UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERIA

SYLLABUS

PROYECTO CURRICULAR: Ingeniería de Sistemas

NOMBRE DEL DOCENTE:	
ESPACIO ACADÉMICO (Asignatura): Modelos de Programación II **Obligatorio (X) : Básico (X) Complementario ( )** **Electivo ( ) : Intrínsecas ( ) Extrínsecas ( )**	**CÓDIGO: 422**
NÚMERO DE ESTUDIANTES:	**GRUPO:**
NÚMERO DE CRÉDITOS: Tres (3)	
TIPO DE CURSO: TEÓRICO ( ) PRACTICO ( ) TEO-PRAC (X) Alternativas metodológicas: Clase Magistral (X), Seminario ( ), Seminario – Taller ( ), Taller (X), Prácticas (X), Proyectos tutoriados( ), Otro: _____	

HORARIO:

DÍA	HORAS	SALÓN

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

Competencias del perfil a las que contribuye la asignatura:	Esta asignatura contribuye al desarrollo de la competencia “Resuelve problemas computacionales algorítmicamente” que se encuentra en el dominio de “programación” del área “básicas de ingeniería” del proyecto curricular de ingeniería de sistemas.
Contribución a la formación:	Modelos de programación II es una materia que plantea la revisión de los diferentes paradigmas de programación y como se pueden abordar al momento de plasmar la solución computacional de un problema, se pretende mostrar aquí que si bien es cierto la orientación a objetos es el paradigma mas empleado, existen otras propuestas interesantes que vale la pena revisar como son por ejemplo la programación funcional , la programación lógica, entre otras; que establecen sus propios principios y que pueden ser bastante útiles a la hora de enfrentar un problema.
Puntos de apoyo para otras asignaturas:	• Estructura lógica conceptual basada en paradigmas de programación • Herramienta fundamental para ingeniería de software. • Herramienta fundamental para bases de datos • Herramienta fundamental para Redes
Requisitos previos:	• Programación estructurada • programación orientada a objetos • principios de diseño orientado a objetos • patrones de diseño

II. PROGRAMACIÓN DEL CONTENIDO

OBJETIVO GENERAL

Presentar al estudiante la conceptualización y aplicación de los paradigmas de programación Imperativa, Declarativa y script, enfatizando en los elementos conceptuales inherentes a cada uno ellos que permitan plantear y aplicar modelos bien formados, y la selección de un lenguaje de programación generando la capacidad en el estudiante de extraer de ellos técnicas y conceptos que pueden ser útiles en el paradigma orientado a objetos.

OBJETIVOS ESPECÍFICOS

1. Identificar las características fundamentales de los paradigmas de programación imperativa, declarativa, script.
2. Desarrollar habilidades y destrezas para el modelamiento de soluciones ajustadas a un paradigma de programación.
3. Analizar las características de un escenario de aplicación o el dominio de un problema para la óptima selección de un paradigma de programación, para la generación asertiva de modelos e implementaciones.
4. Reconocer, interpretar y describir conceptos fundamentales de diseño de lenguajes de programación y compiladores, como: Gramáticas Independientes de Contexto (BNF, EBNF), Recursión, Sistema de tipos, semántica, guardas y Estructuras de Control.
5. Modelar la solución de un problema en diferentes paradigmas de programación.
6. Desarrollar competencias para la traducción de modelos en diferentes lenguajes de programación

COMPETENCIAS DE FORMACIÓN:

Competencias que compromete la asignatura:	El estudiante está en capacidad de seleccionar y aplicar formas, técnicas y mecanismos útiles al momento de plasmar la solución computacional de un problema seleccionando estas del paradigma deseado según el contexto del problema.
Competencias específicas de la asignatura:	• El estudiante entiende el concepto de paradigma de programación y sus implicaciones en el modo de resolver problemas. • Conoce y entiende las diferencias entre los diferentes tipos de paradigmas. • Entiende e identifica los diferentes conceptos de programación en los diferentes paradigmas. • Entiende el tipo de problemas de desarrollo software que soluciona el paradigma imperativo procedimental. • Entiende el tipo de problemas de desarrollo software que soluciona el paradigma imperativo orientado a objetos. • Entiende el tipo de problemas de desarrollo software que soluciona el paradigma declarativo lógico. • Entiende el tipo de problemas de desarrollo software que soluciona el paradigma declarativo funcional. • Conoce las diferentes perspectivas hacia donde se desarrollan los paradigmas existentes y otros planteamientos. • El estudiante es capaz de reflexionar acerca de los diferentes paradigmas de programación.

Competencias Transversales a las que contribuye la asignatura:	• El alumno tiene la capacidad de discernir que tecnología debe utilizar para la resolución de problemas particulares. • Comunica ideas de manera clara de forma oral o escrita. • Actúa estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos.

PROGRAMA SINTÉTICO:

1. Introducción

1.1. Evolución de los lenguajes de programación
1.2. Paradigma de programación
1.3. Lenguajes Interpretados vs Lenguajes Compilados
1.4. Conceptos Lenguajes de programación: sintaxis, semántica, tipos de datos, guardas, estructuras de control

2. Paradigma Imperativo – Procedimental

2.1. Procesos iterativos
2.2. Definición de funciones
2.3. Estructuras del datos
2.4. Lenguajes representativos

3. Paradigma Imperativo - POO

3.1. Clases, Objetos
3.2. Herencia Polimorfismo
3.3. Principios de Diseño
3.4. Lenguajes representativos

4. Paradigma Declarativo – Programación Lógica

4.1. Lógica de Primer Orden – Clausulas de Horn
4.2. Hechos y Reglas
4.3. Consultas
4.4. Lenguajes representativos

5. Paradigma Declarativo – Programación Funcional

5.1. Calculo-λ
5.2. Definición de procedimientos
5.3. Recursión
5.4. Lenguajes representativos

6. Perspectivas y otros modelos

6.1. Programación script
6.2. Programación orientada a aspectos
6.3. Programación guiada por eventos
6.4. Programación lógica con restricciones
6.5. Lenguajes multiparadigma

III. ESTRATEGIAS

Metodología Pedagógica y Didáctica:

- 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

	Horas			Horas Profesor/ semana	Horas Estudiante/semana	Total Horas Estudiante/semestre	Créditos
Tipo de Curso	**TD**	**TC**	**TA**	**(TD + TC)**	**(TD + TC +TA)**	**X 16 semanas**	
	2	2	5	4	9	144	3

Trabajo Presencial Directo (TD): trabajo de aula con plenaria de todos los estudiantes.

Trabajo Mediado_Cooperativo (TC): Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.

Trabajo Autónomo (TA): Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio, etc.

IV. RECURSOS

Medios y Ayudas:

- 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 los lenguajes seleccionados (se sugieren mozart-OZ y python)
- 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.

BIBLIOGRAFÍA

TEXTOS GUÍA

TEXTOS COMPLEMENTARIOS

- Bratko, Ivan. "Prolog : Programming for Artificial Intelligence". Addison-Wesley, 1990
- P.H. Winston and B.K.P. Horn, Lisp, Tercera Edición, Addison-Wesley Iberoamericana, 1991
- Thompson, Simon. “Haskell. The Craft of Functional Programming”. Second Edition. Addison-Wesley.

REVISTAS

DIRECCIONES DE INTERNET

- http://www.haskell.org/tutorial/ A Gentle Introduction to Haskell. Paul Hudak, John Peterson y Joseph H. Fasel.
- http://www.mozart-oz.org/ The Mozart Programming System

V. ORGANIZACIÓN / TIEMPOS

Espacios, Tiempos, Agrupamientos:

Se recomienda trabajar una unidad cada cuatro semanas, trabajar en pequeños grupos de estudiantes, utilizar Internet para comunicarse con los estudiantes para revisiones de avances y solución de preguntas (esto considerarlo entre las horas de trabajo cooperativo).

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**Introducción**																
	Evolución de los lenguajes de programación																
	Paradigma de programación																
	Lenguajes interpretados vs lenguajes compilados																
	Conceptos lenguajes de programación: sintaxis, semántica, tipos de datos, guardas, estructuras de control																
2	**Paradigma Imperativo - procedimental**																
	Procesos iterativos																
	Definición de funciones																
	Estructuras del datos																
	Lenguajes representativos																
3	**Paradigma Imperativo - POO**																
	Clases, objetos																

	Herencia polimorfismo																
	Principios de diseño																
	Lenguajes representativos																
4	**Paradigma Declarativo – Programación Lógica**																
	Lógica de primer orden – clausulas de Horn																
	Hechos y reglas																
	Consultas																
	Lenguajes representativos																
5	**Paradigma declarativo programación funcional**																
	Calculo-λ																
	Definición de procedimientos																
	Recursión																
	Lenguajes representativos																
6	**Perspectivas y otros modelos**																
	Programación script																
	Programación orientada a Aspectos																
	Programación guiada por eventos																
	Programación lógica con restricciones																
	Lenguajes multiparadigma																

VI. EVALUACIÓN

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMER CORTE			
SEGUNDO CORTE			
PROYECTO FINAL			30,00%

ASPECTOS A EVALUAR DEL CURSO

- Claridad y entendimiento de los conceptos.
- Que se haya identificado correctamente el problema y que el modelo lo represente adecuadamente.
- Que la solución diseñada resuelva el problema.

- Apego a la formalidad y estándares requeridos.
- Que el análisis de corrección sea exhaustivo.
- Que el prototipo corresponda al modelo diseñado y no presente errores de sintaxis.
- La asistencia a las clases magistrales y a los laboratorios.
- El esfuerzo y dedicación en la resolución de problemas.
- Que la documentación permita reconocer la forma en que se ha abordado el problema y la estructura del programa implementado.
- En las pruebas escritas se consideran en forma parcial los aspectos considerados en proyectos de programación bajo problemas que requieren un menor tiempo de desarrollo y en una modalidad que no requiere uso del computador, así como la comprensión conceptual.

DATOS DEL DOCENTE

NOMBRE :

PREGRADO :

POSTGRADO :

ASESORIAS: FIRMA DE ESTUDIANTES

NOMBRE	FIRMA	CÓDIGO	FECHA
1. **2.** **3.**			

FIRMA DEL DOCENTE

_____

FECHA DE ENTREGA: