



## 1. Datos Generales de la asignatura

<b>Nombre de la asignatura:</b>	Programación Orientada a Objetos
<b>Clave de la asignatura:</b>	AED-1286
<b>SATCA<sup>1</sup>:</b>	2-3-5
<b>Carrera:</b>	Ingeniería en Sistemas Computacionales, Ingeniería en Animación Digital y Efectos Visuales e Ingeniería en Desarrollo de Aplicaciones.

## 2. Presentación

### Caracterización de la asignatura

Esta asignatura aporta al perfil del Ingeniero 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 es muy importante porque 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.

Para cursarla se requiere de los conocimientos y habilidades adquiridas en Fundamentos de la Programación.

### Intención didáctica

El enfoque sugerido para la asignatura 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 práctico de los temas. En las actividades prácticas sugeridas, es conveniente que el profesor sólo guíe al estudiante en la construcción de su conocimiento.

En el primer tema 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.

El segundo tema se centra en la definición e implementación de clases y objetos permitiendo al estudiante adquirir las competencias fundamentales de la programación orientada a objetos.

El tercer tema 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.

<sup>1</sup> Sistema de Asignación y Transferencia de Créditos Académicos



El cuarto tema 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 el quinto tema se tratan situaciones excepcionales que se presentan en tiempo de ejecución.

En el tema seis, se aplica las operaciones necesarias para el manejo de archivos de texto y binarios, temas que se utilizarán en materias posteriores.

### 3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Instituto Tecnológico de Aguascalientes del 15 al 18 de junio de 2010.	Representantes de los Institutos Tecnológicos de: Centro Interdisciplinario de Investigación y Docencia en Educación Técnica, Acapulco, Aguascalientes, Apizaco, Boca Río, Celaya, Chetumal, Chihuahua, Chilpancingo, Chiná, Cd. Cuauhtémoc, Cd. Juárez, Cd. Madero, Cd. Victoria, Colima, Comitán, Cuautla, Durango, El Llano Aguascalientes, Huixquilucan, Valle Bravo, Guaymas, Huatabampo, Huejutla, Iguala, La Laguna, La Paz, La Zona Maya, León, Lerma, Linares, Los Mochis, Matamoros, Mazatlán, Mérida, Mexicali, Minatitlán, Nuevo Laredo, Orizaba, Pachuca, Puebla, Querétaro, Reynosa, Roque, Salina Cruz, Saltillo, San Luis Potosí, Tehuacán, Tepic, Tijuana, Tlaxiaco, Toluca, Torreón, Tuxtepec, Valle de Oaxaca, Veracruz, Villahermosa, Zacatecas, Zacatepec, Altiplano de Tlaxcala, Coatzacoalcos, Cuautitlán Izcalli, Fresnillo, Irapuato, La Sierra Norte Puebla, Macuspana, Naranjos, Pátzcuaro, Poza Rica, Progreso, Puerto Vallarta, Tacámbaro, Tamazula Gordiano, Tlaxco, Venustiano Carranza, Zacapoaxtla, Zongolica y Oriente del Estado Hidalgo.	Elaboración del programa de estudio equivalente en la Reunión Nacional de Implementación Curricular y Fortalecimiento Curricular de las asignaturas comunes por área de conocimiento para los planes de estudio actualizados del SNEST..



Instituto Tecnológico de Morelia del 10 al 13 de septiembre de 2013.	Representantes de los Institutos Tecnológicos de: Aguascalientes, Apizaco, Boca del Río, Celaya, CRODE Celaya, Cerro Azul, Chihuahua, Cd. Cuauhtémoc, Cd. Hidalgo, Cd. Juárez, Cd. Madero, Cd. Valles, Coacalco, Colima, Iguala, La Laguna, Lerdo, Los Cabos, Matamoros, Mérida, Morelia, Motúl, Múzquiz, Nuevo Laredo, Nuevo León, Oriente del Estado de México, Orizaba, Pachuca, Progreso, Purépecha, Salvatierra, San Juan del Río, Santiago Papasquiaro, Tantoyuca, Tepic, Tlatlauquitpec, Valle de Morelia, Venustiano Carranza, Veracruz, Villahermosa, Zacatecas y Zacatepec.	Reunión Nacional de Seguimiento Curricular de las Asignaturas Equivalentes del SNIT.
----------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

#### 4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplica la programación orientada a objetos para resolver problemas reales y de ingeniería

#### 5. Competencias previas

Aplica algoritmos y lenguajes de programación para diseñar e implementar soluciones a problemáticas del entorno.
------------------------------------------------------------------------------------------------------------------

#### 6. Temario

No.	Temas	Subtemas
1	Introducción al paradigma de la programación orientada 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

## 7. Actividades de aprendizaje de los temas

1. Introducción al paradigma de la programación orientada a objetos	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i> Comprende y aplica los conceptos del paradigma de programación orientada a objetos para modelar situaciones de la vida real.</p> <p><i>Genérica(s):</i></p> <ul style="list-style-type: none"> <li>• Capacidad de análisis y síntesis.</li> <li>• Habilidad para manejo de equipo de cómputo.</li> <li>• Capacidad para trabajar en equipo</li> <li>• Solución de problemas.</li> </ul>	<ul style="list-style-type: none"> <li>• Investigar en diversas fuentes los conceptos principales del paradigma orientado a objetos para elaborar un resumen</li> <li>• Identificar ejemplos de la vida real donde se manifiesten dichos conceptos y comentarlos en clase</li> <li>• Redactar una definición propia de los conceptos de forma simple y entendible</li> <li>• Construir un diagrama de clases aplicados a distintos problemas utilizando un software adecuado e identificar las herramientas de representación utilizadas; elaborar reporte.</li> </ul>



## 2. Clases y objetos

Competencias	Actividades de aprendizaje
<p><b>Específica(s):</b> Aplica los conceptos de clases y objetos en el desarrollo de programas para solución de problemas de acuerdo al paradigma orientado a objetos.</p> <p><b>Genérica(s):</b></p> <ul style="list-style-type: none"> <li>Habilidad para manejo de equipo de cómputo</li> <li>Habilidad para trabajar de forma autónoma</li> <li>Solución de problemas</li> <li>Capacidad para aplicar los conocimientos en la práctica.</li> </ul>	<ul style="list-style-type: none"> <li>Realizar actividades grupales para que el alumno identifique mediante la abstracción las características y comportamientos de objetos de su entorno.</li> <li>Diseñar diagramas de clases relacionados a objetos de su entorno considerando únicamente la identificación de los atributos del objeto e implementar las clases en un lenguaje de programación orientado a objetos. Considerar modificadores de acceso públicos para exponer y comprender la vulnerabilidad de los datos.</li> <li>Diseñar diagramas de clases protegiendo los datos con modificadores de acceso privado o protegido y agregar métodos públicos para obtener acceso seguro a los mismos.</li> <li>Crear clases que reúnan los miembros necesarios para resolver un problema y así implementar el encapsulamiento</li> <li>Identificar el tiempo de vida de las variables al instanciar un objeto</li> <li>Identificar la estructura de un método y crear una aplicación que permita su uso en la resolución de problemas específicos</li> <li>Crear aplicaciones que contengan métodos sobrecargados y probar la utilidad de dichos métodos</li> <li>Elaborar reporte de prácticas.</li> </ul>

## 3. Herencia

Competencias	Actividades de aprendizaje
<p><b>Específica(s):</b> Identifica y aplica relaciones de herencia en clases derivadas para reutilizar los miembros de una clase base en el desarrollo de aplicaciones</p> <p><b>Genérica(s):</b></p> <ul style="list-style-type: none"> <li>Habilidad para manejo de equipo de cómputo</li> <li>Habilidad para trabajar de forma autónoma</li> <li>Solución de problemas</li> <li>Capacidad para aplicar los conocimientos en la práctica.</li> </ul>	<ul style="list-style-type: none"> <li>Elaborar un cuadro sinóptico en el que se muestren las definiciones de herencia y su clasificación</li> <li>Identificar los atributos y comportamientos propios de una especie que comparten los animales pertenecientes a ella</li> <li>Identificar los atributos y comportamientos propios de una categoría de objetos que comparten todos sus miembros</li> <li>Crear aplicaciones que manejen el concepto de herencia implementando redefinición de constructores y métodos</li> <li>Elaborar un reporte de prácticas.</li> </ul>



#### 4. Polimorfismo

Competencias	Actividades de aprendizaje
<p><b>Específica(s):</b> Aplica el concepto de polimorfismo para la definición de clases abstractas e interfaces que permitan reutilización de código.</p> <p><b>Genérica(s):</b></p> <ul style="list-style-type: none"><li>• Capacidad de análisis y síntesis.</li><li>• Habilidad para manejo de equipo de cómputo.</li><li>• Habilidad para trabajar de forma autónoma.</li><li>• Solución de problemas.</li></ul>	<ul style="list-style-type: none"><li>• Identificar clases base que no requieren ser instanciadas o que carezcan de sentido para ello por ser abstractas y discutirlo en clase</li><li>• Investigar en fuentes de información los conceptos y reglas para implementar clases abstractas en un programa y hacer un resumen</li><li>• Crear una aplicación donde se maneje herencia de interfaces para especializar los comportamientos que las clases podrán implementar</li><li>• Crear una aplicación donde se declaren 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</li><li>• Elaborar reporte de prácticas.</li></ul>

#### 5. Excepciones

Competencias	Actividades de aprendizaje
<p><b>Específica(s):</b> Comprende y aplica las condiciones apropiadas para evitar los errores que pueden interrumpir el flujo normal de ejecución de un programa a través del manejo de excepciones.</p> <p><b>Genérica(s):</b></p> <ul style="list-style-type: none"><li>• Capacidad de análisis y síntesis.</li><li>• Habilidad en el manejo de equipo de cómputo.</li><li>• Capacidad para trabajar en equipo.</li></ul>	<ul style="list-style-type: none"><li>• Investigar los tipos de excepciones predefinidas en fuentes de información diversas incluyendo el API.</li><li>• Crear una aplicación que deliberadamente genere excepciones comunes para identificar:<ul style="list-style-type: none"><li>• sus nombres, sus causas, su comportamiento, y reporte de error.</li></ul></li><li>• Crear una aplicación que maneje una clase con varios métodos invocándose en cadena, donde el último método genere una excepción para estudiar y comprender la propagación de las mismas.</li><li>• Crear una aplicación que utilice la selectiva intenta para atrapar excepciones de diferentes tipos, y prevenir la interrupción de ejecución de un programa.</li><li>• 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.</li></ul>



	<ul style="list-style-type: none"> <li>● Crear una aplicación que permita el 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.</li> <li>● Crear una aplicación que implemente 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.</li> <li>● Elaborar reporte de prácticas.</li> </ul>
5. Flujo y archivos	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i> Comprende y aplica la clasificación de archivos y operaciones básicas sobre éstos para manipular su información.</p> <p><i>Genérica(s):</i></p> <ul style="list-style-type: none"> <li>● Habilidad para manejo de equipo de cómputo</li> <li>● Habilidad para trabajar de forma autónoma</li> <li>● Solución de problemas</li> <li>● Capacidad para aplicar los conocimientos en la práctica.</li> </ul>	<ul style="list-style-type: none"> <li>● Investigar en fuentes de información los conceptos y metodologías para manipular archivos de texto y binarios en un lenguaje de programación orientado a objetos y hacer un resumen</li> <li>● Crear una aplicación que maneje un archivo de texto y sus operaciones básicas</li> <li>● Crear una aplicación que maneje un archivo binario y sus operaciones básicas</li> <li>● Elaborar reporte de prácticas.</li> </ul>

## 8. Práctica(s)

<ul style="list-style-type: none"> <li>● 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.</li> <li>● 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.</li> <li>● 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.</li> <li>● 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.</li> <li>● 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.</li> <li>● Implementar la clase Matriz que sobrecargue los operadores +, -, * y / para este tipo de dato definido por el usuario.</li> </ul>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------





- 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.
- 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.
- 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.
- 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.
- Especializar la interfaz Vehiculo en al menos dos subinterfaces (VehiculoTerrestre o VehiculoAereo) que agreguen comportamientos abstractos que las clases deberán implementar.
- Programar clases que generen excepciones comunes como referencias nulas o desbordamientos numéricos para estudiar su naturaleza, comportamiento, prevención y lanzamiento.
- Implementar aplicaciones que almacenen y recuperen información de diferentes tipos de datos simples a través de un archivo de texto para persistir información.
- 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).

## 9. Proyecto de asignatura

El objetivo del proyecto que plantee el docente que imparta esta asignatura, es demostrar el desarrollo y alcance de la(s) competencia(s) de la asignatura, considerando las siguientes fases:

**Fundamentación:** marco referencial (teórico, conceptual, contextual, legal) en el cual se fundamenta el proyecto de acuerdo con un diagnóstico realizado, mismo que permite a los estudiantes lograr la comprensión de la realidad o situación objeto de estudio para definir un proceso de intervención o hacer el diseño de un modelo.

**Planeación:** con base en el diagnóstico en esta fase se realiza el diseño del proyecto por parte de los estudiantes con asesoría del docente; implica planificar un proceso: de intervención empresarial, social o comunitario, el diseño de un modelo, entre otros, según el tipo de proyecto, las actividades a realizar los recursos requeridos y el cronograma de trabajo.

**Ejecución:** consiste en el desarrollo de la planeación del proyecto realizada por parte de los estudiantes con asesoría del docente, es decir en la intervención (social, empresarial), o construcción del modelo propuesto según el tipo de proyecto, es la fase de mayor duración que implica el desempeño de las competencias genéricas y específicas a desarrollar.



**Evaluación:** es la fase final que aplica un juicio de valor en el contexto laboral-profesión, social e investigativo, ésta se debe realizar a través del reconocimiento de logros y aspectos a mejorar se estará promoviendo el concepto de “evaluación para la mejora continua”, la metacognición, el desarrollo del pensamiento crítico y reflexivo en los estudiantes.

## 10. Evaluación por competencias

Las técnicas, herramientas y/o instrumentos sugeridos que permiten obtener el producto del desarrollo las actividades de aprendizaje: prácticas, mapas conceptuales, mapas mentales, resúmenes, reportes de prácticas.

Las técnicas, herramientas y/o instrumentos sugeridos que me permite constatar el logro o desempeño de las competencias del estudiante: matriz de valoración, listas de cotejo, listas de verificación, guías de observación.

## 11. Fuentes de información

1. Budd, T. (2002). Object Oriented Programming. Third edition: Addison Wesley.
2. Ceballos F. (2003). Programación Orientada a Objetos con C++. 3ra. Edición: Editorial Ra-Ma
3. Ceballos J. (2012) Microsoft C# -Curso de Programación. España: Alfaomega
4. Clark D., (2013). Begining C# Object-Oriented Programming, Apress.
5. Craig I. (2002). The Interpretation of Object-Oriented Programming Languages. 2nd Edition: Springer. London.
6. Dean J. y Dean R. (2009) Introducción a la programación con Java.: McGraw Hill
7. Deitel P., Deitel H. (2013). Como programar en java. 9a. Edición. Pearson.
8. Doyle, B (2013) C# Programming: From Problem Analysis to Program Design. Cengage Learning
9. Groussard, T. (2009) Visual Basic.NET (VB.NET) - Programe con Visual Studio 2008. Espana: Eni Ediciones
10. Groussard, T. (2011) Recursos Informáticos C#4 Los fundamentos del lenguaje- desarrollar con visual estudio 2010. España: Eni Ediciones
11. Gutiérrez F., Duran F., Pimentel E. (2007). Programación Orientada a Objetos con JAVA: Ediciones Paraninfo, S.A.
12. Harvey M. (2008). Como programa en Java. México: Prentice Hall.
13. Holmes B., Joyce D. (2001). Object-Oriented Programming with JAVA. Jones and Barlett: Canada. Publishers Inc.
14. Joyanes L. (2011). Fundamentos de programación: Algoritmos, Estructuras de Datos y Objetos. 3ra. Edición. Mac-Graw Hill
15. Joyanes, L. y Zahonero, I. (2011) Programación en Java 6. España: McGraw Hill
16. Larman G. (2002). UML y Patrones 2/E: Pearson Educacion.
17. Marrer G. (2009). Fundamentals of Programming with Object Orientated Programming. Python Edition. Ebook Edition.
18. Muñoz C., Nino A. Vizcaino A. (2002). Introducción a la Programación con Orientación a Objetos: Pearson Educacion.
19. Schildth H. (2002). Fundamentos de programación en Java: Mac-Graw Hill
20. Solana Aroa (2009). Programación con C# 4.0. Madrid, España.
21. Velez Serrano, J.F., Peña Abril, A., Gortázar Bellas, F. Sánchez Calle, A. (2010) Diseñar y Programar, Todo Es Empezar: Una Introducción a la Programación Orientada a Objetos Usando UML Y JAVA (EBOOK).
22. VV.AA. (2004). C/C++ Y Java. Como programar (4ª Ed). México



23. VV.AA., (2006) Introducción a la Programación Orientada a Objetos: Universidad de Alicante. Servicio de Publicaciones.
24. Warnes D., Kolling M.(2007). Programación Orientada a Objetos con Java. 3ra Edición: Prentice- Hall.
25. BlueJ, (2014). A free Java Development Environment designed for beginners. Disponible en internet en [www.bluej.org](http://www.bluej.org). Fecha de acceso: 13 de Febrero de 2014.
26. Greenfoot, (2014). Teach and learn java programming. Disponible en internet en [www.greenfoot.org](http://www.greenfoot.org). Fecha de acceso: 13 de Febrero de 2014.
27. Jeroo, (2014). Jeroo. Disponible en internet en <http://home.cc.gatech.edu/dorn/jeroo>. Fecha de acceso: 13 de Febrero de 2014
28. Oracle (2014). Java platform – Standard edition 7 API specification. Disponible en internet en <http://docs.oracle.com/javase/7/docs/api/>. Fecha de acceso: 13 de Febrero de 2014