1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura:	**Programación Orientada a Objetos**
Carrera:	**ISC**
Clave de la asignatura:	**SCD1020**
(Créditos) SATCA[1]	**2-3-5**

2.- PRESENTACIÓN

Caracterización de la asignatura.

Esta asignatura aporta al perfil del Ingeniero en Sistemas Computacionales la capacidad de analizar, desarrollar, implementar y administrar software de aplicación orientado a objetos, cumpliendo con estándares de calidad, con el fin de apoyar la productividad y competitividad de las organizaciones.
Esta materia proporciona soporte a otras, más directamente vinculadas con desempeños profesionales; se ubica en el segundo semestre de la trayectoria escolar. Proporciona al estudiante las competencias necesarias para abordar el estudio de cualquier lenguaje orientado a objetos, metodología de análisis y diseño orientado a objetos, de los sistemas gestores de bases de datos, y en general de cualquier materia basada en el modelo orientado a objetos.

Intención didáctica.

El enfoque sugerido para la materia requiere que las actividades prácticas promuevan el desarrollo de habilidades para la resolución de problemas, tales como:
identificación, manejo, control de variables, datos relevantes, planteamiento de hipótesis, trabajo en equipo, asimismo, propicien procesos intelectuales como inducción-deducción y análisis-síntesis con la intención de generar una actividad intelectual compleja; las actividades teóricas se han descrito como actividades previas al tratamiento practico de los temas. En las actividades prácticas sugeridas, es conveniente que el profesor sólo guie al estudiante en la construcción de su conocimiento.

[1] Sistema de asignación y transferencia de créditos académicos

En la primera unidad se presentan los conceptos de la programación orientada a objetos, teniendo la intención de introducir al estudiante en los elementos del modelo de objetos, así como el uso básico del lenguaje de modelado unificado.

La segunda unidad se centra en la introducción al lenguaje C# y su plataforma para la implementación de aplicaciones en un entorno visual.

La tercera unidad tiene como propósito el diseño de clases y la creación de objetos que incorporen propiedades y métodos de otros objetos, construyéndolos a partir de éstos sin necesidad de reescribirlo todo.

La cuarta unidad aborda las relaciones entre clases (herencia, composición y agregación) con la finalidad de reutilizar el código y diseñar aplicaciones cuyos objetos estén embebidos dentro de otros.

La quinta unidad trata una de las características fundamentales de la programación orientada a objetos: polimorfismo, que permite reutilizar métodos con el mismo nombre, pero con relación a la clase a la que pertenece cada uno, con comportamientos diferentes.

En la sexta unidad el estudiante adquirirá los conocimientos para tratar situaciones excepcionales que se presentan en tiempo de ejecución.

La unidad siete introduce al estudiante el concepto de delegados y métodos anónimos como prerrequisito para abordar el tema de eventos.

En la unidad ocho, el estudiante aborda el tema de eventos para implementar aplicaciones que disparen notificaciones automáticas.

En la unidad nueve, el estudiante aplica las operaciones necesarias para el manejo de archivos de texto y binarios, temas que se utilizarán en materias posteriores.

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**
Diseñar e implementar objetos de programación que permitan resolver situaciones reales y de ingeniería.	**Competencias instrumentales** • Capacidad de análisis y síntesis • Capacidad de organizar y planificar • Comunicación oral y escrita • Habilidad para buscar y analizar • información proveniente de fuentes diversas.

	• Solución de problemas. • Toma de decisiones. **Competencias interpersonales** • Capacidad crítica y autocrítica • Habilidades interpersonales **Competencias sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad). • Habilidad para trabajar en forma autónoma. • Búsqueda del logro

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	**Participantes**	**Observaciones (cambios y justificación)**
Instituto Tecnológico de Nuevo Laredo. Fecha: 9 de junio del 2016.	Ing. Gloria María Rodríguez Morales, Ing. Verónica Dávila Toscano, Ing. Bertha Alicia Fernández De la Mora, Ing. Thelma Gpe. Cantú Treviño, Ing. Bruno López Takeyas, Ing. Jaime David Johnston Barrientos.	Análisis, enriquecimiento y elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la carrera de ISC

5.- OBJETIVO(S) GENERAL(ES) DEL CURSO (competencias específicas a desarrollar en el curso)

Diseñar e implementar objetos de programación que permitan resolver situaciones reales y de ingeniería.

6.- COMPETENCIAS PREVIAS

Analizar, diseñar y desarrollar soluciones de problemas reales utilizando algoritmos computacionales para implementarlos en un lenguaje de programación orientado a objetos.

7.- TEMARIO

Unidad	Temas	Subtemas
1	Introducción al paradigma de la Programación Orientada a Objetos	1. Introducción a la POO 2. Elementos del modelo de objetos *Clases* *Objetos* *Abstracción* *Encapsulamiento* *Mensajes* *Constructor y destructor* *Herencia* *Clases abstractas* *Sobrescritura* *Sobrecarga* *Polimorfismo* 3. Modelado de objetos reales en UML 4. Características de los objetos *Estado* *Comportamiento* *Identidad* 5. Atributos, métodos y propiedades 6. Diseño de diagramas de clases mediante software
2	Introducción a C# y al diseño de formas	1. Introducción a la programación en C# 2. El framework y sus componentes 3. Uso de Microsoft Visual Studio 4. El CLR (Common Language Runtime) 5. Librerías de clases 6. Estructura de una aplicación visual en C# 7. Diseño de aplicaciones visuales mediante Visual Studio 8. Los controles visuales de una aplicación en C# *Propiedades Name y Text* *Form* *TextBox* *Button* *MessageBox* *CheckBox* *RadioButton*

		ComboBox *ListBox* *DataGridView* *PictureBox* *Selección y uso correcto de los controles visuales* 9. Nomenclatura sugerida para identificar los componentes de una aplicación
3	Clases y objetos	1. Clases 2. Objetos 3. Instanciación de una clase 4. Componentes de una clase *Atributos* *Métodos* *Propiedades* *Delegados* *Eventos* 5. Modificadores de acceso *Público* *Privado* *Protegido* 6. Componentes estáticos vs componentes de instancia 7. Diagramas UML 8. Diseño de un proyecto con formas agregando clases 9. Declaración de clases 10. Creación de objetos 11. Métodos 12. Mutators y accesors 13. Constructor y destructor 14. Diagramas de flujo con métodos de clases 15. Referencia al objeto actual (this) 16. Propiedades *Accesadores get y set* *De solo lectura* *Con nivel asimétrico* *Autoimplementadas* 17. Métodos *Firma* *Sobrecarga* *Contructores* *Destructor* 18. Destrucción de objetos 19. Diagramas de clases en UML 20. Diseño de proyectos de POO en C# *Diagramas de clases usando software* *Diagramas de flujo de métodos usando software* *Codificación de aplicaciones en C#*
4	Relaciones (herencia, composición y agregación)	1. Herencia *Conceptos* *Ejemplos de la vida cotidiana* *Ejemplos de herencia con varios niveles* *Uso* 2. Clase base 3. Clase derivada 4. Clasificación de herencia *Simple*

		Múltiple 5. Representación de herencia en UML 6. Diagramas de UML con herencia 7. Llamando métodos de la clase base 8. Secuencia de ejecución de constructores y destructores en herencia 9. Ejercicios llamando el constructor de la clase base 10. Sobrescritura del método ToString() 11. Clases selladas 12. Clases parametrizadas o genéricas 13. Colecciones genéricas en C# 14. Clase genérica List 15. Principales métodos y propiedades de la clase List 16. El iterador GetEnumerator y uso del ciclo foreach 17. Creación de un objeto con una lista genérica 18. Diseño e implementación de aplicaciones visuales con herencia 19. Composición *Grados de dependencia (cardinalidad o multiplicidad)* *Representación de la composición* *Reglas para que exista composición* *Composición 1..1* *Composición 1..** *Recorrido de las partes de una composición* *Diseño e implementación de iteradores* *Modelo UML con varias composiciones* *Diseño e implementación de una clase con varios iteradores* *Implementación de iteradores mediante propiedades* *Diseño e implementación de aplicaciones visuales con composición* 20. Agregación *Representación de la agregación* *Reglas para que exista agregación* *Agregación 1..1* *Agregación 1..** *Modelo UML con varias agregaciones* *Diseño e implementación de aplicaciones visuales con agregación* 21. Composición vs agregación
5	Polimorfismo	1. Concepto 2. Tipos de polimorfismo *Paramétrico* *De sobrecarga* *De subtipo* *Ejemplos* 3. Métodos virtuales, sobrescritos y sellados 4. Ocultar métodos heredados 5. Clases abstractas 6. Componentes abstractos 7. Interfaces 8. Diseño e implementación 9. Uso de interfaces 10. La interfase IComparable

		11. La interfase IEquatable 12. Diseño e implementación de aplicaciones visuales polimórficas con clases abstractas e interfaces
6	Excepciones	1. Concepto 2. Control de excepciones 3. Tipos de excepciones 4. Excepciones en C# 5. Manejo de excepciones en C# 6. Tratamiento de desborde aritmético *7.* La sentencia *checked* *8.* La sentencia *throw* 9. Uso de propiedades para validar la captura de datos 10. Diseño e implementación de aplicaciones visuales con excepciones
7	Delegados	1. Conceptos y uso 2. Declaración de variables de tipo delegado 3. Ejemplos de uso 4. Métodos anónimos 5. Uso de métodos anónimos 6. Expresiones lambda 7. Ejemplos de uso de expresiones lambda 8. Uso de delegados y expresiones lambda 9. Diseño e implementación de aplicaciones visuales con delegados
8	Eventos	1. Conceptos 2. Uso de eventos 3. Diseño de eventos 4. La clase publicadora 5. Disparo de eventos 6. Suscripción a eventos 7. La clase suscriptora 8. El método gestor 9. Cancelar la suscripción a un evento 10. La interfase INotifyPropertyChanged 11. Envío de correos electrónicos desde una aplicación en C# 12. La clase MailMessage 13. Diseño e implementación de aplicaciones visuales con eventos
9	Flujos y archivos	1. Introducción a los archivos 2. Relación entre la memoria principal y la memoria secundaria 3. Conceptos de archivos 4. Tipos de archivos *Texto* *Binarios* *Secuenciales* *Relativos o de acceso directo* 5. Analogías entre un archivo digital y un archivero 6. Definición del nombre de un archivo 7. Extensiones de los nombres de los archivos 8. Introducción a los flujos 9. Operaciones internas con archivos 10. Apertura de archivos

		11. Pasos para grabar y leer datos en un archivo 12. Espacios de nombres necesarios para diseñar aplicaciones con archivos 13. Clase para el manejo de archivos *FileStream* *StreamWriter* *StreamReader* *File* 14. Modos de apertura de archivos 15. Modos de acceso de archivos 16. Cómo detectar la existencia de un archivo 17. La propiedad EndOfStream 18. Cierre de archivos 19. Serialización 20. Uso de la serialización 21. Tipos de serialización 22. Cómo preparar una clase para serializar sus objetos 23. Cómo grabar y leer un objeto serializado en un archivo 24. Diseño e implementación de aplicaciones visuales con archivos

8.-SUGERENCIAS DIDÁCTICAS (desarrollo de competencias genéricas)

- Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes.
- Propiciar el uso de las nuevas tecnologías en el desarrollo de los contenidos de la asignatura.
- Propiciar la planeación y organización del proceso de programación orientada a objetos en la construcción de nuevos conocimientos.
- Fomentar actividades grupales que propicien la comunicación, el intercambio argumentado de ideas, la reflexión, la integración, la colaboración de y entre los estudiantes.
- Propiciar el desarrollo de capacidades intelectuales relacionadas con la lectura, la escritura y la expresión oral.
- Propiciar en el estudiante el desarrollo de actividades intelectuales de inducción-deducción y análisis-síntesis, las cuales lo encaminan hacia la investigación, la aplicación de conocimientos y la solución de problemas.
- Relacionar los contenidos de esta asignatura con las demás del plan de estudios a las que ésta da soporte para desarrollar una visión interdisciplinaria en el estudiante.
- Proponer problemas que permitan al estudiante la integración de contenidos de la asignatura y entre distintas asignaturas, para su análisis y solución.
- Relacionar los contenidos de la asignatura con el respeto al marco legal, el cuidado del medio ambiente y con las prácticas de una ingeniería con enfoque sustentable.

9.- SUGERENCIAS DE EVALUACIÓN

La evaluación debe ser continua y formativa por lo que se debe considerar el desempeño de cada una de las actividades de aprendizaje, haciendo especial énfasis en:

- Información obtenida durante las investigaciones solicitadas, plasmadas en documentos escritos o digitales
- Solución algorítmica a problemas reales o de ingeniería utilizando el diseño escrito o en herramientas digitales
- Codificación en un lenguaje de programación orientada a objeto de las soluciones diseñadas
- Participación y desempeño en el aula y laboratorio
- Dar seguimiento al desempeño en el desarrollo del temario (dominio de los conceptos, capacidad de la aplicación de los conocimientos en problemas reales y de ingeniería)
- Se recomienda utilizar varias técnicas de evaluación con un criterio específico para cada una de ellas (teórico-práctico).
- Desarrollo de un proyecto por unidad que integre los tópicos vistos en la misma
- Desarrollo de un proyecto final que integre todas las unidades de aprendizaje
- Uso de una plataforma educativa en internet la cual puede utilizarse como apoyo para crear el portafolio de evidencias del alumno (integrando: tareas, prácticas, evaluaciones, etc.)

10.- UNIDADES DE APRENDIZAJE

Unidad 1: Introducción al paradigma de la Programación Orientada a Objetos

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Dominar los conceptos relacionados con el paradigma orientado a objetos para resolver problemas	• Investigar el paradigma orientado a objetos • Realizar un mapa conceptual sobre los términos del paradigma orientado a objetos

Unidad 2: Introducción a C# y al diseño de formas

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Conocer el lenguaje de programación C#, su entorno y sus	• Revisar el entorno de programación visual en C#

herramientas para implementar aplicaciones visuales Conocer las características principales del lenguaje de programación orientado a objetos. Codificar algoritmos en un lenguaje de programación orientado a objetos. Compilar y ejecutar programas.	• Dominar los controles visuales para desarrollar aplicaciones en C# • Codificar y ejecutar aplicaciones visuales en C#

Unidad 3: Clases y objetos

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Diseñar clases e implementar objetos cumpliendo las reglas de la programación orientada a objetos	• Programar clases con atributos públicos para exponer y comprender la vulnerabilidad de los datos. • Proteger los atributos con modificadores de acceso privados o protegidos y programar métodos públicos para otorgar acceso seguro a los mismos. • Reunir dentro de una clase los miembros necesarios para resolver un problema en particular, y así implementar el encapsulamiento. • Instanciar objetos para identificar el nacimiento y muerte de los mismos. • Programar constructores y destructores para las clases, de manera que permitan dar un valor inicial a sus atributos cuando nazcan sus objetos, o liberar recursos cuando mueran los mismos. • Identificar los comportamientos de una clase que pueden variar dependiendo del paso, cantidad, tipo u orden de argumentos. • Programar cada variación del comportamiento en métodos sobrecargados para agregar flexibilidad a la clase.

	• Identificar operaciones que puedan ser realizadas entre dos objetos de la misma clase.

Unidad 4: Relaciones (herencia, composición y agregación)

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Analizar las principales relaciones entre clases para diseñar aplicaciones orientadas a objetos que reutilicen y optimicen el código	• Analizar analogías taxonómicas de los seres vivos que compartan rasgos comunes por estar relacionados mediante una herencia genética e identificar la especie a la que pertenecen. • Analizar objetos reales que compartan características comunes por pertenecer a una misma categoría de objetos. • Identificar los atributos y comportamientos propios de una categoría de objetos que compartan todos sus miembros. • Investigar en fuentes de información los conceptos relacionados con la herencia y su implementación en un lenguaje de programación orientado a objetos. • Programar una clase base para una especie de animales con los atributos y comportamientos comunes a todos los animales pertenecientes a ella. • Implementar clases derivadas para animales pertenecientes a la misma especie de la cual se programó la clase base anteriormente. • Especializar cada clase derivada con comportamientos y atributos específicos de un tipo de animal para identificarlo y distinguirlo de los demás. • Crear varias instancias de clases derivadas diferentes para verificar la existencia de los miembros heredados comunes en todas ellas, y la diversidad de sus especializaciones.

	• Repetir las mismas actividades, pero utilizando objetos y categorías de objetos reales. • Sobrecargar los constructores de las clases base y derivadas para analizar y experimentar el comportamiento y uso de los constructores en combinación con la herencia. • Analizar el comportamiento de objetos creados a partir de otros mediante relaciones de composición y agregación

Unidad 5: Polimorfismo

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Implementar interfaces y clases polimórficas.	• Analizar clases base que no requieran ser instanciadas, o que carezcan de sentido para ello por ser abstractas. • Investigar en fuentes de información los conceptos y reglas para implementar clases abstractas en un lenguaje de programación orientado a objetos. • Implementar clases abstractas en clases base que no requieran ser instanciadas con al menos un método abstracto para que sea implementado por sus clases derivadas en múltiples formas. • Implementar una clase con todos sus comportamientos abstractos. Investigar en diversas fuentes de información el concepto de interfaz y compararlo con la clase cien por ciento abstracta. • Programar interfaces para definir los comportamientos que una clase deberá de tener al implementarla. • Implementar una misma interfaz en diferentes clases para dar en cada una un comportamiento diferente a sus métodos. • Realizar una herencia de interfaces para especializar los comportamientos que las clases podrán implementar.

Unidad 6: Excepciones

Competencia específica a desarrollar	Actividades de Aprendizaje
Identificar, manejar, gestionar y crear las condiciones de error que interrumpan el flujo normal de ejecución de un programa.	• Crear un programa que deliberadamente genere excepciones comunes para identificar: sus nombres, sus causas, su comportamiento, y reporte de error. • Programar una clase con varios métodos invocándose en cadena, donde el último método genere una excepción para estudiar y comprender la propagación de las mismas. • Utilizar la selectiva intenta para atrapar excepciones de diferentes tipos, y prevenir la interrupción de ejecución de un programa. • Analizar situaciones en las que un método no pueda devolver un valor de retorno como indicador de un error interno, y tenga la necesidad de levantar una excepción por el usuario que le indique que su función no pudo ser realizada. • Programar la instanciación y lanzamiento de excepciones definidas por el lenguaje para situaciones en que no es posible regresar un valor desde un método que indique una condición de error interna. • Identificar condiciones de error requeridas por el usuario y no previstas por el lenguaje que requieran la creación de un nuevo tipo de excepción. • Implementar un nuevo tipo de excepción definido por el usuario heredando de la clase base de las excepciones o alguna otra ya definida por el lenguaje que más se aproxime al comportamiento deseado del usuario. • Programar y experimentar el lanzamiento, propagación y manejo de una excepción definida por el usuario.

Unidad 7: Delegados

Competencia específica a desarrollar	Actividades de Aprendizaje
Conocer los delegados y métodos anónimos como una estrategia para enviar métodos a otros métodos y aumentar su funcionalidad	• Analizar los conceptos relacionados con delegados • Revisar el funcionamiento de los métodos anónimos • Implementar aplicaciones orientadas a objetos donde se envíen métodos a otros métodos para incrementar su funcionalidad • Revisar el concepto de delegados como prerrequisito para abordar el tema de eventos

Unidad 8: Eventos

Competencia específica a desarrollar	Actividades de Aprendizaje
Implementar aplicaciones orientadas a objetos capaces de enviar notificaciones automáticas	• Analizar el funcionamiento de las clases publicadoras y clases suscriptoras • Revisar el concepto de eventos y sus requisitos de diseño e implementación • Implementar aplicaciones orientadas a objetos donde se disparen notificaciones automáticas

Unidad 9: Flujos y archivos

Competencia específica a desarrollar	Actividades de Aprendizaje
Implementar aplicaciones orientadas a objetos que creen y manipulen archivos para guardar y recuperar información.	• Investigar en fuentes de información los conceptos y metodologías para manipular archivos de texto y binarios en un lenguaje de programación orientado a objetos. • Programar una clase que cree, consulte, modifique y borre archivos de texto. • Programar una clase que cree, consulte, modifique y borre archivos binarios.

	• Diseñar un caso de estudio que requiera el uso de archivos para que sea resuelto por el alumno.

11.- FUENTES DE INFORMACIÓN

1. López Takeyas, Bruno. (2016). Curso de programación orientada a objetos en C# .NET. Ejemplos con aplicaciones visuales y de consola. Editorial Alfaomega.
2. López Takeyas, Bruno. (2016). Programación orientada a objetos en C#. 9 junio 2016, de Instituto Tecnológico de Nuevo Laredo. Sitio web: https://nlaredo.tecnm.mx/takeyas/Materias/POO/index.htm

12.- PRÁCTICAS PROPUESTAS

- Investigar la metodología para resolver problemas a través de la computadora.
- Analizar, diseñar e implementar aplicaciones orientadas a objetos
- Solucionar problemas con algoritmos a partir de enunciados proporcionados por el profesor.
- Crear, compilar y ejecutar programas de consola en C# mediante Microsoft Visual Studio.
- Las prácticas puedes accederse y descargarse de https://nlaredo.tecnm.mx/takeyas/Apuntes/POO/Practicas/index.htm