Universidad Tecnológica de Pereira

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

MATERIA:	**PROGRAMACIÓN ORIENTADA A OBJETOS**
CÓDIGO:	**IS563**
CRÉDITOS:	**3**
INTENSIDAD:	**4 Horas semanales para 64 horas totales**
REQUISITO:	**IS304 Estructura de datos**

JUSTIFICACIÓN:

Lisp, SML, Java, Perl, Python, C++, TCL, y muchos más. ¿Porqué hay tantos lenguajes de programación? ¿No son suficientes? ¿Porqué se siguen desarrollando nuevos lenguajes? El origen de un lenguaje de programación determina sus principales características. Lisp es un lenguaje que se basó en las matemáticas y Fortran otro lenguaje basado en la arquitectura de máquina. Esto los hace muy diferentes. Cada lenguaje tiene sus ventajas para la solución de determinada clase de problemas. Y los problemas que la computación trata de resolver son cada día más complejos.
Una primera clasificación ubica a los lenguajes de programación en dos clases: lenguajes imperativos y lenguajes declarativos. La filosofía subyacente al paradigma declarativo se resume en la conocida ecuación de Kowalski:

Algoritmo = Lógica + Control

La parte humana debe encargarse de la lógica mientras que la máquina debe estar encargada completamente del control. Partiendo de la lógica matemática (cálculo de predicados), Kowalski y otros crearon el paradigma de la programación lógica. Pero dentro de cada una de las clases anteriores hay otros paradigmas: Programación funcional, programación orientada a objetos, programación concurrente, etc., cada uno con sus ventajas (y desventajas). Para quienes desarrollan algunos de estos lenguajes es muy importante que el lenguaje soporte varios de estos paradigmas permitiendo una gran capacidad expresiva.

Dos de los paradigmas más interesantes que se han desarrollado en las últimas décadas son la programación concurrente con restricciones y la programación con objetos activos. La programación con restricciones, que puede verse como un desarrollo mas avanzado de la programación lógica, permite resolver problemas para los que no se conocen algoritmos eficientes y que requieren búsqueda. Y la programación con agentes activos permite modelar situaciones en las que múltiples agentes se encargan, básicamente mediante el paso de mensajes, de resolver diferentes clases ce pro Lemas. Mozart es un lenguaje desarrollado a partir de un núcleo declarativo que soporta muchos de los principales paradigmas de programación. Sus características le dan muchas ventajas tanto en el aspecto técnico como en el aspecto pedagógico siendo uno de los lenguajes más adecuados para estudiar y aplicar los distintos paradigmas.

OBJETIVO GENERAL:

Estudiar los fundamentos de varios paradigmas de programación como son: la programación Orientada a Objetos y complementar con la Programación Lógica y Relacional, Concurrente por Restricciones y la de objetos activos; igualmente aplicarla a la solución de problemas de la vida real.

OBJETIVOS ESPECÍFICOS:

1. Eastudiar los conceptos de POO y poner en práctica dichos conceptos en lenguaje JAVA.
2. Estudiar las restricciones en programación como técnica para la solución de problemas combinatorios y poner en práctica dichios concepton en PROLOG
3. Estudiar el uso de las restricciones en programación como modelo formal de computación.
4. Construir una aplicación para resolver un problema de optimización usando las técnicas estudiadas.
5. Estudiar un lenguaje multiparadigma: Mozart.

CONTENIDO:

1. INTRODUCCIÓN
1.1. Historia de algunos lenguajes de programación.
1.2. Programación imperativa y programación declarativa.
1.3. Principales paradigmas de la programación.
1.4. Porqué nuevos lenguajes?
1.5Lectura: la venganza de los nerds. Paul graham.

2. PARADIGMA ORIENTADO A OBJETOS (OO):
2.1 Conceptos introductorios al paradigma.
2.2 Clases y objetos.
2.3 Reusabilidad y extensibilidad.
2.4 Abstracción.
2.5 Encapsulamiento.
2.6 Interfase (protocolo).
2.7 Mecanismo de herencia.
2.8 Polimorfismo: estático y dinámico.
2.9 Mensajes.
2.10 Relaciones de generalización/especialización, todo/parte, asociación y uso.
2.11 Cardinalidad. Modelos de especificación.
2.12 PATRONES en un lenguaje OO:

3. PROGRAMACIÓN LÓGICA Y RELACIONAL
3.1. De la lógica de predicados a la programación.
3.2. Programación lógica. Prolog.
3.3. Programación relacional.

4. LENGUAJES MULTIPARADIGMA. MOZART
.1. El núcleo declarativo. Variables de flujo de datos.
3.2. Programación con recursión.
3.3. Programación funcional.
3.4. Concurrencia. Hilos.

4. PROGRAMACIÓN CONCURRENTE CON RESTRICCIONES
4.1. Propagadores.
4.2. Estrategias de distribución.

4.3. Servicios de programación con restricciones.
4.4. Modelos y aplicaciones.

5. PROGRAMACIÓN CON OBJETOS ACTIVOS
5.1. Programación orientada a objetos.
5.2. Puertos. Paso de mensajes.
5.3. Programación con agentes.
5.4. Aplicaciones.

METODOLOGIA
El curso se dictará con base en clases magistrales y con el apoyo de recursos multimediales cuando ello convenga. Además, se realizarán prácticas en computadora (Java) para dar solidez a los temas vistos en clase.

Dentro del esquema de formación integral del ser humano, el profesor podrá traer temas y ayudas que le permitan al estudiante reconocer la historia de la ciencia y la responsabilidad de la tecnología frente a la sociedad. Estos temas y ayudas se presentaran a discrecionalidad del profesor

COMPETENCIAS
COMPETENCIAS TRANSVERSALES / GENÉRICAS:
- Aprendizaje autónomo
- Capacidad de análisis y síntesis
- Capacidad de aplicar los conocimientos a la práctica
- Resolución de problemas
- Trabajo individual y por parejas
- Comunicación oral y escrita

COMPETENCIAS ESPECÍFICAS:
- Cognitivas (Saber):
- Idioma
- Matemáticas
- Nuevas tecnologías TIC
- Conocimientos de informática
- Procedimentales / Instrumentales (Saber hacer):
- Redacción en interpretación de documentación técnica
- Estimación y programación del trabajo
- Planificación, organización y estrategia.
- Actitudinales (Ser):
- Calidad
- Toma de decisión
- Capacidad de iniciativa y participación

TÉCNICAS DOCENTES
Las técnicas docentes que se van a utilizar son:
- Clases de teoría
- Exposiciones sobre trabajos de casos prácticos.
- Tutorías colectivas de teoría
- Clases de prácticas
- Corrección de las prácticas
- Tutorías colectivas de prácticas
- Tutorías individualizadas

MECANISMOS DE CONTROL Y SEGUIMIENTO

El profesor podrá comprobar el grado de seguimiento de la asignatura mediante:

- La asistencia a las clases de teoría y prácticas
- Las exposiciones de temas de teoría.
- La corrección de las prácticas.
- Las tutorías personales
- Los parciales

EVALUACIÓN

Según el reglamento estudiantil vigente, en sus artículos 72 y 73."*…**ARTÍCULO 72o.:** Se entiende por Prueba Parcial aquella que se realiza individualmente para verificar el logro de los objetivos de las diferentes unidades o temas en que se divide cada asignatura. Estas **no podrán ser menos de dos** para cada asignatura... **ARTÍCULO 73o**.: Se entiende por Prueba Final aquella que se realiza individualmente para verificar el logro de los objetivos generales de cada asignatura. Esta prueba se realizará con estricta observancia de las fechas establecidas en el calendario académico...",* subrayado y resaltado nuestro.

Se recomienda al profesor el siguiente sistema de evaluación:

I Previa:	20%
II Previa:	20%
Talleres, Tareas, etc.:	20 %
Exàmen Final:	20 %
Proyectos:	20%

BIBLIOGRAFIA

- David J. Barnes, Michael Kolling , "Objects First With Java: A Practical Introduction Using BlueJ", Ed. Prentice Hall; 4 edition (September 1, 2008.
- Allen Holub, " Holub on Patterns: Learning Design Patterns by Looking at Code", Apress (September 27, 2004)
- Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, " Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional; (November 10, 1994).
- Leon Sterling, Ehud Shapiro, " The Art of Prolog, Second Edition: Advanced Programming Techniques", Ed. The MIT Press; 2 edition (March 10, 1994).
- Peter Van Roy, Seif Haridi, "Concepts, Techniques, and Models of Computer Programming", The MIT Press (March 1, 2004.
- K. R.. Apt, "From Logic Programming to Prolog", Ed. Prentice Hall. 1997.
- C. S. Clocksin, C. W. Mellish, "Programacion en Prolog", Ed. Gustavo Gili. 1988.

· J. Lloyd, “Foundations of Logic Programming”, Ed. Springer-Verlag. 1987.

· Ulf Nilsson, Jan Maluszynski, “Logic,