Las respuestas a las preguntas de la clase demuestran un entendimiento sólido del tema y la habilidad para comunicar ideas de manera clara y concisa.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Unidad Didáctica 3:	Aplicación práctica	**Nivel:**	Quinto
Propósito General:	Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		
Unidad de Competencia No 3:	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.		
Aprendizaje Esperado No 2:	Selecciona cada uno de los elementos de la programación orientada a objetos (POO), de manera creativa para diseñar aplicaciones que resuelvan problemas reales de manera efectiva.	**Tiempo estimado para obtener el Aprendizaje Esperado:**	12 horas

Contenidos de Aprendizaje

Conceptuales	Procedimentales	Actitudinales
• Modelado de aplicaciones mediante la POO. • Estilo del programador	• Estructura aplicaciones de software utilizando la POO para dar solución a problemas del mundo real. ✓ Práctica N°7 "Diseño de aplicación de software mediante POO".	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Gamificación

El estudiante:

- Se divide en equipos y recibe un conjunto de materiales que incluyen el tablero de juego, tarjetas de preguntas y fichas para cada equipo.
- Cada equipo elige un nombre y un avatar que los representará durante el juego.
- Cada equipo avanza por el tablero respondiendo preguntas y enfrentando desafíos relacionados con los conceptos de POO. Las preguntas desarrolladas y planteadas por cada docente deben abarcar los temas como encapsulamiento, herencia, polimorfismo, etc., y pueden presentarse en formato de opción múltiple, verdadero/falso, o de respuesta abierta. Los desafíos podrían incluir resolver problemas de programación simples o completar diagramas UML.
- Los equipos compiten entre sí para avanzar por el tablero y alcanzar la meta final, acumulando puntos por respuestas correctas y superación de desafíos.
- Fomenta la colaboración entre los miembros de su equipo, quienes pueden discutir las respuestas y estrategias para superar los desafíos.

El docente:

- Diseña el juego de mesa y prepara las preguntas y desafíos relacionados con los conceptos de POO estudiados a lo largo del curso y que servirán al alumno para retomarlos en la elaboración de su práctica No. 7, donde diseñarán una aplicación de software mediante POO.
- Establece las reglas y mecánicas del juego para garantizar una experiencia equilibrada y divertida.
- Facilita el desarrollo del juego, explicando las reglas y respondiendo preguntas de los estudiantes durante el transcurso del juego.
- Se asegura de que todos los equipos participen activamente y sigan las reglas del juego.
- Evalúa el desempeño de los estudiantes durante el juego, observando su capacidad para aplicar los conceptos de POO y trabajar en equipo.
- Proporciona retroalimentación individual y grupal al final del juego, destacando los puntos fuertes y áreas de mejora.
- Otorga premios y reconocimientos a los equipos con mejor desempeño, tanto en términos de puntos acumulados como de colaboración y trabajo en equipo.

En el laboratorio:

- El docente presenta los criterios a utilizar para la realización de la Práctica N°7 "Diseño de aplicación de software mediante POO".

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Ambientes de aprendizaje: Laboratorio de cómputo (Desarrollo de horas teóricas y horas prácticas).

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Instrumento y Criterios de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Material audiovisual proporcionado a lo largo del curso. • Apuntes de la unidad de aprendizaje vistos a lo largo del curso. • Ejemplos de código proporcionados a lo largo del curso. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado. • Casos de estudio y ejercicios prácticos realizados durante el curso. • Tablero de juego impreso. • Tarjetas de preguntas y desafíos. • Fichas y dados para avanzar por el tablero. • Presentación digital con las reglas del juego y los conceptos de POO.	• Productos creativos relacionados con los conceptos de POO que se aplicaron durante el juego (diagramas UML de las clases y relaciones utilizadas en el juego, o incluso pequeñas demostraciones de código que ilustren cómo se podrían implementar los conceptos en un entorno de programación real)	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • El producto tiene una estructura clara y organizada que facilite su comprensión. Se presentan los conceptos de manera lógica y coherente. • El producto es fácil de leer y entender. Se utilizan términos claros y precisos para comunicar los conceptos de Programación Orientada a Objetos (POO). • Los elementos visuales que se incluyen, como diagramas UML, son legibles y están bien diseñados. Ayudan a mejorar la comprensión de los conceptos. • El producto muestra originalidad en la forma en que se presentan los conceptos de POO. Se utilizan enfoques creativos para demostrar el entendimiento de los estudiantes. Fondo: • El producto demuestra un entendimiento preciso y completo de los conceptos de POO. Se aplican correctamente los principios de la Programación Orientada a Objetos. • El producto profundiza en los conceptos de POO de manera significativa. Se analizan las relaciones entre los diferentes elementos y se ofrecen percepciones relevantes. • El producto muestra cómo los conceptos de POO pueden aplicarse en situaciones prácticas. Se presentan ejemplos concretos y relevantes que ilustren la utilidad de los conceptos. • El producto muestra un enfoque original y creativo para abordar los conceptos de POO. Se proponen ideas innovadoras o soluciones creativas para problemas relacionados con la Programación Orientada a Objetos.

Programa Académico: Técnico en Sistemas Digitales

Unidad de Aprendizaje: Programación Orientada a Objetos

PRÁCTICAS

Nombre de la Práctica:	Instancia de objeto y métodos de entrada y salida de datos	**N° de la Práctica:**	1	**Tiempo:**	2 horas
Unidad de Competencia No. 1:	Implementa programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real.				
Aprendizajes Esperados Relacionados con la Práctica:	Identifica los conceptos básicos de la POO diferenciando entre clase y objeto para su uso correcto en la resolución de situaciones del mundo real.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Definición de Programación Orientada a Objetos (POO). • Características de la POO. • Concepto de clases y objetos.	• Describe los conceptos de clase y objeto, así como sus partes, identificando sus diferencias en código predefinido para determinar su uso correcto dentro de programa orientado a objetos.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Trabajo colaborativo

- El docente presenta una breve introducción sobre los conceptos básicos de la POO, como clases y objetos, y su aplicación en situaciones del mundo real y facilita la discusión inicial sobre los conceptos básicos de la POO y guía a los estudiantes para que identifiquen situaciones del mundo real que puedan modelarse con este enfoque.
- Los estudiantes se dividen en parejas y discuten sobre ejemplos de situaciones del mundo real que podrían modelarse utilizando la POO. Cada grupo identifica al menos una situación y cómo se podrían representar mediante clases y objetos.
- Cada pareja selecciona uno o varios de los ejemplos discutidos y comienza a desarrollar su modelo de objetos en un entorno de programación. Utilizan un lenguaje de programación orientado a objetos para implementar las clases y crear instancias de objetos según el modelo definido.
- Los estudiantes prueban sus implementaciones y depuran cualquier error encontrado, compartiendo sus desafíos y soluciones con otros equipos.
- Cada pareja presenta su modelo de objetos y explica cómo han aplicado los conceptos de clases y objetos para resolver la situación del mundo real. Los demás equipos brindan retroalimentación y sugerencias para mejorar.
- El docente está disponible para ayudar a los grupos durante el desarrollo de sus modelos de objetos, proporcionando orientación y resolviendo dudas que puedan surgir.
- El docente realiza evaluaciones formativas durante todo el proceso, observando la participación de los estudiantes, la comprensión de los conceptos y la calidad de las implementaciones.
- Después de las presentaciones de los equipos, el docente proporciona retroalimentación específica sobre los modelos de objetos presentados, destacando puntos fuertes y áreas de mejora.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas.	• Documentación de los modelos de objetos.	**Instrumento de Evaluación:** • Matriz de evaluación **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje: Programación Orientada a Objetos**

Recursos didácticos • Apuntes de la unidad de aprendizaje. • Cuadernillo de prácticas. • Videos interactivos en la WEB • Bibliografía física o electrónica sugerida sobre POO		• El trabajo está bien organizado y estructurado de manera lógica. • La presentación del trabajo es clara y legible. • Las ideas presentadas están conectadas de manera coherente y hay una relación clara entre ellas. Fondo: • Demuestra una comprensión clara y completa de los conceptos básicos de la POO. • Identifica qué clases y objetos son necesarios en un escenario dado. • Utiliza un razonamiento lógico y coherente para justificar sus decisiones de diseño de clases y objetos. • Propone soluciones originales y efectivas. • Comunica sus ideas de manera efectiva tanto de forma oral como escrita.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Principios de Encapsulamiento en la POO	**N° de la Práctica:**	2	**Tiempo:**	6 horas
Unidad de Competencia No. 1:	Implementa programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real.				
Aprendizajes Esperados Relacionados con la Práctica:	Explica los principios de encapsulamiento en la programación orientada a objetos (POO), identificando los tipos de acceso a una clase, para asegurar la integridad y seguridad de los datos.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Modificadores de acceso (públicos, privados o protegidos). • Principios del encapsulamiento (Ocultamiento de la información, protección de la integridad de los datos, abstracción, facilidad de mantenimiento y de evolución del código).	• Compara las ventajas y las desventajas de los diferentes modificadores de acceso a la encapsulación de datos • Hace uso de principios del encapsulamiento para la implementación de clases y objetos.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Aula invertida

Actividades del Estudiante:

- Revisa material didáctico en línea sobre los principios de encapsulamiento en la POO.
- Explora ejemplos de código fuente que ilustren la aplicación del encapsulamiento.
- Trabaja en grupos para analizar ejemplos de código que ejemplifiquen el encapsulamiento.
- Aplica los principios de encapsulamiento en la implementación de clases y objetos en un entorno de desarrollo integrado.
- Propone problemas prácticos que requieren el uso adecuado del encapsulamiento para su solución.
- Discute en grupo sobre los desafíos encontrados y reflexionan sobre la importancia del encapsulamiento en el diseño de software.

Actividades del Docente:

- Prepara material didáctico en línea y ejemplos de código para la práctica.
- Presenta los objetivos y expectativas de la práctica, así como una breve revisión de los conceptos clave.
- Supervisa y guía el trabajo de los estudiantes, respondiendo preguntas y proporcionando orientación según sea necesario.
- Observa el progreso de los estudiantes y ofrece retroalimentación durante la actividad.
- Facilita una discusión en clase para sintetizar los aprendizajes y reforzar la importancia del encapsulamiento en la POO.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Entornos de desarrollo integrado (IDE). • Herramientas de comunicación en línea para facilitar la colaboración entre estudiantes y docentes. • Equipo de cómputo con acceso a internet. • Proyector para mostrar ejemplos y diagramas.	• Ejemplos de código con demostración del correcto uso del encapsulamiento.	**Instrumento de Evaluación:** • Matriz de evaluación **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Recursos didácticos: • Material didáctico en línea sobre encapsulamiento en la POO. • Ejemplos de código fuente en lenguajes orientados a objetos. • Presentaciones interactivas sobre los principios de encapsulamiento. • Material impreso con ejercicios prácticos.		• El código está organizado de manera clara y estructurada, con una adecuada separación de clases, métodos y atributos. • Se utilizan nombres significativos para las variables, métodos y clases. • Se incluyen comentarios adecuados en el código para explicar su funcionamiento y el propósito de cada sección. • El código sigue las convenciones de codificación establecidas para el lenguaje de programación utilizado. Fondo: • Utiliza correctamente el encapsulamiento para ocultar el estado interno de los objetos y proteger los datos de acceso no autorizado. • Logra un adecuado nivel de abstracción en la implementación, donde los detalles internos de la clase están ocultos y solo se exponen las interfaces públicas necesarias. • Garantiza la integridad y seguridad de los datos al restringir el acceso a través de modificadores de acceso apropiados (públicos, privados o protegidos). • El encapsulamiento se aplica de manera coherente y consistente en todas las clases y métodos relevantes del código. • Logra una implementación eficiente, evitando redundancias y maximizando la reutilización de código.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Modelado de clases con UML	**N° de la Práctica:**	3	**Tiempo:**	3 horas
Unidad de Competencia No. 1:	Implementa programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real.				
Aprendizajes Esperados Relacionados con la Práctica:	Realiza diagramas de clases básicos empleando el Lenguaje Unificado de Modelado (UML) para representar relaciones entre clases y objetos.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Definición del Lenguaje Unificado de Modelado (UML) • Diagramas de clases.	• Reconoce las características del Lenguaje Unificado de Modelado para la identificación de los componentes modulares del sistema. • Realiza diagramas de clases simples agrupando correctamente atributos y métodos de un objeto del mundo real para representar las relaciones entre estos dentro de su programa.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Aula invertida

Actividades del Estudiante:

- Discute con su compañero sobre el problema a modelar y planificar la estrategia para abordarlo.
- Identifica las entidades principales, sus atributos y relaciones.
- Trabaja en equipo para crear el diagrama UML correspondiente al problema planteado.
- Utiliza herramientas de modelado UML para diseñar el diagrama de manera clara y estructurada.
- Revisa y analiza el diagrama UML creado, buscando posibles mejoras y corrigiendo errores.
- Colabora con su compañero para discutir y resolver cualquier discrepancia o problema identificado en el diagrama.
- Presenta el diagrama UML desarrollado ante el grupo, explicando su estructura y las decisiones de diseño tomadas.
- Escucha la retroalimentación de sus compañeros y del docente, y reflexiona sobre posibles mejoras.

Actividades del Docente:

- Prepara y comparte con los estudiantes materiales de lectura, videos o recursos en línea sobre UML.
- Asigna tareas específicas y establece un plazo para completarlas.
- Anima a los estudiantes a tomar notas y preparar preguntas para la sesión en clase.
- Prepara una presentación o material didáctico que detalle los conceptos y usos de los diferentes tipos de diagramas UML.
- Presenta ejemplos de casos de uso y diagramas de clases, y explica cómo se utilizan en el desarrollo de software.
- Observa las habilidades y conocimientos previos de los estudiantes y forma parejas que complementen sus fortalezas y debilidades.
- Fomenta la colaboración y el trabajo en equipo entre los estudiantes, donde puedan ayudarse mutuamente y aprender unos de otros.
- Explica detalladamente la tarea asignada, incluyendo los objetivos, los criterios de evaluación y los plazos de entrega.
- Proporciona ejemplos de problemas o escenarios que los estudiantes deben abordar utilizando diagramas UML, y muestra cómo se pueden modelar.
- Establece horarios de consulta o sesiones de tutoría donde los estudiantes puedan acudir para recibir ayuda y orientación.
- Responde a las preguntas de los estudiantes y brinda asistencia individualizada según sea necesario.
- Revisa los trabajos en progreso de los estudiantes y proporciona retroalimentación oportuna y constructiva.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

- Identifica áreas de mejora y proporciona sugerencias específicas para mejorar los diagramas UML de los estudiantes.
- Reconoce el progreso y los logros de los estudiantes, y fomenta un ambiente de aprendizaje positivo y de apoyo.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje sobre UML y diagramas de clases. • Ejemplos de diagramas de clases. • Cuadernillo de prácticas. • Presentaciones interactivas sobre UML y diagramas de clases. • Bibliografía física o electrónica sugerida sobre POO. • Casos de estudio del mundo real para ejemplificar el uso de UML.	• Programas implementados basados en los diagramas de clases desarrollados.	**Instrumento de Evaluación:** • Lista de verificación **Criterios de Evaluación:** Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • El diagrama está organizado de manera clara y estructurada. • Los elementos del diagrama son legibles y están representados de manera clara. • Las relaciones entre las clases están representadas correctamente. • Los atributos y métodos de cada clase están correctamente identificados. • El diagrama sigue las convenciones y estándares de UML. • El programa implementado cumple con los requisitos especificados. • El código está organizado en módulos claros y reutilizables. • El código es fácil de entender y mantener. • El programa utiliza las clases y métodos definidos en el diagrama de clase de manera adecuada. Fondo: • El diagrama refleja correctamente las entidades y relaciones del sistema. • El diagrama logra un equilibrio adecuado entre complejidad y abstracción. • El diseño del diagrama es coherente en todo el sistema. • El programa implementado cumple con todos los requisitos del caso de estudio. • El programa es eficiente y robusto en su funcionamiento. • El estudiante muestra habilidades de pensamiento crítico y creatividad en la implementación del programa.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Implementación de herencia en POO	**N° de la Práctica:**	4	**Tiempo:**	3 horas
Unidad de Competencia No. 2:	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.				
Aprendizajes Esperados Relacionados con la Práctica:	Construye una nueva clase estableciendo la relación entre superclases y subclases para mejorar la reutilización del código, el mantenimiento, la extensibilidad, la flexibilidad y la organización de este.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Herencia. • Relación de superclase y subclase.	• Identifica las características de una superclase y una subclase para comprender la relación entre estos. • Reconoce el concepto y el uso de la herramienta de herencia para ilustrar la manera eficiente de reutilizar el código • Realiza una nueva clase mediante el uso de herencia para reutilizar el código de forma adecuada dentro de su programa.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Trabajo colaborativo

Actividades del Estudiante:

- Trabaja en pareja para desarrollar y planificar la implementación de herencia en el código de solución a la problemática planteada, discutiendo ideas, estableciendo objetivos y dividiendo las tareas.
- Colabora estrechamente en la codificación de la solución propuesta, trabajando juntos en el diseño e implementación de las clases y relaciones de herencia.
- Realiza pruebas exhaustivas del código implementado en colaboración, identificando errores y depurando el código de manera conjunta.
- Colabora en la elaboración de la documentación del código, describiendo el proceso de diseño, implementación, así como cualquier otra información relevante.

Actividades del Docente:

- Capacita a los estudiantes en los principios y prácticas del trabajo en equipo, destacando la importancia de la comunicación efectiva, la colaboración y la responsabilidad compartida.
- Organiza a los estudiantes en parejas, teniendo en cuenta sus habilidades y experiencias previas, para fomentar una colaboración eficaz y complementaria.
- Establece objetivos claros para la solución de la problemática planteada y la implementación de herencia y define las tareas específicas que cada pareja deberá llevar a cabo.
- Brinda orientación y apoyo continuo a las parejas durante todo el proceso, respondiendo preguntas, resolviendo problemas y proporcionando retroalimentación constructiva.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Apuntes de la unidad de aprendizaje sobre herencia en POO. • Ejemplos de código que ilustren el concepto de herencia. • Cuadernillo de prácticas.	• Código fuente funcional • Documentación del código que resuelve la problemática planteada.	**Instrumento de Evaluación:** • Lista de verificación (Código fuente funcional) • Escala de valoración (Documentación del programa que resuelve la problemática planteada) **Criterios de Evaluación:** *Código fuente funcional:* Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • Está bien estructurado y organizado, siguiendo convenciones de codificación.

• Presentaciones interactivas sobre herencia en POO. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado. • Casos de estudio del mundo real para ejemplificar el uso de herencia en POO.		• Se utilizan nombres significativos para clases, métodos y variables. • La tabulación y la presentación del código facilitan su lectura y comprensión. • Está debidamente comentado para explicar su funcionamiento y propósito. Fondo: • Cumple con los requisitos funcionales establecidos. • Funciona correctamente y produce los resultados esperados. • Está optimizado en términos de eficiencia y rendimiento. • Se utilizan estructuras de datos adecuadas y se evitan operaciones innecesarias. • El diseño del programa permite futuras extensiones y modificaciones de manera fácil y eficiente. • Se consideran posibles cambios en los requisitos del sistema al diseñar la estructura de clases y relaciones de herencia. *Documentación del código que resuelve la problemática planteada:* Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • Se proporciona documentación clara sobre las clases, métodos y relaciones de herencia implementadas. Fondo: • Se establecen correctamente las relaciones de herencia entre las clases. • Los atributos y métodos de la superclase se heredan adecuadamente en las subclases. • Se implementan métodos adicionales en las subclases según sea necesario. • Se evita la redundancia de código mediante la reutilización efectiva de la funcionalidad de la superclase en las subclases. • El estudiante demuestra habilidades creativas al implementar la herencia en el diseño del programa. • Se exploran soluciones innovadoras y se aplican enfoques creativos para resolver problemas específicos utilizando la herencia.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Desarrollo colaborativo de código polimórfico	**N° de la Práctica:**	5	**Tiempo:**	6 horas
Unidad de Competencia No. 2:	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.				
Aprendizajes Esperados Relacionados con la Práctica:	Simplifica la codificación de un programa orientado a objetos mediante el uso del polimorfismo adecuando su comportamiento a las necesidades del código.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Creación de objetos. • Polimorfismo.	• Implementa métodos polimórficos a partir de una clase base para adaptarlos a diferentes procesos dentro de un programa.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Trabajo colaborativo

Actividades del Estudiante:

- Retoma los conceptos de polimorfismo en programación orientada a objetos y revisa ejemplos de su aplicación en diversos contextos.
- En pareja revisa un código base proporcionado por el docente que incluya ejemplos de polimorfismo.
- Discute y planifica cómo se puede mejorar y expandir el código base para demostrar diferentes formas de polimorfismo.
- Trabaja colaborativamente para implementar las mejoras planeadas en el código, utilizando técnicas de programación orientada a objetos y polimorfismo.
- Realiza pruebas exhaustivas del código para verificar su funcionamiento correcto y corregir cualquier error o problema encontrado.
- Cada pareja documenta el código desarrollado y prepara una presentación para compartir sus aprendizajes y experiencias con la clase.

Actividades del Docente:

- Proporciona una introducción al polimorfismo y explicará el propósito y los objetivos de la práctica.
- Asigna parejas de estudiantes, teniendo en cuenta su nivel de habilidad y experiencia en programación.
- Está disponible para responder preguntas, brindar orientación y apoyar a las parejas durante todo el proceso de desarrollo del código.
- Realiza una revisión intermedia del progreso de las parejas, proporcionando retroalimentación y sugerencias para mejorar el código.
- Evalúa el código desarrollado por cada pareja de estudiantes, según los criterios de evaluación establecidos.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Presentaciones interactivas sobre conceptos de polimorfismo.	• Código fuente funcional. • Documentación del programa que resuelve la problemática planteada.	**Instrumento de Evaluación:** • Lista de verificación (Código fuente funcional) • Escala de valoración (Documentación del programa que resuelve la problemática planteada) **Criterios de Evaluación:** *Código fuente funcional:* Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • El código está bien estructurado y organizado de manera lógica y coherente.

• Apuntes de la unidad de aprendizaje sobre polimorfismo en POO. • Ejemplos de código que ilustren el concepto de polimorfismo. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado.		• Se siguen convenciones de codificación adecuadas para la legibilidad y mantenibilidad del código. • Se utilizan nombres de variables, funciones y clases descriptivos y significativos. • Se incluyen comentarios pertinentes y claros dentro del código para facilitar su comprensión. Fondo: • El código demuestra una comprensión clara y correcta del concepto de polimorfismo. • Se implementan adecuadamente métodos polimórficos para adaptar el comportamiento de los objetos en diferentes contextos. • El código funciona correctamente y cumple con los requisitos establecidos. • Se logra el propósito deseado del polimorfismo, mejorando la flexibilidad y la capacidad de reutilización del código. • Los estudiantes demuestran una colaboración efectiva y un trabajo en equipo durante el desarrollo del código. • Se distribuyen equitativamente las tareas y se resuelven los conflictos de manera constructiva. *Documentación del programa que resuelve la problemática planteada:* Forma: • Entrega en tiempo y con limpieza. • Presenta buena ortografía y redacción. • La presentación de la solución propuesta es profesional y bien organizada. • La documentación de la propuesta planteada, incluyendo cualquier informe o explicación escrita, es completa y detallada. Fondo: • Contiene de manera clara y coherente el funcionamiento del programa y cómo se resuelve la problemática planteada. • Está organizada de manera lógica y fácil de seguir, con secciones claramente definidas para cada aspecto del programa. • Profundiza en los conceptos relevantes del programa, explicando detalladamente cómo se implementaron ciertas funcionalidades o decisiones de diseño. • Se incluyen ejemplos o casos de uso para ilustrar el funcionamiento del programa en situaciones prácticas. • Refleja el pensamiento crítico y la creatividad del estudiante en el diseño y desarrollo del programa.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Aplicación práctica	**N° de la Práctica:**	6	**Tiempo:**	6 horas
Unidad de Competencia No. 3:	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.				
Aprendizajes Esperados Relacionados con la Práctica:	Establece excepciones en la programación, mediante la implementación de bloques try-except, con el propósito de prevenir la interrupción abrupta del programa.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Definición. • Excepción try-except. • Importancia en las aplicaciones.	• Resuelve errores comunes durante la ejecución del programa mediante el uso de excepciones para evitar la detención abrupta de esta.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Design Thinking

Actividades del Estudiante:

- Retoma los temas abordados durante la clase para comprender los conceptos de definición de excepción, excepción try-except y su importancia en las aplicaciones.
- Participa en la lluvia de ideas para compartir experiencias y desafíos relacionados con el manejo de excepciones, y reflexionar sobre la aplicación de los conceptos aprendidos.
- Contribuye a la definición de un desafío específico relacionado con el manejo de excepciones en POO, aplicando el pensamiento lógico y crítico.
- Participa en actividades de generación de ideas y soluciones innovadoras durante la práctica, empleando los conceptos y procedimientos aprendidos.
- Trabaja de forma colaborativa en la propuesta de solución durante la práctica, aplicando el conocimiento adquirido sobre excepciones en la POO.
- Prueba y ajusta la propuesta de solución en función de la retroalimentación recibida durante la misma práctica, y reflexionar sobre los aprendizajes desarrollados.
- Participa en la evaluación rápida de la efectividad y viabilidad de la propuesta de solución durante la práctica, empleando el pensamiento lógico y crítico.
- Reflexiona sobre los resultados de las pruebas y colabora en la identificación de áreas de mejora durante la misma práctica, promoviendo la reflexión sobre los aprendizajes desarrollados.
- Prepara una presentación con la solución propuesta a la problemática planteada por el docente.

Actividades del Docente:

- Facilita una lluvia de ideas para compartir las experiencias y desafíos relacionados con el manejo de excepciones, incluyendo los conceptos de definición de excepción, excepción try-except y su importancia en las aplicaciones.
- Proporciona recursos y guía para retomar temas vistos en clase e investigar más durante la práctica.
- Guía a los estudiantes en la identificación de patrones y tendencias emergentes en los datos recopilados durante la práctica, relacionados con la aplicación de excepciones en la POO.
- Brinda orientación para la formulación de un desafío claro y enfocado durante la práctica, que incorpore los conceptos y procedimientos aprendidos.
- Genera el entorno para reflexionar en la importancia del pensamiento lógico y crítico durante la práctica para abordar los desafíos planteados.
- Ofrece retroalimentación constructiva sobre las ideas generadas por los estudiantes durante la práctica, fomentando la reflexión sobre los aprendizajes desarrollados.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet.	• Presentación oral del código propuesto para la solución planteada.	**Instrumento de Evaluación:** • Escala de valoración **Criterios de Evaluación:**

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

• Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades. • Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Presentaciones interactivas sobre conceptos de polimorfismo. • Apuntes de la unidad de aprendizaje sobre polimorfismo en POO. • Ejemplos de código que ilustren el concepto de polimorfismo. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado.		Forma: • Entrega en tiempo y con limpieza. • Tiene una estructura clara con una introducción, desarrollo y conclusión bien definidos. • Los recursos visuales utilizados son efectivos para apoyar la presentación y mejorar la comprensión del tema. • La expresión oral es clara, fluida y fácil de entender. • La demostración tiene una estructura clara y está bien organizada. • Los pasos para ejecutar el código y demostrar su funcionamiento están presentados de manera clara y concisa. Fondo: • Se demuestra una comprensión profunda del manejo de excepciones en la programación durante la presentación. • Se explican claramente los conceptos y principios relacionados con el tema. • El código desarrollado demuestra un manejo efectivo de excepciones utilizando bloques try-except. • El estudiante puede explicar claramente cómo se implementan los bloques try-except en el código y cómo contribuyen al manejo de excepciones. • Las respuestas a las preguntas de los estudiantes demuestran un entendimiento sólido del tema y la habilidad para comunicar ideas de manera clara y concisa. • El código es robusto y estable, evitando interrupciones abruptas del programa debido a excepciones no controladas.

INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

Nombre de la Práctica:	Diseño de aplicación de software mediante POO	**N° de la Práctica:**	7	**Tiempo:**	9 horas
Unidad de Competencia No. 3:	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.				
Aprendizajes Esperados Relacionados con la Práctica:	Selecciona cada uno de los elementos de la programación orientada a objetos (POO), de manera creativa para diseñar aplicaciones que resuelvan problemas reales de manera efectiva.				

Contenidos de Aprendizaje Relacionados con la Práctica

Conceptuales	Procedimentales	Actitudinales
• Modelado de aplicaciones mediante la POO. • Estilo del programador	• Estructura aplicaciones de software utilizando la POO para dar solución a problemas del mundo real.	• Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

Estrategia Didáctica y Ambiente de Aprendizaje

Estrategia Didáctica: Aprendizaje orientado a proyectos

Actividades del Estudiante:

- Elige una problemática del mundo real que desea abordar con su aplicación de software. Estos problemas pueden estar relacionados con áreas como gestión de inventarios, sistemas de reservas, gestión de clientes, entre otros y para los cuales debe retomar los términos repasados en clase para incluirlos en la solución propuesta.
- Realiza un análisis detallado del problema seleccionado y diseña soluciones utilizando el modelado de aplicaciones mediante la POO, además de utilizar herramientas como diagramas de clases UML para visualizar la estructura de sus aplicaciones.
- Trabaja en equipos para desarrollar la aplicación utilizando el estilo del programador y siguiendo las prácticas recomendadas de la POO. Implementan la estructura de su aplicación utilizando el lenguaje de programación orientados a objetos seleccionado al inicio del semestre por parte del docente.
- Realiza pruebas exhaustivas de su aplicación para garantizar su funcionalidad y solidez. Identifica y corrige errores utilizando técnicas de depuración.
- Documenta todo el proceso de desarrollo de su aplicación, incluyendo el análisis, diseño, implementación y pruebas. Esta documentación incluye manual de usuario, diagramas de flujo, UML, comentarios en el código, entre otros.
- Prepara una presentación para mostrar sus aplicaciones al resto de la clase. Durante la presentación, explica el problema que está abordando, la solución propuesta y cómo su aplicación resuelve efectivamente el problema.

Actividades del Docente:

- Guía a los estudiantes en la selección de un problema adecuado y proporciona ejemplos de aplicaciones exitosas desarrolladas utilizando la POO.
- Brinda asesoramiento continuo a los estudiantes durante todo el proceso de desarrollo de su aplicación. Ofrece retroalimentación constructiva sobre el análisis, diseño, implementación y pruebas.
- Proporciona recursos adicionales, como material de lectura, tutoriales en línea y herramientas de desarrollo, para apoyar a los estudiantes en el desarrollo de su aplicación.
- Evalúa el progreso de los estudiantes a lo largo del proyecto, teniendo en cuenta la calidad del análisis, diseño, implementación, pruebas y documentación de sus aplicaciones.

Ambiente de Aprendizaje: Laboratorio de cómputo

Herramientas Tecnológicas y Recursos Didácticos	Evidencia de Aprendizaje para la Evaluación Formativa	Criterios e Instrumentos de Evaluación
Herramientas Tecnológicas: • Entorno de desarrollo para el desarrollo de la POO (IDE) • Herramientas de modelado UML. • Equipo de cómputo con acceso a internet. • Plataforma de gestión del aprendizaje (LMS) para compartir recursos y actividades.	• Aplicación • Documentación de la aplicación que resuelve la problemática planteada.	**Instrumento de Evaluación:** • Rúbrica (Aplicación) • Lista de verificación (Documentación de la aplicación que resuelve la problemática planteada) **Criterios de Evaluación:**

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

• Proyector para mostrar ejemplos y diagramas. **Recursos didácticos** • Material audiovisual proporcionado a lo largo del curso. • Apuntes de la unidad de aprendizaje vistos a lo largo del curso. • Ejemplos de código proporcionados a lo largo del curso. • Cuadernillo de prácticas. • Bibliografía física o electrónica sugerida sobre POO. • Documentación sobre el lenguaje de programación utilizado. • Casos de estudio y ejercicios prácticos realizados durante el curso.		*Aplicación:* Forma: • Entrega en tiempo y con limpieza. • Tiene una interfaz de usuario limpia, organizada y estéticamente atractiva. • El código fuente está bien organizado y comentado, facilitando su comprensión y mantenimiento. Fondo: • Cumple con los requisitos especificados en el análisis y diseño previos. • Cumple con sus funciones previstas y resuelve efectivamente el problema planteado. • El diseño de la aplicación utiliza de manera efectiva los principios de la Programación Orientada a Objetos (POO), como encapsulamiento, herencia y polimorfismo. *Documentación de la aplicación que resuelve la problemática planteada:* Forma: • La documentación asociada, incluyendo análisis, diseño y manuales de usuario, está formateada de manera clara y profesional. • Sigue una estructura lógica y coherente, con secciones claramente definidas para el análisis, diseño, código fuente y manuales de usuario. • El análisis y diseño están presentados de manera clara y concisa, utilizando lenguaje técnico apropiado pero comprensible. Fondo: • Desarrolla una descripción detallada de cómo se implementaron los requisitos en la aplicación. • Proporciona evidencia clara de que la aplicación fue probada y funciona correctamente. • Contiene una explicación clara de la estructura de clases y relaciones entre objetos. • Demuestra que se han considerado diferentes enfoques y soluciones durante el proceso de desarrollo.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

PLAN DE EVALUACIÓN SUMATIVA

N°	Unidad de Competencia	Evidencia Integradora	Criterios e Instrumento de Evaluación	Porcentaje de Acreditación
1	Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.	Portafolio de evidencias: • Organizador grafico: "Programación Orientada a Objetos" • Documentación de los modelos de objetos. • Ejemplos de código con demostración del correcto uso del encapsulamiento. • Programas implementados basados en los diagramas de clases desarrollados.	**Instrumento de Evaluación:** • Lista de verificación **Criterios de Evaluación:** Forma: • Está organizado de manera clara y coherente y presenta una estructura lógica que facilita la navegación. • Se utiliza una presentación visual adecuada que mejora la comprensión de los trabajos presentados. • Se utilizan títulos, subtítulos y párrafos para organizar la información de manera clara. • Se incluyen todos los trabajos solicitados que ya han sido evaluados y están claramente etiquetados y organizados. • Se proporcionan referencias o citas adecuadas cuando sea necesario. Fondo: • Se demuestra una comprensión clara y precisa de los conceptos de Programación Orientada a Objetos (POO) y encapsulamiento en los trabajos presentados. • Se muestra la aplicación práctica de los conceptos de POO y encapsulamiento en los ejemplos de código y programas implementados. • Los trabajos presentados son coherentes entre sí y muestran consistencia en la aplicación de los conceptos de POO y encapsulamiento. • La documentación de los modelos de objetos y los ejemplos de código demuestran un nivel de calidad adecuado. • Se muestra originalidad y creatividad en la aplicación de los conceptos de POO y	20%

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

			encapsulamiento en los programas implementados y otros trabajos presentados.	
2	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.	Documentación del código que resuelve la problemática planteada.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • El código está bien estructurado y organizado de manera clara y coherente. • El código utiliza nombres significativos para clases, métodos y variables. • El código está debidamente comentado para explicar su funcionamiento y propósito. • Se proporciona documentación clara sobre las clases, métodos y relaciones de herencia implementadas. • Se utiliza una tabulación consistente y se mantienen las reglas de estilo de código. • El programa implementado cumple con los requisitos funcionales establecidos. Ofrece una solución efectiva al problema planteado. • Se evitan interrupciones abruptas del programa debido a excepciones no controladas. Fondo: • Se utiliza la herencia y el polimorfismo de manera efectiva para estructurar y extender la funcionalidad del programa. • Se aprovechan las ventajas de la reutilización de código y la flexibilidad en el diseño. • El diseño del programa refleja de manera coherente los conceptos de herencia y polimorfismo. • Las clases relacionadas a través de herencia están integradas de manera adecuada en el diseño general del programa. • Presenta y comunica claramente su solución utilizando un lenguaje claro y conciso. • Justifica sus decisiones de diseño y explicar el funcionamiento del programa de manera efectiva.	30%

			• Muestra originalidad y creatividad en el diseño y la implementación del programa. • Se demuestra la capacidad del alumno para pensar de manera lógica y crítica en la adaptación del programa a situaciones variables.	
3	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.	Documentación de la aplicación que resuelve la problemática planteada.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • El diseño de la aplicación está bien estructurado y organizado de manera lógica y coherente. • El diseño de la aplicación es fácil de entender y seguir. • Se utilizan nombres significativos para clases, métodos y variables. • Se proporciona una documentación clara y completa sobre la aplicación, incluyendo análisis, diseño y manuales de usuario. • El diseño de la aplicación sigue convenciones de diseño adecuadas para la legibilidad y mantenibilidad del código. • Se mantienen las reglas de estilo de diseño. • La aplicación diseñada cumple con los requisitos funcionales establecidos. • Ofrece una solución efectiva y práctica al problema planteado. Fondo: • El diseño de la aplicación demuestra un entendimiento claro y correcto de los principios de la programación orientada a objetos (POO). • Se utilizan de manera efectiva conceptos como encapsulamiento, herencia y polimorfismo en el diseño de la aplicación. • La aplicación diseñada resuelve de manera efectiva un problema del mundo real. • Se demuestra la capacidad del estudiante para identificar y abordar problemas prácticos utilizando la POO.	50%

			• La aplicación tiene una interfaz de usuario intuitiva y fácil de usar. • El manual de usuario proporciona instrucciones claras y concisas para el uso adecuado de la aplicación. • El diseño de la aplicación muestra originalidad y creatividad en su enfoque y funcionalidad. • Explora soluciones innovadoras y se aplican enfoques creativos para resolver el problema planteado. • El diseño de la aplicación permite futuras extensiones y modificaciones de manera fácil y eficiente. • Se consideran posibles cambios en los requisitos del sistema al diseñar la estructura de clases y relaciones de POO. • Realiza un análisis crítico y reflexiona sobre los aprendizajes desarrollados durante el diseño de la aplicación. • Justifica sus decisiones de diseño y explicar el funcionamiento de la aplicación de manera efectiva.	
Propósito de la Unidad de Aprendizaje		**Evidencia Integradora**	**Criterios e Instrumento de Evaluación**	**Porcentaje de Acreditación**
Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para fomentar la resolución autónoma y creativa de problemas de programación en situaciones del mundo real.		Aplicación funcional que aborda un problema real utilizando los principios de la POO.	**Instrumento de Evaluación:** • Rúbrica **Criterios de Evaluación:** Forma: • El código de la aplicación está bien estructurado y organizado de manera lógica y coherente. • El código es fácil de entender y seguir. Se utilizan nombres significativos para clases, métodos y variables. • Se proporciona una documentación clara y completa sobre la aplicación, incluyendo análisis, diseño y manuales de usuario. • El código sigue las convenciones de codificación y las mejores prácticas de programación.	**100%**

INSTITUTO POLITÉCNICO NACIONAL
Dirección de
Educación Media Superior

		• La interfaz de usuario de la aplicación es intuitiva y fácil de usar. Se sigue un diseño coherente y estéticamente agradable. • La aplicación ha sido probada exhaustivamente para garantizar que funcione correctamente y cumpla con los requisitos especificados. • Los diagramas de clase están correctamente estructurados y representan de manera precisa las relaciones entre las clases y objetos de la aplicación. Se utilizan etiquetas y convenciones adecuadas para mejorar la comprensión. Fondo: • El diseño de la aplicación demuestra un entendimiento claro y correcto de los principios de la programación orientada a objetos (POO). Se utilizan adecuadamente conceptos como encapsulamiento, herencia y polimorfismo. • La aplicación resuelve de manera efectiva el problema planteado. • Aborda las necesidades y requerimientos de los usuarios de manera adecuada. • La aplicación muestra originalidad y creatividad en su diseño y funcionalidad. Se exploran soluciones innovadoras para resolver el problema. • El diseño de la aplicación permite futuras extensiones y modificaciones de manera fácil y eficiente. Es escalable y adaptable a diferentes situaciones. • La aplicación funciona correctamente y produce resultados rápidos y eficientes. • Se utilizan adecuadamente las estructuras de datos y algoritmos para optimizar el rendimiento. • El estudiante realiza un análisis crítico y reflexiona sobre los aprendizajes desarrollados durante el desarrollo de la aplicación. • Justifica sus decisiones de diseño y explicar el funcionamiento de la aplicación de manera efectiva.	

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

PROGRAMA SINTÉTICO

PROPÓSITO DE LA UNIDAD DE APRENDIZAJE			
Desarrolla aplicaciones de software de manera efectiva, empleando la programación orientada a objetos (POO) para satisfacer las necesidades de los usuarios de forma eficiente y con responsabilidad social.			
N°	**UNIDAD DE COMPETENCIA**	**APRENDIZAJES ESPERADOS**	**CONTENIDOS DE APRENDIZAJE/SABERES**
1	Integra programas básicos utilizando la programación orientada a objetos para abordar problemas de la vida real a partir de un pensamiento lógico y crítico.	Compara las características de clase y objeto identificando los conceptos básicos de la POO para su uso correcto en la resolución de situaciones del mundo real.	**Conceptuales:** • Definición de Programación Orientada a Objetos (POO). • Características de la POO. • Concepto de clases y objetos. **Procedimentales:** • Reconoce los conceptos y las partes de clase y objeto, identificando sus diferencias en código predefinido. • Relaciona las características de clase y objeto para determinar su uso correcto dentro de programa orientado a objetos. • Práctica N°1 "Instancia de objeto y métodos de entrada y salida de datos" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente. • Realiza una comunicación asertiva.
		Usa los principios de encapsulamiento en la programación orientada a objetos (POO), identificando los tipos de acceso a una clase, para asegurar la integridad y seguridad de los datos	**Conceptuales:** • Modificadores de acceso (públicos, privados o protegidos). • Principios del encapsulamiento (Ocultamiento de la información, protección de la integridad de los datos, abstracción, facilidad de mantenimiento y de evolución del código). **Procedimentales:** • Compara las ventajas y las desventajas de los diferentes modificadores de acceso a la encapsulación de datos • Hace uso de principios del encapsulamiento para la implementación de clases y objetos correcto dentro de programa orientado a objetos. • Práctica N°2 "Principios de Encapsulamiento en la POO" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados. • Trabaja colaborativamente.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

			• Realiza una comunicación asertiva.
		Estructura diagramas de clases básicos utilizando Lenguaje Unificado de Modelado (UML) para representar relaciones entre clases y objetos	**Conceptuales:** • Definición del Lenguaje Unificado de Modelado (UML) • Diagramas de clases. **Procedimentales:** • Reconoce las características del Lenguaje Unificado de Modelado para la identificación de los componentes modulares del sistema. • Realiza diagramas de clases simples agrupando correctamente atributos y métodos de un objeto del mundo real para representar las relaciones entre estos dentro de su programa. • Práctica N°3 "Modelado de clases con UML" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.
2	Estructura programas sencillos empleando la herencia y polimorfismo para ofrecer solución a problemas reales bajo la programación orientada a objetos mediante el empleo de una comunicación asertiva.	Construye una nueva clase estableciendo la relación entre superclases y subclases para mejorar la reutilización del código, el mantenimiento, la extensibilidad, la flexibilidad y la organización de este.	**Conceptuales:** • Herencia. • Relación de superclase y subclase. **Procedimentales:** • Identifica las características de una superclase y una subclase para comprender la relación entre estos. • Reconoce el concepto y el uso de la herramienta de herencia para ilustrar la manera eficiente de reutilizar el código • Realiza una nueva clase mediante el uso de herencia para reutilizar el código de forma adecuada dentro de su programa. • Práctica N°4 "Implementación de herencia en POO" **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados
		Simplifica la codificación de un programa orientado a objetos mediante el uso del polimorfismo adecuando su comportamiento a las necesidades del código.	**Conceptuales:** • Creación de objetos. • Polimorfismo (Definición, características y ventajas). **Procedimentales:** • identifica las características, los beneficios y las desventajas de hacer uso de métodos polimórficos en el desarrollo de un código. • Implementa métodos polimórficos a partir de una clase base para adaptarlos a diferentes procesos dentro de un programa.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

			• Práctica N°5 "Desarrollo colaborativo de código polimórfico". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.
3	Diseña aplicaciones de software básicas utilizando los principios de la programación orientada a objetos (POO), para resolver problemas del mundo real a través de proyectos prácticos, apoyándose del trabajo colaborativo.	Establece excepciones en la programación, mediante la implementación de bloques try-except, con el propósito de prevenir la interrupción abrupta del programa.	**Conceptuales:** • Definición de excepción • Excepción try-except. • Importancia en las aplicaciones. **Procedimentales:** • Reconoce el concepto excepción try-except para identificar las situaciones en las que se puede ejecutar. • Resuelve errores comunes durante la ejecución del programa mediante el uso de excepciones para evitar la detención abrupta de esta. • Práctica N°6 "Implementación de excepciones en POO". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados
		Selecciona cada uno de los elementos de la programación orientada a objetos (POO), de manera creativa para diseñar aplicaciones que resuelvan problemas reales de manera efectiva.	**Conceptuales:** • Modelado de aplicaciones mediante la POO. • Estilo del programador **Procedimentales:** • Estructura aplicaciones de software utilizando la POO para dar solución a problemas del mundo real. • Práctica N°7 "Diseño de aplicación de software mediante POO". **Actitudinales:** • Emplea el pensamiento lógico. • Piensa de manera crítica. • Reflexiona sobre los aprendizajes desarrollados • Trabaja colaborativamente • Realiza una comunicación asertiva.

Programa Académico: Técnico en Sistemas Digitales **Unidad de Aprendizaje**: Programación Orientada a Objetos

BIBLIOGRAFÍA BÁSICA Y COMPLEMENTARIA

Número y Nombre de la Unidad Didáctica	FORMATO APA	CLASIFICACIÓN	
		Básico	Consulta
UNIDAD 1: INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS	Alarcón, P. L. G. (2015). UML 2: Aprenda UML 2.5. Ediciones ENI.	X	
	González Duque, R. (2012). POO: Fundamentos y Práctica. Alfaomega Grupo Editor.	X	
	Alquati, R. (2019). Python para todos: Explorando la Programación. Marcombo.	X	
	Gutiérrez, D. (2018). Python: De principiante a experto. CreateSpace Independent Publishing Platform.	X	
	Beazley, D. (2010). Python esencial. Anaya Multimedia.		X
	Joyanes Aguilar, L. (2016). Fundamentos de Programación: Algoritmos, estructuras de datos y objetos. McGraw-Hill.		X
	López, M. M., García, M. V. B., & Rodríguez, J. A. T. (2016). Análisis y diseño de sistemas de información con UML. RA-MA Editorial.		X
	Rumbaugh, J., Jacobson, I., & Booch, G. (1999). El lenguaje unificado de modelado. Addison-Wesley.	X	
UNIDAD 2: CODIFICACIÓN	Deitel, P., & Deitel, H. (2011). Java: cómo programar. Pearson Educación.	X	
	Galdames, F. R. (2015). Aprende Java con ejercicios resueltos. Ediciones ENI.	X	
	Martelli, A., Ravenscroft, A., & Ascher, D. (2010). Python: lenguaje de programación. Anaya Multimedia.	X	
	Martín-González, M. (2019). Python para todos. Marcombo.	X	
	Pilgrim, M. (2004). Aprenda a pensar como un programador con Python. O'Reilly Media.		X
	Lutz, M. (2013). Learning Python: Powerful Object-Oriented Programming. O'Reilly Media.		X
	Kendall, K. E., & Kendall, J. E. (2011). Análisis y Diseño de Sistemas. Pearson Educación.		X
	Sánchez, J. L. V., & Sánchez, M. A. V. (2017). Programación orientada a objetos en java: fundamentos y aplicaciones. Alfaomega.	X	
	Sierra, K., & Bates, B. (2005). Head First Java. O'Reilly Media.		X

Programa Académico: **Técnico en Sistemas Digitales** **Unidad de Aprendizaje**: **Programación Orientada a Objetos**

	Vega Sánchez, H. (2007). Programación Orientada a Objetos: Conceptos, Métodos y Aplicaciones. Pearson Educación.		X
UNIDAD 3: APLICACIÓN PRÁCTICA	McConnell, S. (2014). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.		X
	Meyer, B. (2000). Diseño de Software: Orientado a Objetos. Pearson Educación.	X	
	Beazley, D., & Jones, B. K. (2000). Programación en Python: Guía integral del lenguaje. Anaya Multimedia.	X	
	Pilgrim, M. (2014). Dive into Python 3. Apress.	X	
	Beazley, D., & Jones, B. K. (2000). Programación en Python: Guía integral del lenguaje. Anaya Multimedia.		X
	Lutz, M. (2016). Python Pocket Reference. O'Reilly Media.		X
	Pressman, R. S. (2010). Ingeniería del Software: Un enfoque práctico. McGraw-Hill.	X	
	Sommerville, I. (2005). Ingeniería de Software. Pearson Educación.		X