UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Programa Ingeniería de Sistemas

PROGRAMACIÓN ORIENTADA A OBJETOS

CÓDIGO: 22951	**NÚMERO DE CRÉDITOS: 4**
REQUISITOS:	
INTENSIDAD HORARIA SEMANAL: 12 HORAS	**TAD: 5** HORAS **TI: 7** HORAS
TALLERES: 1 **LABORATORIO: 3** **TEÓRICA: 1**	

JUSTIFICACIÓN

El continuo aprendizaje se fortalece en la investigación, es ahí donde el estudiante como parte de su compromiso con la academia debe enfatizar. Lograr la excelencia profesional sólo es posible si se crea una cultura mental desde los inicios de la carrera, parte de este logro dependerá de los contenidos que el programa ofrece, sin embargo, no se puede obviar la profundización de la temática que haga el estudiante.

En esa medida, desde el punto de vista de la academia, se considera importante perfilar los contenidos programáticos para que le proporcionen al estudiante herramientas necesarias a fin de convertir las debilidades en fortalezas y así a futuro, encontrar oportunidades de involucrarse con sus diferencias competitivas y comparativas en el ámbito laboral.

PROPÓSITO Y COMPETENCIAS

PROPÓSITOS:

- Perfilar la filosofía del diseño orientado a objetos y los conceptos de encapsulación, clases, herencia y polimorfismo.
- Desarrollar programas sencillos de programación orientada a objetos y aplicaciones para ambientes de sistemas de interfaces visuales.

COMPETENCIAS:

El estudiante:

- Interpreta mediante el razonamiento lógico, el análisis y la reflexión, diversos modelos UML en términos de diagramas de casos de uso, diagramas de clases, diagramas de actividades y diagramas estado.
- Propone problemas prácticos y teóricos mediante su formulación lógica para resolver dichos problemas mediante propuestas de especificaciones de requisitos (diagramas de casos de uso), diseños de vistas: estáticas (diagramas de clases), funcionales (diagramas de actividades) y dinámicas (diagramas de estado).
- Argumenta el porqué de los modelos matemáticos a utilizar en la resolución de problemas prácticos y teóricos específicos de las diferentes áreas de actividad de su profesión utilizando los algoritmos apropiados para las representaciones que requiera.
- Construye programas orientados a objetos con interfaces de usuario en líneas de comandos y gráficas.
- Selecciona los elementos constitutivos de los lenguajes de programación y diseño orientados a objetos para el desarrollo de sistemas software orientados a objetos y conducidos por eventos.

CONTENIDO:

Construcción y reparación de programas

1.1. Introducción
1.2. El nuevo concepto de la programación
1.3. La composición de una aplicación
1.4. La construcción de aplicaciones con interfaz de texto
1.5. La construcción de aplicaciones con interfaz gráfica
1.6. El mantenimiento de programas
1.7. La construcción de programas
1.8. La transferencia de datos entre funciones

2. Tipos abstractos de datos: Especificación e implementación
2.1. Introducción
2.2. Especificación
2.3. Implementación

3. Objetos y clases: Conceptos, diseño e implementación
3.1. Introducción
3.2. Los objetos
3.3. Las clases
3.4. Notaciones
3.5. Implementación

4. Los pilares de la programación orientada a objetos
4.1. Introducción
4.2. ¿Qué es la programación orientada a objetos?
4.3. Los mecanismos básicos de la programación orientada a objetos
4.4. Características de la programación orientada a objetos
4.5. Los lenguajes orientados a objetos
4.6. El modelo de la programación orientada a eventos

5. Herencia
5.1. Definición
5.2. Herencia simple: modelado e implementación
5.3. Herencia múltiple: modelado e implementación
5.4. Clases abstractas: modelado e implementación
5.5. Anulación de operaciones

6. Polimorfismo
6.1. Definición
6.2. Polimorfismo de funciones
6.3. Polimorfismo de operadores
6.4. Funciones virtuales

7. Enlaces y asociaciones
7.1. Conceptos
7.2. Multiplicidad
7.3. Atributos para los enlaces
7.4. Agregación
7.5. Composición
7.6. Modelado de sistemas complejos

8. Plantillas
8.1. Definiciones
8.2. Plantillas de funciones
8.3. Plantillas de clases
8.4. Plantillas predefinidas en los lenguajes de programación

9. Clases contenedoras de datos: Modelado e implementación
9.1. Definiciones
9.2. La clase listas enlazadas
9.3. La clase pila
9.4. La clase cola
9.5. La clase árbol
9.6. Clases contenedoras predefinidas en los lenguajes de programación

ESTRATEGIAS DE ENSEÑANZA Y APRENDIZAJE QUE APOYARAN EL TAD Y EL TI

Este curso se desarrolla mediante exposiciones del profesor con participación de los estudiantes en la discusión de conceptos y solución de problemas y con prácticas en el laboratorio de computadores. Las exposiciones teóricas se realizan en dos horas y las prácticas en tres horas semanales. Entre las estrategias pedagógicas para el logro de los propósitos están:

- Uso de Bibliotecas de clases para que el estudiante analice y utilice su código.
- Laboratorios con prácticas diseñadas y probadas previamente por el grupo de profesores del área.
- Talleres para trabajos en equipo.
- Documentos de lectura puestos en el portal institucional con administración de contenidos por parte del profesor
- Casos de estudio puestos en el portal institucional con administración por parte del profesor.
- Enunciados de problemas para que el estudiante los solucione. Documentos virtuales puestos en el portal institucional con administración del profesor.

ESTRATEGIAS DE EVALUACIÓN

Indicadores de logros

- Comprende los conceptos generales en los que se fundamenta las tecnologías orientada a objetos, tales como: Complejidad, abstracción, encapsulado y reutilización.
- Comprende la importancia de crear software correcto, fiable, reutilizable, fácil de uso y mantenimiento.
- Demuestra capacidad de abstracción necesaria para las tareas de análisis y diseño de software orientado a objetos.
- Interpreta la información contenida en un diagrama de clases UML y a partir de éste codifica dicha información en un lenguaje de programación orientado a objetos.
- Aplica los conceptos de clases, objetos, herencia y polimorfismo en forma correcta al proponer soluciones de problemas utilizando las tecnologías orientadas a objetos.
- Realiza en forma correcta la parte operativa con los entornos de desarrollo orientados a objetos e interpreta en forma correcta las partes que conforman los diferentes proyectos de software desarrollados en estos entornos orientados a objetos.
- Diferencia entre un sistema de software no orientado a objetos y uno orientado a objetos
- Interpreta los diferentes diagramas de diseño y es capaz de convertir estos diseños en código fuente en un lenguaje de programación orientado a objetos.
- Aplicar en forma correcta las clases predefinidas por el entorno de desarrollo escogido para las prácticas de laboratorio.
- Descompone un problema en clases con las definiciones de los objetos que se involucran dentro del contexto del problema.
- Reconoce los tipos de métodos gestores de los eventos que ocurren en la interacción del usuario con las interfaces graficas de usuario.
- Manipula en forma adecuada la información (datos y métodos) definidos dentro de un sistema de clases.
- Aplica los conceptos orientada a objetos en el aprendizaje de los lenguajes orientados a objetos utilizados para el desarrollo de las prácticas en el laboratorio.
- Comprende y aplica algunas características opcionales de los lenguajes orientados a objetos como la persistencia, los mecanismos de gestión de errores o la concurrencia y las plantillas de las clases genéricas.
- Utiliza con facilidad los entornos de desarrollo como son Java y .NET

Estrategias de evaluación

Durante el semestre se realizarán mínimo tres evaluaciones. El porcentaje de cada una de estas evaluaciones debe ser menor al cincuenta por ciento del total de la nota. De estas tres por lo menos dos evaluaciones escritas. La tercera evaluación puede ser dividida en trabajos de laboratorio cuestionarios sobre un determinado visto en clase o conceptos que el estudiante debe aplicar en el

desarrollo de sus trabajos de laboratorio y fuera del aula de clase teórica o práctica. También puede constituirse una evaluación por controles de asistencia y participación en clase. Las ponderaciones para cada una de las evaluaciones serán asignadas por el profesor al comenzar el semestre y estas deben cumplirse sin modificaciones durante o al final del semestre.

Equivalencia cuantitativa

La evaluación de la asignatura se define mediante la calificación de las estrategias de evaluación, con calificaciones de cero punto cero (0.0) a cinco punto cero (5.0). Otras consideraciones de acuerdo al reglamento de pregrado de la universidad.

BIBLIOGRAFÍA

Libros clásicos

- Fundamentos de Java.Herbert Schildt, ed. Mc Graw Hill, tercera edición, 2007.
- Java Cómo Programar, Deitel, ed. Prentice Hall, quinta edición, 2004.
- El Lenguaje Unificado de Modelado. Manual de referencia, RUMBAUGH, James / JACOBSON, Ivar / BOOCH Grady, ed. Addison Wesley.
- El Lenguaje Unificado de Modelado, RUMBAUGH, James / JACOBSON, Ivar / BOOCH Grady, ed. Addison Wesley.
- El Lenguaje Unificado de Modelado – Introducción, RUMBAUGH, James / JACOBSON, Ivar / BOOCH Grady, ed. Addison Wesley.
- UML y Patrones, LARMAN, Craig, ed. Prentice Hall.

Libros herramientas de desarrollo

- Programación en Java 5.0, James Cohoon/Jack Davidson. Ed. Mc Graw Hill, primera edición,2006.
- Estructuras de datos, Luis Joyanes Aguilar/Ignacio Zahonero Martinez, ed. Mc Graw Hill. Primera edición, 2008.
- Java 2 Características avanzadas, Cay S. Horstmann / Gary Cornell, ed. Prentice Hall, 2003.
- Java 2 Lenguaje y Aplicaciones, Fco Javir Ceballos, ed. Alfaomega y Ra-Ma, 2007
- UTILIZACION DE UML EN INGENIERÍA DEL SOFTWARE CON OBJETOS Y COMPONENTES / Perdita Stevens, Rob Pooley
- INGENIERÍA DE SOFTWARE ORIENTADA A OBJETOS CON UML, JAVA E INTERNET / Alfredo Wetzenfeld
- Microsoft C# Lenguajes y aplicaciones, CEVALLOS, Francisco Javier. Alfaomega-Rama.
- Microsoft Visual C#. NET Aprenda YA, SHARP, John / JAGGER, Jon. McGraw- Hill.
- Desarrollo de aplicaciones .NET con Visual C#, RODRÍGUEZ Miguel / BESTEIRO Marco Antonio, ed. McGraw-Hill.
- Apuntes de la Programación orientada objetos, HERRERA CASTILLO, Jorge.
- Manual de referencia C#, SCHILDT, Herbert, ed. McGraw-Hill.