Formato para la elaboración y actualización del Microcurrículo.

Dependencia: Desarrollo del Pensamiento Abstracto y Programación		**Facultad o Departamento: Ingeniería**
Programa:		Ingeniería en Software/Tecnología en Sistemas por Ciclos Propedéuticos
1. Identidad[1] del microcurrículo, asignatura o módulo.		Paradigma Orientado a Objetos
Nivel en el plan de estudios:		II SEMESTRE
Código en el plan de estudios:		PREISESP17005
CINE -código, según la Clasificación Internacional Normalizada de la Educación-:		0613
Número de créditos académicos:		3
Número de Horas Intensidad Trabajo Presencial:		64
Número de Horas Intensidad Trabajo Independiente:		80
Modalidad:		Presencial
Condicionantes:	Prerrequisitos	Introducción al desarrollo de Software
	Correquisitos:	Ninguno
	Homologable:	Ninguno
	Validable:	Ninguno
	Vacacional:	Ninguno

2. **Alineamiento constructivo del mesocurrículo al cual pertenece el microcurrículo, asignatura o módulo.**

Perfiles de formación con los cuales se compromete el mesocurrículo.	**Competencias del mesocurrículo.**	**Resultados de aprendizaje del mesocurrículo.**

[1] Los diferentes numerales del microcurrículo –asignatura o módulo-, se encuentran en coherencia curricular con lo argumentado en el mesocurrículo –área, componente, campo o línea-, al cual pertenecen en referencia al diseño curricular del programa.

Revisado por:	**Aprobado por:**
Cargo: Coordinación del Observatorio Pedagógico	Cargo: Vicerrectoría Académica
Firma:	Firma:

El estudiante estará en la capacidad de participar activamente en los procesos de construcción de software siendo un líder fundamental en las etapas de definición, análisis y diseño basado en los requerimientos del usuario.	Aplicar métodos y herramientas para el diseño y desarrollo de sistemas basados en computadores.	Aplica las etapas del desarrollo de software en el proceso de construcción de los componentes, las soluciones informáticas y los sistemas de información.

3. Alineamiento constructivo del microcurrículo: procesos de aprendizaje.
a) Comprender y aplicar los diagramas UML en el modelado de clases y relaciones
b) Introducción a la Programación Orientada a Objetos (POO)
c) Desarrollar la capacidad para modelar soluciones de software utilizando la metodología ABP y ABPy en UML
d) Evaluar y presentar proyectos de modelado UML con patrones de diseño y arquitecturas

3. Justificación del microcurrículo (máximo 250 palabras).

La Programación Orientada a Objetos (POO) es una disciplina fundamental para los estudiantes de ingeniería de software, ya que promueve un enfoque modular y reutilizable en el desarrollo de sistemas informáticos. Este microcurrículo se enfoca en desarrollar una comprensión profunda de los conceptos clave de la POO, tales como clases, objetos, atributos, métodos, y patrones de diseño, sin la necesidad de involucrarse directamente en la programación, permitiendo que los estudiantes se concentren en la estructura y el diseño del software desde una perspectiva más conceptual.

El uso de UML (Unified Modeling Language) como herramienta principal para el modelado de clases, objetos y relaciones permite a los estudiantes visualizar de manera clara y comprensible cómo se estructuran y relacionan los componentes de un sistema. A través de diagramas como el de clases, objetos, secuencia, y casos de uso, los estudiantes pueden representar gráficamente el diseño de sistemas complejos, lo cual les ayudará a desarrollar un pensamiento lógico y crítico para la solución de problemas.

Además, la integración de metodologías como el Aprendizaje Basado en Problemas (ABP) y el Aprendizaje Basado en Proyectos de Software (ABPy) en este curso fomenta la capacidad de los estudiantes para aplicar los conceptos aprendidos en situaciones reales y problemas complejos. Estas metodologías los animan a colaborar en equipo, abordar desafíos prácticos y presentar soluciones efectivas mediante diagramas UML, lo que refuerza el aprendizaje activo y la integración de conocimientos.

El enfoque en patrones de diseño a lo largo del curso es crucial, ya que estos patrones proporcionan soluciones probadas a problemas comunes en la ingeniería de software. Al estudiar patrones creacionales, estructurales y de comportamiento, los estudiantes no solo aprenden a crear software más eficiente y escalable, sino que también adquieren herramientas para abordar problemas de diseño de una manera más estructurada y profesional.

En resumen, este microcurrículo se fundamenta en proporcionar a los estudiantes las herramientas conceptuales necesarias para entender y aplicar la POO en el diseño de software mediante UML, lo cual es esencial para construir aplicaciones robustas, modulares y fáciles de mantener. Con el enfoque en la reutilización de código, modularidad y patrones de diseño, los estudiantes estarán mejor preparados para afrontar los retos del desarrollo de software en un entorno profesional.

4. Unidades temáticas de enseñanza, formación y aprendizaje.

Unidad 01:		**Denominación global de la unidad temática:** Introducción a POO		
Proceso-resultado de aprendizaje.		**Proceso-resultado de aprendizaje.** Comprender y aplicar los diagramas UML en el modelado de clases y relaciones		
Ítems	**Contenido de enseñanza**	**Trabajo Presencial o metodologías de enseñanza.**	**Trabajo presencial en equipos o independiente.**	**Trabajo independiente.**
Semana 1	Introducción a UML y sus diagramas. Diferencia entre diagramas estructurales y de comportamiento. Uso de herramientas como (StarUML, plantUML entre otras).	Clase magistral. Explicación de UML y su importancia.	Trabajo en equipos: análisis de diagramas UML de software real.	Lectura de material y visualización de videos sobre UML.

Semana 2	Diagramas de clases y objetos: modelado de atributos, métodos y relaciones.	Desarrollo de diagramas UML en clase.	Trabajo en equipos: modelado de una estructura básica en UML.	Desarrollo de diagramas UML individuales.
Semana 3	Relaciones entre clases: asociación, agregación, composición y herencia en UML.	Clase magistral y resolución de ejercicios.	Desarrollo de diagramas en equipos, análisis de casos.	Ejercicios de modelado UML en casa.
Semana 4	Diagramas: paquetes, casos de uso y secuencia. Modularización de software.	Explicación teórica y práctica en clase.	Trabajo grupal en modelado UML de un sistema.	Lectura de material sobre diagramas de paquetes y su importancia.
Evidencia de conocimiento, procedimiento o producto observable, donde los estudiantes comunican o demuestran el logro del aprendizaje sugerido.		**Evidencia de producto:** Aplicación de los conocimientos adquiridos en las clases en el desarrollo de algoritmos, con el fin de evaluar el entendimiento del grupo **Evidencia de conocimiento y procedimiento:** Desarrollo de Algoritmos.		
Recursos, herramientas, materiales, portadores de texto y ayudas educativas requeridas en la unidad.		Diapositivas del docente Plataforma Moodle Videos y páginas de la web. Sala de Sistemas Consultas en Internet. Computadores con acceso a Internet. Videobeam o Televisor. Herramientas ofimáticas. Herramienta de desarrollo para Java		
Unidad 02:		**Denominación global de la unidad temática:** Métodos y clases		
Proceso-resultado de aprendizaje.		**Proceso-resultado de aprendizaje.** Introducción a la Programación Orientada a Objetos (POO		

Ítems	Contenido de enseñanza	Trabajo Presencial o metodologías de enseñanza.	Trabajo presencial en equipos o independiente.	Trabajo independiente.
Semana 5	Introducción a la Programación Orientada a Objetos (POO): clases, objetos, atributos y métodos.	Clase magistral. Análisis de conceptos de POO sin programación.	Trabajo en equipos: modelado de clases y objetos en UML.	Resolución de ejercicios de modelado UML.
Semana 6	Diagramas de actividad y de estados: modelado de comportamiento.	Explicación con ejemplos de modelado de procesos.	Trabajo en equipos para diseñar diagramas de actividad.	Estudio de diagramas UML y aplicación en casos prácticos.
Semana 7	Abstracción y reutilización de clases: modelado de clases reutilizables en UML.	Clase magistral y ejercicios guiados.	Trabajo grupal en la creación de clases reutilizables.	Ejercicios de modelado UML sobre abstracción.
Semana 8	Examen Parcial: Modelado UML en un caso de estudio.	Examen teórico y práctico en clase.	Evaluación en equipos e individual.	Revisión de conceptos antes del examen.
Evidencia del logro del aprendizaje.		**Evidencia de producto:** Programa en Java aplicando los conceptos de básicos de POO para la solución a problemas propuestos **Evidencia de conocimiento y procedimiento:** Desarrollo de Algoritmos. Desarrollo de programas Orientados a Objetos		
Recursos o herramientas.		Diapositivas del docente Plataforma Moodle Videos y páginas de la web. Sala de Sistemas Consultas en Internet. Computadores con acceso a Internet.		

		Videobeam o Televisor. Herramientas ofimáticas. Herramienta de desarrollo para Java		
Unidad 03:		**Denominación global de la unidad temática:** la metodología ABP y ABPy		
Proceso-resultado de aprendizaje.		**Proceso-resultado de aprendizaje.** Desarrollar la capacidad para modelar soluciones de software utilizando la metodología Aprendizaje Basado en Problemas (ABP) y Aprendizaje Basado en Proyectos (ABPy) en UML		
Ítems	**Contenido de enseñanza**	**Trabajo Presencial o metodologías de enseñanza.**	**Trabajo presencial en equipos o independiente.**	**Trabajo independiente.**
Semana 9	Introducción a Aprendizaje Basado en Problemas (ABP) y Aprendizaje Basado en Proyectos de Software (ABPy).	Clase magistral y presentación de la metodología.	Trabajo en equipos: identificación de problemas de software.	Investigación sobre ABP y ABPy.
Semana 10	Aplicación de ABP y ABPy en modelado UML. Diseño basado en problemas reales.	Desarrollo de un proyecto con ABP en clase.	Trabajo colaborativo en el modelado de soluciones.	Lectura sobre estrategias ABP y ABPy en software.
Semana 11	Diseño de arquitecturas de software con UML.	Explicación de modularización con diagramas de paquetes.	Desarrollo en equipos de una arquitectura basada en UML.	Análisis y aplicación en proyectos individuales.
Semana 12	Simulación de problemas en UML. Evaluación de soluciones a	Trabajo práctico en clase con modelado de problemas.	Presentación de soluciones en equipos.	Aplicación de UML en problemas propuesto

	través de diagramas.			
Evidencia del logro del aprendizaje.		**Evidencia de producto:** Programa - Proyecto en Java aplicando los cuatro pilares de la POO en la solución a problemas propuestos **Evidencia de conocimiento y procedimiento:** Desarrollo de Algoritmos. Desarrollo de programas bajo el paradigma de Objetos Uso del ABP y ABPy		
Recursos o herramientas.		Diapositivas del docente Plataforma Moodle Videos y páginas de la web. Sala de Sistemas Consultas en Internet. Computadores con acceso a Internet. VIdeobeam o Televisor. Herramientas ofimáticas. Herramienta de desarrollo para Java		
Unidad 04:		**Denominación global de la unidad temática:** patrones de diseño		
Proceso-resultado de aprendizaje.		**Proceso-resultado de aprendizaje.** Evaluar y presentar proyectos de modelado UML con patrones de diseño y arquitecturas		
Ítems	**Contenido de enseñanza**	**Trabajo Presencial o metodologías de enseñanza.**	**Trabajo presencial en equipos o independiente.**	**Trabajo independiente.**

emana 13	Introducción a Patrones de Diseño de Software. Diferencia entre Patrones GRASP y GoF.	Clase magistral sobre patrones de diseño.	Trabajo en equipos analizando patrones en diagramas UML.	Lectura de material sobre patrones de diseño.
Semana 14	Patrones de Diseño Creacionales: Singleton, Factory y Builder.	Explicación y modelado UML de patrones creacionales.	Desarrollo en equipos de patrones en UML.	Aplicación de patrones en diagramas individuales.
Semana 15	Patrones de Diseño Estructurales y de Comportamiento: Adapter, Decorator, Observer, Strategy. Proyecto de Modelado UML con Patrones de Diseño.	Clase magistral con ejemplos en UML.	Trabajo grupal en la representación UML de patrones.	Estudio de patrones y aplicación a problemas reales.
Semana 16	Examen Final y	Evaluación teórica y práctica en clase.		
Evidencia del logro del aprendizaje.		**Evidencia de producto:** Aplicación de los conocimientos adquiridos en las clases en el desarrollo de algoritmos y programas, con el fin de evaluar el entendimiento del grupo **Evidencia de conocimiento y procedimiento:** Desarrollo de Proyecto de aula.		
Recursos o herramientas.		Diapositivas del docente Plataforma Moodle Videos y páginas de la web.		

	Sala de Sistemas Consultas en Internet. Computadores con acceso a Internet. Videobeam o Televisor. Herramientas ofimáticas. Herramienta de desarrollo para Java

5. Evaluación o comunicación de los aprendizajes.

Proceso -resultado de aprendizaje (ver numeral 3-5).	**Evidencia de conocimiento, procedimiento o producto observable, donde los estudiantes comunican o demuestran el logro del aprendizaje sugerido (ver numeral 5).***	**Observación**	**Porcentaje.**	**Fecha**
Comprender y aplicar los diagramas UML en el modelado de clases y relaciones	Taller 1 (DIAGRAMAS UML)	Evaluación Grupal	20%	Semana 5
Introducción a la Programación Orientada a Objetos (POO)	Taller 2 POO	Evaluación Grupal	20%	Semana 10
Desarrollar la capacidad para modelar soluciones de software utilizando la metodología ABP y ABPy en UML	Evaluación de saberes – Examen Parcial	Evaluación individual / grupal	20%	Semana 7
Evaluar y presentar proyectos de modelado UML con patrones de diseño y arquitecturas	Proyecto final	Evaluación individual.	20%	Semana 15
Prueba escrita (parcial)	Evaluación de saberes – Examen Final	Evaluación individual.	20%	Semana 16

*Las evidencias y los productos de seguimiento y comunicación de los aprendizajes responden al 60% según el reglamento estudiantil (CA, TdeA, Acuerdo N° 6/2018). El parcial y el final responden cada uno al 20% para un total de 100%.

6. Referencias.

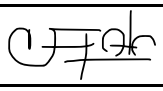
VILLALOBOS J., CASALLAS R. Fundamentos de programación: Aprendizaje activo basado en casos. Bogotá, Pearson - Prentice Hall. 2006.

JOYANES LUIS. Fundamentos de Programación, Algoritmos, estructuras de Datos y Objetos. Madrid: McGraw-Hill/Interamericana de España. Madrid, 2004

BOTERO R., CASTRO C., MAYA J., TABORDA G. y VALENCIA M.. Lógica y Programación orientada a objetos: un enfoque basado en problemas. CITIA, proyecto SISMOO, Tecnológico de Antioquia. Medellín, 2009.

Actualizado por: Santiago Hoyos	Aprobado por Comité Curricular.
Firma: Santiago Hoyos	Fecha:
Coordinador de área o línea o programa: Santiago Hoyos	Acta del Comité Curricular:
Firma: Santiago Hoyos	

8. Socialización.

Nombre del curso:	Paradigma OO	Grupo:	402
Fecha de la socialización:	08 / Nov / 2025	Firma del Docente:	
Nombres y Apellidos Estudiante- 1.		Firma estudiante-1.	
Nombres y Apellidos Estudiante- 2.		Firma estudiante-2.	