1.- DATOS DE LA ASIGNATURA.

Nombre de la asignatura:	Programación Orientada a Objetos.
Carrera:	Ingeniería en Sistemas Computacionales.
Clave de la asignatura:	SCD-1020
(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.

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 definición e implementación de clases y objetos permitiendo al estudiante adquirir las competencias fundamentales de la

[1] Sistema de asignación y transferencia de créditos académicos

programación orientada a objetos.

La tercera unidad tiene como propósito 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 trata una de las características fundamentales de la programación orientada a objetos; el 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 quinta unidad el estudiante adquirirá los conocimientos para tratar situaciones excepcionales que se presentan en tiempo de ejecución.

La unidad seis, 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 • Trabajo en equipo • 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 de logro.

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	Participantes	Observaciones (cambios y justificación)
Instituto Tecnológico de Saltillo.	Representantes de los Institutos Tecnológicos de:	Reunión nacional de Diseño e innovación curricular de la carrera de Ingeniería en
Fecha del 12 de Octubre de 2009 al 19 de Febrero del 2010. Institutos Tecnológicos de: La Laguna, Mérida, Ciudad Cuauhtémoc, Mexicali, Tijuana, Parral, Villahermosa, Istmo y Matamoros. Institutos Tecnológicos Superiores de: Lerdo, Coatzacoalcos, Occidente del Estado de Hidalgo, Sur de Guanajuato, Tepexi de Rodríguez.	Representante de la Academia de Ciencias Básicas	Análisis, enriquecimiento y elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la carrera de
Instituto Tecnológico de fecha	Representantes de los Institutos Tecnológicos participantes en el diseño de la carrera de Ingeniería	Reunión nacional de consolidación de la carrea de ingeniería en

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 orientado a objetos.	1.1 Elementos del modelo de objetos: clases, objetos, abstracción, modularidad, encapsulamiento, herencia y polimorfismo. 1.2 Lenguaje de modelado unificado: diagrama de clases.
2	Clases y objetos.	2.1 Declaración de clases: atributos, métodos, encapsulamiento. 2.2 Instanciación de una clase. 2.3 Referencia al objeto actual. 2.4 Métodos: declaración, mensajes, paso de parámetros, retorno de valores. 2.5 Constructores y destructores: declaración, uso y aplicaciones. 2.6 Sobrecarga de métodos. 2.7 Sobrecarga de operadores: Concepto y utilidad, operadores unarios y binarios.
3	Herencia.	3.1 Definición: clase base, clase derivada. 3.2 Clasificación. herencia simple, herencia múltiple. 3.3 Reutilización de miembros heredados. 3.4 Referencia al objeto de la clase base. 3.5 Constructores y destructores en clases derivadas. 3.6 Redefinición de métodos en clases derivadas.
4	Polimorfismo.	4.1 Definición. 4.2 Clases abstractas: definición, métodos abstractos, implementación de clases abstractas, modelado de clases abstractas. 4.3 Interfaces:

		definición, implementación de interfaces, herencia de interfaces. 4.4 Variables polimórficas (plantillas): definición, uso y aplicaciones. 4.5 Reutilización de código.
5	Excepciones.	5.1 Definición. 5.2 Tipos de excepciones. 5.3 Propagación de excepciones. 5.4 Gestión de excepciones: manejo de excepciones, lanzamiento de excepciones. 5.5 Creación y manejo de excepciones definidas por el usuario.
6	Flujos y Archivos.	6.1 Definición. 6.2 Clasificación: Archivos de texto y binarios. 6.3 Operaciones básicas y tipos de acceso. 6.4 Manejo de objetos persistentes.

8.- SUGERENCIAS DIDÁCTICAS.

- 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 orientado a objetos.

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Comprender, describir y modelar los conceptos principales del paradigma de programación orientado a objetos y aplicarlos a situaciones de la vida real.	• Investigar y seleccionar en diversas fuentes de información los conceptos principales del paradigma de programación orientado a objetos. • Identificar ejemplos de la vida real que apliquen o manifiesten dichos conceptos. • Redactar una definición propia de los conceptos de forma simple y entendible. • Comentar en clase las definiciones de

	otros compañeros para enriquecer la propia y consensar una grupal. • Desarrollar un mapa conceptual entre los distintos paradigmas señalando sus ventajas y desventajas. • Analizar la información del lenguaje UML referente al modelado de clases. • Diseñar diagramas de clases aplicados a distintos problemas.

Unidad 2: Clases y objetos.

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Implementar clases y objetos cumpliendo las reglas de la programación orientada a objetos. Implementar constructores y destructores para inicializar atributos y liberar recursos. Sobrecargar métodos y operadores para optimizar el código de una clase.	• 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. Seleccionar un operador existente del lenguaje y sobrecargarlo en la clase de los objetos para implementarles dicha funcionalidad.

Unidad 3: Herencia.

Competencia específica a desarrollar	Actividades de Aprendizaje
Implementar la herencia en clases derivadas para reutilizar los miembros de una clase base.	• 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. • Identificar los atributos y comportamientos propios de una especie que comparten los animales pertenecientes a ella. • 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 qué animales u objetos de la vida real rompen algún comportamiento heredado para reinventar el suyo propio por sobre el resto de sus parientes que siguen respetando el heredado. • Redefinir un método en una clase derivada para sobrescribir el de su clase base e introducirse al polimorfismo.

Unidad 4: 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. • Declarar variables miembro de tipo clase abstracta o interfaz para que en tiempo de ejecución se inicialice con diferentes subtipos o implementaciones de las mismas, y se demuestre así, toda la flexibilidad del polimorfismo al cambiar el

	comportamiento de un objeto en tiempo de ejecución.

Unidad 5: 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 lanzamineto 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 6: 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. Taylor David. Object Orient informations systems, planning and implementations. Canada: Wiley. 1992.
2. Larman Craig. UML y patrones introducción al análisis y diseño orientado a objetos. México: Pretince Hall. 1999.
3. Winblad, Ann L. Edwards, Samuel R. Software orientado a objetos. USA: Addison. Wesley/ Díaz Santos. 1993.
4. Fco. Javier Ceballos. Java 2 Curso de Programación. Alfaomega.
5. Agustín Froufe. Java 2 Manual de usuario y tutorial. Alfaomega.
6. Laura Lemay, Rogers Cadenhead. Aprendiendo JAVA 2 en 21 días. Prentice Hall.
7. Herbert Schildt. Fundamentos de Programación en Java 2. McGrawHil.
8. J Deitel y Deitel. Como programar en Java. Prentice Hall.
9. Stephen R. Davis. Aprenda Java Ya. McGrawHill.
10. Kris Jamsa Ph D. ¡Java Ahora!. McGrawHill.
11. Francisco Charte Ojeda. Visual C# .NET. ANAYA MULTIMEDIA
12. Kingsley-Hughes, Kathie; Kingsley-Hughes, Adrian. C# 2005. ANAYAMULTIMEDIA
13. Ceballos Francisco Javier.Enciclopedia de Microsoft Visual C#. 2ª Edición
14. El lenguaje de programación C#.Fco. Javier Ceballos Sierra. Editorial Ra-ma.
15. Tom Archer. A fondo C#. McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S.A.U.

12.- PRÁCTICAS PROPUESTAS.

1. Crear un programa que instancie y use un objeto predefinido por el lenguaje para practicar el envío de mensajes, el uso de parámetros y la recepción de su respuesta. Sugerencia: objeto de clase String.
2. Analizar objetos concretos (puerta, elevador, televisor, etc.) y abstractos (cuenta bancaria, préstamo, viaje, etc.) de la vida real para abstraer y modelar sus atributos y comportamientos. Implementar clases para instanciar objetos que modelen sus contrapartes de la vida real usando tipos de datos simples y objetos como parámetros y valores de retorno, así como métodos sin valores de retorno.
3. Intercambiar clases de objetos entre compañeros para usar sus miembros con valores o situaciones erróneas que evidencien la necesidad de protegerlos con modificadores de acceso. Modificar el código fuente aplicando los distintos niveles de acceso para experimentar y descubrir (aprender) el impacto de cada uno de ellos.
4. Implementar la clase Persona con los atributos nombre y edad; un constructor, un destructor, y al menos el método crecer para mapear el ciclo de vida de una persona con el de un objeto.
5. Implementar la clase Calculadora que realice al menos las cuatro operaciones básicas de la aritmética sobrecargando métodos para cada tipo de dato numérico del lenguaje de los parámetros.
6. Implementar la clase Matriz que sobrecargue los operadores +, -, * y / para este tipo de dato definido por el usuario.
7. Programar una aplicación sobre figuras geométricas que implemente la clase base FiguraGeometrica de la cual hereden sus miembros las clases derivadas y que éstas solo especialicen sus características o comportamientos.
8. Implementar constructores y destructores a las clases base y derivadas de la aplicación sobre figuras geométricas para experimentar y comprender su funcionamiento cuando está implicada la herencia.
9. Modificar la clase FiguraGeometrica para convertirla en abstracta y programar al menos un método abstracto que todas las clases derivadas deberán implementar con su propio comportamiento.
10. Programar la interfaz Vehiculo con un conjunto de métodos abstractos que todo vehículo de la vida real debería tener. Programar varias clases que implementen la interfaz anterior y definan el comportamiento particular de sus métodos.
11. Especializar la interfaz Vehiculo en al menos dos subinterfaces (VehiculoTerreste o VehiculoAereo) que agreguen comportamientos abstractos que las clases deberán implementar.
12. Programar clases que generen excepciones comunes como referencias nulas o desbordamientos numéricos para estudiar su naturaleza, comportamiento, prevención y lanzamiento.
13. Implementar aplicaciones que almacenen y recuperen información de diferentes tipos de datos simples a través de un archivo de texto para persistir información.
14. Programar una clase que tome un objeto de cierto tipo y lo persista en un archivo de texto para ser recuperado posteriormente restableciendo el estado que tenía antes de ser persistido (serializarlo).