
 UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS	FORMATO DE SYLLABUS		Código: AA-FR-003	 SIGUD <small>Sistema Integrado de Gestión</small>
	Macroproceso: Direccionamiento Estratégico		Versión: 01	
	Proceso: Autoevaluación y Acreditación		Fecha de Aprobación: 27/07/2023	

FACULTAD:	FACULTAD DE INGENIERÍA				
PROYECTO CURRICULAR:	INGENIERÍA			CÓDIGO PLAN DE ESTUDIOS:	

I. IDENTIFICACIÓN DEL ESPACIO ACADÉMICO

NOMBRE DEL ESPACIO ACADÉMICO: Programación orientada a objetos

Código del espacio académico:		Número de créditos académicos:			3	
Distribución horas de trabajo:	HTD	2	HTC	4	HTA	3
Tipo de espacio académico:	Asignatura	X	Cátedra			

NATURALEZA DEL ESPACIO ACADÉMICO:

Obligatorio Básico	X	Obligatorio Complementario		Electivo Intrínseco		Electivo Extrínseco	
--------------------	---	----------------------------	--	---------------------	--	---------------------	--

CARÁCTER DEL ESPACIO ACADÉMICO:

Teórico		Práctico		Teórico-Práctico	X	Otros:		Cuál: _____
---------	--	----------	--	------------------	---	--------	--	-------------

MODALIDAD DE OFERTA DEL ESPACIO ACADÉMICO:

Presencial	X	Presencial con incorporación de TIC		Virtual		Otros:		Cuál: _____
------------	---	-------------------------------------	--	---------	--	--------	--	-------------

II. SUGERENCIAS DE SABERES Y CONOCIMIENTOS PREVIOS

Programación básica.

III. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

El aprendizaje de la programación orientada a objetos (POO) es altamente justificable dentro de la carrera de Ingeniería por varias razones fundamentales: 1. Modelado del mundo real. La POO se basa en el concepto de modelar objetos del mundo real y sus interacciones. Esto es fundamental para comprender y resolver los problemas del mundo real que suelen enfrentar los ingenieros de sistemas. 2. La capacidad de representar sistemas y procesos complejos a través de objetos y sus relaciones es esencial en la resolución de problemas de ingeniería.

IV. OBJETIVOS DEL ESPACIO ACADÉMICO (GENERAL Y ESPECÍFICOS)

Objetivo General:

Presentar al estudiante la conceptualización y aplicación del paradigma orientado a objetos, enfatizando en los elementos conceptuales propios de este que permitan plantear y aplicar modelos bien formados utilizando un lenguaje de programación orientado a objetos.

Objetivos Específicos:

- Determinar los tipos de aplicación y las situaciones en las que se debe aplicar el paradigma orientado a objetos.
- Comprender, interpretar y analizar el cambio de enfoque en el modo de resolver problemas que supone el uso del paradigma orientado a objetos respecto a otros paradigmas.
- Aplicar los conceptos del paradigma de programación orientada a objetos tales como: polimorfismo, encapsulamiento, herencia, sobrecarga, funciones virtuales, etc.
- Manejar adecuadamente conceptos tales como: recursividad, objetos transientes, residentes y persistentes; generalización y generacidad; clases plantillas; asociación, agregación y composición.
- Identificar problemas de: portabilidad, efectos colaterales y transparencia referencial.
- Comprender la enorme importancia de crear software fiable, reutilizable y mantenible.
- Dominar estrategias básicas de reutilización como son el uso de librerías o paquetes de software.
- Aplicar el modelo orientado a objetos en programación de dispositivos de cómputo.

V. PROPÓSITOS DE FORMACIÓN Y DE APRENDIZAJE (PFA) DEL ESPACIO ACADÉMICO

Competencias	Dominio-Nivel	RA	Resultados de Aprendizaje
Conoce y aplica los estándares y recomendaciones para la codificación de programas computacionales.	Cognitivo- Conocer	01	Aplicar los conceptos fundamentales de la programación orientada a objetos.
Conoce la evolución de los esquemas y mecanismos de abstracción.	Cognitivo- Conocer	02	Identificar y utilizar los diferentes paradigmas de programación, comprendiendo la importancia de la abstracción y el encapsulamiento en el desarrollo de software.
Comprende el modelo orientado a objetos y sus divisiones.	Cognitivo-modelar	03	Diseñar y modelar aplicaciones utilizando clases, atributos y métodos, aplicando el principio de encapsulamiento y ocultamiento de la información.
Modela clases, interfaces, sistemas y sus relaciones mediante diagramas de clases.	Cognitivo-modelar	04	Comprender el concepto de modularidad y sus criterios, principios y reglas, así como el uso de interfaces y el diseño de aplicaciones orientadas a objetos
Realiza un estudio de requerimientos para problemas sencillos.	Cognitivo- Comprender	05	Establecer relaciones entre clases, como asociación, generalización, especialización, composición y agregación.
Descubre las clases que permiten modelar una solución basado en los requerimientos.	Cognitivo-modelar	06	Entender e implementar la herencia simple y múltiple, así como la herencia de interfaz y de implementación.
Hace uso de las tarjetas CRC para plantear clases.	Cognitivo- Comprender	07	Comprender los beneficios y costos de la herencia y podrá tomar decisiones informadas sobre la técnica de reutilización más adecuada.
Implementa clases basado en los principios de diseño de clases.	Cognitivo- Comprender	08	Aplicar el polimorfismo en jerarquías de herencia, utilizar variables polimórficas y comprender el concepto de sobrecarga. Además, estará familiarizado con la genericidad y su aplicación en la programación orientada a objetos
El alumno tiene la capacidad de discernir que tecnología debe utilizar para la resolución de problemas particulares.	Cognitivo- Comprender	09	Aplicar principios de diseño y buenas prácticas en el desarrollo de software orientado a objetos
Comunica ideas de manera clara de forma oral o escrita.	Cognitivo- Conocer	10	Utilizar diagramas de clases para modelar sistemas y aplicar estrategias de almacenamiento.
Actúa estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos.	Afectivo- Cooperar	11	Comprender los principios de diseño de clase, como la cohesión y el acoplamiento, y aplicar el diseño por contrato.
		12	Seguir buenas prácticas en el diseño orientado a objetos y estar familiarizado con estándares de codificación y gestión de errores.
		13	Conocer sobre persistencia en archivos, prototipos de clase, diseño por capas, diseño de interfaz gráfica y el uso de herramientas de documentación.
VI. CONTENIDOS TEMÁTICOS			

1. El paradigma orientado a objetos (2 semanas)
- Identificación y evolución de los esquemas de abstracción.

- Modelado de requerimientos: Historias de usuario, casos de uso.

- El concepto de objeto.

- Aplicación de encapsulación y ocultamiento de la información.

- Modularidad (criterios, principios y reglas)

- El concepto de interfaz.

- El diseño de aplicaciones OO.

- Relaciones entre clases: dependencia, asociación, generalización, especialización.
2. Herencia y polimorfismo (2 semanas)
- Introducción a la herencia.

- Herencia simple.

- Herencia múltiple.

- Herencia de Interfaz.

- Herencia de Implementación.

- Beneficios y costes de la herencia.

- Elección de la técnica de reutilización.

- Polimorfismo y reutilización.

- Sobrecarga.

- Polimorfismo en jerarquías de herencia.

- Variables polimórficas.

- Genericidad.
3. Modelamiento en orientación a objetos (4 semanas)
- Importancia del modelamiento: documentación.

- Diagramas de clases.

- Esquemas dinámicos: secuencias, actividades.
4. Principios de diseño y buenas prácticas (4 semanas)
- Principios de diseño de clase: Principios de cohesión, principios de acoplamiento.

- Buenas orácticas en diseño O.O.

- Estándares de codificación y gestión de errores: validación y verificación.
5. Introducción al diseño por capas: MVC, 3 capas (3 semanas)
6. Concurrencia (1 semana)

VII. ESTRATEGIAS DE ENSEÑANZA QUE FAVORECEN EL APRENDIZAJE

Tradicional		Basado en Proyectos	X	Basado en Tecnología	X
Basado en Problemas	X	Colaborativo	X	Experimental	X
Aprendizaje Activo	X	Autodirigido	X	Centrado en el estudiante	X

VIII. EVALUACIÓN

Resultados de aprendizaje (RA) a ser evaluados:	Resultados de aprendizaje asociados a las evaluaciones (T: Teórico / P: Práctico)					
	Actividades Entregables	Talleres	Parciales	Informes de proyecto final	Proyecto final	Examen Finla
RA01						
RA02						
Tipo de evaluación**						EE
Porcentaje de evaluación (%)						30
Trabajo Individual (I) o Grupal (G)						I
Tipo de nota	0-5	0-5	0-5	0-5	0-5	0-5

IX. MEDIOS Y RECURSOS EDUCATIVOS

- Aula normal con tablero para sesiones de cátedra y para sesiones de discusión.
- Disponibilidad para acceder a proyector multimedia.

- Laboratorio de computación, para las sesiones de laboratorio.
- IDE's para desarrollar en java (Eclipse, Netbeans, ...)
- Página web para publicar material didáctico, guías de ejercicios, soluciones, tareas, etc.
- Acceso al material bibliográfico recomendado.
- Asignación de una persona que tenga las plenas competencias del curso (monitor) para asesorar a los estudiantes en dudas durante las sesiones del laboratorio de computación.
- Acceso a material digital a través de bibliotecas digitales

X. PRÁCTICAS ACADÉMICAS - SALIDAS DE CAMPO

- Asistencia a clases expositivas y de discusión
- Elaboración y lectura de paper (documentación).
- Se debe procurar incentivar el trabajo de grupo más que el trabajo individual. (se recomienda trabajar en grupos de dos o tres estudiantes)
- Implementación y prueba de prototipos (programas) en laboratorio de computación

XI. BIBLIOGRAFÍA

Referencias Básicas

1. Deitel, P., & Deitel, H. Python: Cómo programar. Pearson, 2020.	Una guía práctica y completa para aprender Python, incluyendo fundamentos de
2. Sedgewick, R. y Wayne, K. Algoritmos. Addison-Wesley, 2011.	Explica algoritmos esenciales y cómo aplicarlos en problemas básicos de programación.
3. C++ Primer (5th Edition). Stanley B. Lippman, Josée Lajoie, Barbara E. Moo.	Este libro es una referencia completa y accesible para los principiantes, cubriendo
4. C++ How to Program (10th Edition). Paul Deitel, Harvey Deitel. Editorial: Pearson.	Este libro combina teoría con una gran cantidad de ejemplos prácticos, haciéndolo ideal
5. "C: Cómo programar" - Deitel & Deitel	Un recurso fundamental para entender los conceptos básicos y avanzados de la
5. "Introducción a los Algoritmos" - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, y Clifford Stein	Este libro es una referencia estándar en algoritmos. Aunque avanzado, es una base sólida para estudiantes de ingeniería.
Parsons, D. y Bowe, H. Fundamentos de programación con Java .Pearson, 2019.	Enfocado en aprender la lógica de programación básica usando Java.

Recursos Complementarios

Tanenbaum, Andrew. Structured Computer Organization. Prentice Hall.
Levine, Guillermo. Computación y Programación Moderna. Addison Wesley.
Bajarme Stroustrup El C ++ Lenguaje de Programación Addison Wesley.
Burton Harvey, Simon Robinson, Julian Templeman, Karli Watson. C ++ Programming . Wrox Press Ltda.
Becerra, Cesar. Lenguaje C. Por Computador.
Rodríguez C., Llana L.F, Martínez, R., Palao P., Pareja, C. Ejercicios de Programación Creativos y Recreativos en C ++. Prentice Hall.

Recursos en línea

Páginas web:	https://geeksroom.com/2020/04/11-libros-programacion/128041/
Documentación de Python. Oficial de Python con ejemplos y conceptos básicos.	https://docs.python.org/
W3Schools: Excelente para aprender a programar con tutoriales interactivos.	https://www.w3schools.com/python/
Code Academy: Una plataforma interactiva para aprender los fundamentos de programación.	https://www.codecademy.com/learn/learn-python-3
GeeksforGeeks: Recurso muy utilizado para resolver problemas de programación y algoritmos.	https://www.geeksforgeeks.org/
Sitio oficial de C++. Un recurso esencial con documentación y ejemplos sobre todas las funciones y	cppreference.com
Una plataforma con explicaciones claras sobre algoritmos, estructuras de datos y fundamentos de	GeeksforGeeks (https://www.geeksforgeeks.org/)

XII. SEGUIMIENTO Y ACTUALIZACIÓN DEL SYLLABUS

Fecha revisión por Consejo Curricular:			
Fecha aprobación por Consejo Curricular:		Número de acta:	