IDENTIFICACIÓN			
Nombre de la asignatura	Programación de computadores II		
Código de la asignatura	SS300		
Programa Académico	Ingeniería de Sistemas		
Créditos académicos	3		
Trabajo semanal del estudiante	Docencia directa: 4	Trabajo Independiente: 5	
Trabajo semestral del estudiante	144		
Pre-requisitos	Programación de Computadores I		
Co-requisitos			
Departamento oferente	Ingenierías y Tecnologías		
Tipo de Asignatura	Teórico:	Teórico-Práctico: X	Práctico:
Naturaleza de la Asignatura	Habilitable:	No Habilitable: **X**	
	Validable:	No Validable: X	
	Homologable:	No Homologable: X	

PRESENTACIÓN

Esta asignatura precisa necesariamente de un conocimiento básico de los conceptos de programación, y el paradigma de la programación orientada a objetos está principalmente enfocada a mejorar el diseño de los propios programas. De esta manera, se introduce al alumno en la creación de programas desde un punto de vista más abstracto que la simple algorítmica básica, promoviendo un buen diseño enfocado a resolver los aspectos más relevantes del problema, mientras lo específico queda oculto. De este modo, a diferencia de las asignaturas orientadas a la programación procedural, donde la sintaxis y la semántica del lenguaje son primordiales, esta asignatura se orienta hacia el diseño y estructura de los programas en su conjunto. Sin embargo, también se introduce y se enseña el lenguaje Java para implementar los programas diseñados.

La organización de la asignatura sigue la siguiente estructura: Al principio de la asignatura se estudian los conceptos y principios fundamentales de la programación orientada a objetos, como la abstracción, la encapsulación y la reutilización. En una segunda parte de la asignatura, utilizando los conocimientos básicos anteriores, se introducen conceptos más avanzados como la herencia, el polimorfismo, la sobrecarga y las interfaces; estudiando especialmente como aplicarlos y utilizarlos en la construcción de programas orientados a objetos. En la última parte de la asignatura se aprende como aplicar todos los conceptos en su conjunto para resolver problemas complejos. Eso se desarrolla mediante el estudio del modelado de objetos y sus relaciones, así como la reutilización de código, y también el estudio de problemas concretos resueltos utilizando objetos.

JUSTIFICACIÓN

La Orientación a Objetos - OO, es un paradigma de programación que permite reducir la complejidad inherente al desarrollo de software, mediante la abstracción de datos, la división del programa en módulos manejables y la reutilización de código, permitiendo reducir el tiempo y esfuerzo en la creación de soluciones y desarrollo de pruebas, por lo cual, se ha consolidado a través de los años como una de las herramientas indispensables para la ejecución de proyectos empresariales de construcción de software, y una de las más utilizadas en los últimos desarrollos empresariales implementados.

La industria moderna del software tiene en la programación orientada a objetos toda una filosofía para conceptuar y representar la realidad tangible e intangible, debido a la versatilidad que proporciona a los programadores y a la flexibilidad que demuestra para el desarrollo de sistemas de información. Es por esto, que en la actualidad la orientación a objetos se constituye en una herramienta fundamental para el diseño y desarrollo de aplicaciones para computador, y como evidencia de ello, el crecimiento exponencial del número de lenguajes programación que soportan la orientación a objetos que se ha dado desde sus orígenes.

Es tanta la importancia que ha tomado este paradigma de programación, que las universidades más importantes e influyentes del mundo incluyen dentro de su sistema curricular, como curso básico y fundamental, los conceptos de la orientación a objetos en etapas de iniciación a las ciencias de la computación, con la finalidad de aportar a los estudiantes y futuros ingenieros elementos innovadores en cuanto a métodos y estrategias adoptadas para la solución de problemas susceptibles de implementar en computadora mediante la concepción de algoritmos.

Por tanto, la posibilidad de ofrecer a los estudiantes, de manera complementaria a los conceptos de estructuras algorítmicas tradicionales, las técnicas y mecanismos de la programación orientada a objetos, permitirá garantizar, desde los procesos formativos en cierne, la formación de profesionales competentes, con dominio de las herramientas fundamentales requeridas por la actual industria del software.

OBJETIVO GENERAL

Lograr que el estudiante domine los conceptos y técnicas básicas del Paradigma de Programación Orientada a Objetos, y los aplique en el diseño y construcción de soluciones de software de mediana complejidad, para obtener como resultado aplicativos robustos, fiables, flexibles, mantenibles, escalables y bien documentados.

OBJETIVOS ESPECÍFICOS

- Comprender el concepto de paradigma de programación y las principales diferencias conceptuales entre los paradigmas imperativo, funcional, y orientado a objetos.
- Entender la estructura de un programa organizado con objetos.
- Diseñar las clases necesarias y construir los objetos requeridos que permitan dar solución a problemas computacionales en términos del paradigma orientado a objetos.
- Reconocer los principales elementos del Lenguaje Unificado de Modelado y representar mediante diagramas UML las clases y relaciones requeridas para construir soluciones de software dentro del paradigma orientado a objetos.
- Reconocer las características y sintaxis de un lenguaje de programación basado en el paradigma orientado a objetos.
- Desarrollar aplicaciones de mediana complejidad utilizando objetos, clases y demás mecanismos propios de la POO.
- Implementar programas utilizando un lenguaje de programación de alto nivel orientado a objetos (opcional Java).

COMPETENCIAS GENERALES Y ESPECÍFICAS

COMPETENCIAS GENERALES

Competencias instrumentales:

- Capacidad de análisis y síntesis.
- Capacidad de organizar y planificar.
- Conocimientos básicos de la carrera.
- Comunicación oral y escrita.
- Habilidades del manejo de la computadora.
- Habilidad para buscar y analizar información proveniente de fuentes confiables.
- Habilidad para la solución de problemas.
- Capacidad para la toma de decisiones.

Competencias interpersonales:

- Capacidad crítica y autocrítica.
- Ser capaz de trabajar en equipo
- Compromiso con el trabajo, que permita una buena planificación de éste y la consecución de entregables en un plazo prescrito.
- Tener motivación por la calidad del software producido.

Competencias sistémicas:

- Ser capaz de discernir los distintos tipos de aplicación y las situaciones en las que es posible y necesario aplicar el paradigma orientado a objetos.
- Ser capaz de comparar distintos lenguajes de programación orientados a objetos y apreciar sus ventajas e inconvenientes con base en su grado de implementación de las principales características del paradigma orientado a objetos.
- Ser capaz de aprender y aplicar de forma autónoma nuevos conocimientos y métodos relacionados con el paradigma orientado a objetos.

COMPETENCIAS ESPECIFICAS

- Capacidad para solucionar problemas mediante el diseño y creación de algoritmos.
- Utilizar técnicas de modelado para la solución de problemas.
- Aplicar la sintaxis de un lenguaje orientado a objetos.
- Aplicar un lenguaje orientado a objetos para la solución de problemas.

METODOLOGÍA

Las diferentes actividades realizadas durante la asignatura se dividen en:

Sesiones de teoría: donde se introducen y se explican los diferentes conceptos y principios de la programación orientada a objetos.

Sesiones de práctica: donde los alumnos, tienen que trabajar para implementar programas escritos en el lenguaje orientado a objetos que resuelvan diferentes problemas, aplicando las técnicas expuestas en las clases de teoría.

Proyectos de aula: donde los alumnos, organizados en grupos, tienen que diseñar y modelar, utilizando objetos, una solución a los problemas particulares presentados, las cuales más tarde tendrán que implementar en las sesiones de prácticas.

Trabajo Independiente: Esta estrategia corresponde al autoaprendizaje por parte del estudiante. Para ello, debe documentarse y preparar los diferentes temas de la asignatura con anticipación; teniendo en cuenta el contenido suministrado por el docente, utilizando las diferentes fuentes bibliográficas y las franjas de usuario programadas en las salas de informática.

Ejercicios individuales (actividad opcional): donde los alumnos tendrán la oportunidad de resolver ejercicios de programación, fomentando los conceptos de la programación orientada a objetos.

Horas de Asesoría: Esta estrategia corresponde a la asesoría que debe brindar el docente a los estudiantes, sobre las tareas asignadas y en horas estipuladas independientemente de las horas de docencia directa.

ESTRATEGIAS METODOLÓGICAS

Las primeras sesiones (clases) estarán dedicadas a la presentación de la asignatura, la revisión de su importancia, sus diferencias, las ventajas y desventajas frente a otros paradigmas de programación, la difusión de los conceptos básicos e imprescindibles para el correcto entendimiento de la materia y al establecimiento de las bases y normas de la dinámica de trabajo de los participantes en el proceso de evaluación continua.

Así mismo, se desarrollaran sesiones de clases teórico-prácticas, con el fin de fortalecer el grado de adquisición de competencias que los participantes en el proceso formativo muestren.

En la asignatura se hará uso combinado de las siguientes estrategias metodológicas:

- ✓ Exponer los objetivos de la clase
- ✓ Exposición de los temas, apoyándose de medios audiovisuales de enseñanza.
- ✓ Análisis de casos de estudio de POO.
- ✓ Analogías (Fundamenta la abstracción)

- ✓ Ilustración a través diagramas de clases
- ✓ Ejemplos de programas de POO.
- ✓ Resolución de problemas
- ✓ Realización de prácticas en computadora.
- ✓ Proyecciones relacionadas con el tema.
- ✓ Participación en Foros y Mesas redondas.
- ✓ Proyecto de aula de clases
- ✓ Lecturas
- ✓ Realización de talleres de investigación formativa
- ✓ Tutoría y asesoría en clase

PROYECTO AULA DE CLASES: debe contener por lo mínimo los siguientes componentes:

- ✓ Implícitamente debe contener una CRUD (crear, consultar, modificar y eliminar)
- ✓ Métodos dinámicos
- ✓ Encapsulamiento
- ✓ Colecciones de objetos
- ✓ Agregación y composición de clases
- ✓ Herencia, polimorfismo
- ✓ Manejo de excepciones
- ✓ Persistencia de datos
- ✓ Interfaz grafica de usuario

CONTENIDO

Unidad 1. Introducción a la POO
1.1. Paradigmas de la programación
1.2. Características de la POO
1.3. Conceptos de Programación Orientada a Objetos
1.3.1. Abstracción
1.3.2. Encapsulamiento
1.3.3. Polimorfismo
1.3.4. Herencia
1.3.5. Modularidad
1.4. Clases y Objetos
1.5. Mensajes y Métodos

Unidad 2. Programación estructurada - Introducción a Java
2.1 Un recorrido rápido por Java
2.2 Sintaxis básica en Java
2.3 Comentarios, identificadores y palabras reservadas

2.4 Tipos datos primitivos
2.5 Variables y constantes
2.6 Operadores
2.7 Sentencias de control de flujo de ejecución
2.8 Enumerados
2.9 Arrays (Vectores, Matrices)

Unidad 3. Programación basada en objetos
3.1 Clases y Objetos (Estado, comportamiento e identidad)
3.2 Abstracción de objetos – Creación de Clases
3.3 Visibilidad de Clases y Objetos
3.3.1 Vista pública de las Clases y objetos
3.3.4 Vista privada de las Clase y objetos
3.4 Atributos y Métodos
3.4.1 Miembros de Instancia y de Clase
3.4.2 Método constructor
3.4.3 Métodos Getter y Setters (encapsulamiento)
3.4.4 Sobrecarga de métodos
3.4.4 Funciones miembro

Unidad 4. UML Básico y Relaciones entre clases
4.1 Introducción a UML
4.2 Vistas UML
4.3 Diagramas de casos de uso
4.4 Diagramas de clases
4.4.1 Representación de clases en UML
4.4.2 Relación de asociación
4.4.3 Relación de agregación
4.4.4 Relación de composición
4.4.5 Relación de herencia (Especialización/Generalización)

Unidad 5. Herencia, Polimorfismo e Interfaces
5.1 Herencia Simple
5.2 Herencia Múltiple
5.3 Polimorfismo
5.3.1 Clases abstractas
5.3.2 Métodos abstractos
5.4 Construcción e implementación de interfaces
5.5 Herencia Múltiple con Interfaces

Unidad 6. Programación con excepciones
6.1 Excepciones

6.2 Manejo de excepciones (bloque try – catch - finally)
6.2.1 Excepciones verificadas y no verificadas
6.2.2 Lanzar excepciones (clausula throws)
6.3 Creación de excepciones personalizadas

Unidad 7. Colecciones de Objetos
7.1 Arrays de objetos
7.2 Clase ArrayList
7.3 ArrayList de Objetos
7.3.1 Operaciones sobre una colección de objetos
7.3.2 Adicionar objetos
7.3.3 Buscar objetos
7.3.4 Ordenar Colecciones
7.3.5 Remover objetos

Unidad 8. Programación modular – Modelo tres capas
8.1 Concepto de arquitectura de software
8.2 Arquitectura tres capas
8.3 Capa de vista/presentación
8.4 Capa de modelo/negocio
8.5 Capa de datos

Unidad 9. Persistencia de datos
9.1. Archivos binarios
9.2. Archivos de texto
9.3. Serialización (objetos)
9.4 Operaciones CRUD sobre archivos

Unidad 10. Interfaz gráfica de usuario - GUI
10.1 Componentes gráficos básicos
10.2 Manejo de layouts
10.3 Gestión de eventos
10.4 Arquitectura de la interfaz gráfica

EVALUACIÓN

La evaluación de la asignatura se basa en medir el aprendizaje del alumno respecto al contenido presentado. Las herramientas utilizadas para la realización de esta medida serán las siguientes:

Control del desarrollo del proceso educativo: Se propone un control continuo en el proceso docente educativo como ayuda al proceso de asimilación del conocimiento dado a

los alumnos. Se pueden hacer preguntas, talleres para la casa, pero sin notas como herramienta diagnostica.

Evaluación Parcial: Se sugiere 3 (tres) evaluaciones parciales. Las unidades 1 a 3 para el primer previo (valorado con un 30%, distribuidos así: 10% prueba teórica y 20% practica), Las unidades 4-7 para el segundo previo (valorado con 30%, distribuidos en prueba teórica con 10% de valoración y prueba práctica con 20%). Las unidades 8-9 para el tercer previo (valorada con 40%, prueba práctica con 10% de valoración, prueba teórica con 10% de valoración, y proyecto final de aula con 20%).

REFERENCIAS BIBLIOGRÁFICAS

Biblioteca:

- Cernura, A**.** *Análisis y diseño estructurado y orientado a objetos de sistemas informáticos*, Addison-Wesley (1996).
- Joyanes, L. *Programación orientada a objetos,* McGraw Hill (1998).
- Ceballos, F. *Java 2 interfaces gráficas y aplicaciones para internet*, Grupo AlfaOmega (2006).
- Joyanes, L. *Programación en Java 2: Algoritmos, estructuras de datos y programación orientada a objetos*, McGraw Hill (2002).
- Fuente. P. *UML para programadores java*, Pearson Educación (2009)
- Herbert, James. *El arte de programar en java*, McGraw Hill (2004).
- Joyanes, L. Fundamentos de Programación: Algoritmos, estructura de datos y objetos, McGraw Hill (2008)
- BRENTA, B. Programación orientada a objetos con java, PEARSON (2007).
- Bennet, S. Análisis y diseño orientado a objetos de sistemas usando UML, McGraw Hill (2007).

- Deitel y Deitel. Como programar en Java, Prentice Hall Hispanoamericana (1998).
- Barnes,D, Kolling, M. Programacion orientada a objetos con java usando BlueJ. Pearson. 6ª Ed.

Base de datos digitales:

e-libro

Jaramillo Valbuena, Sonia, Cardona Torres, Sergio Augusto, and Hernández Rodríguez, Leonardo Alonso. Programación orientada a objetos. Bogotá, CO: Ediciones Elizcom, 2010. ProQuest ebrary. Web. 14 June 2017.

Flórez Fernández, Héctor Arturo. Programación orientada a objetos usando java. Bogotá, CO: Ecoe Ediciones, 2012. ProQuest ebrary. Web. 14 June 2017.

García Llinás, Luis Fernando. Todo lo básico que debería saber: sobre programación orientada a objetos en Java. Bogotá, CO: Ediciones de la U, 2010. ProQuest ebrary. Web. 14 June 2017.

Vélez Serrano, José, Peña Abril, Alberto, and Gortazar Bellas, Patxi. Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java. Madrid, ES: Dykinson, 2011. ProQuest ebrary. Web. 14 June 2017.

Moreno Pérez, Juan Carlos. Programación. Madrid, ES: RA-MA Editorial, 2014. ProQuest ebrary. Web. 14 June 2017.

Oviedo, Regino, Efraín. Lógica de programación orientada a objetos, Ecoe Ediciones, 2015. ProQuest Ebook Central.