

Nombre de la Asignatura	Español	Inglés
	Programación Orientada a Objetos	Object Oriented Programming

Clave	Nivel	Tipo de asignatura (área)

Créditos	Bajo conducción docente	Trabajo Independiente del Estudiante
10	6	4

Planes a los que se ofrece como mayor	5IS, 4IE, 1TI, 1RT
---------------------------------------	--------------------

Prerrequisitos	Fundamentos de Programación
----------------	-----------------------------

CONTEXTO:
 En el mundo real se puede observar a través de los objetos que residen en él (personas, animales, plantas, edificios, vehículos) y de la interacción que tienen entre ellos. De esta forma el ser humano piensa y genera nuevos conocimientos en términos de objetos.

El mundo de los objetos se llevó al área de la computación, donde los artesanos de ésta pudieran desarrollar sus ideas para satisfacer las necesidades en una forma más ordenada y natural para el ser humano, una forma que se le llamó "programación orientada a objetos". Este nuevo enfoque tiene como propósito reducir la distancia entre el razonamiento humano y el lenguaje de los ordenadores.

El surgimiento del enfoque orientado a objetos (OO) en el desarrollo del software, ha tenido grandes ventajas sobre los enfoques tradicionales. Los desarrolladores pueden visualizar los sistemas como grupos de entidades que se relacionan e interactúan, lo que les permite desarrollar sistemas más grandes y más complicados en menor tiempo y en forma más organizada.

La programación orientada a objetos (POO) en conjunto con lenguajes que permiten desarrollarla, está brindado una plataforma para el desarrollo de software distribuido, software que antes era inimaginable de construirse.

- OBJETIVOS ORIENTADOS A COMPETENCIAS:**
1. El alumno comprenderá los conceptos relacionados con el paradigma de programación orientada a objetos
 2. El alumno conocerá las características fundamentales y la sintaxis básica del lenguaje Java
 3. El alumno aplicará los conceptos de orientación a objetos (modularidad, encapsulamiento, abstracción, herencia, agregación, polimorfismo) codificando aplicaciones en lenguaje Java, a partir de un modelo en notación UML
 4. Con el fin de aprovechar las facilidades del lenguaje Java, el alumno desarrollará aplicaciones que impliquen almacenamiento de datos en disco, utilizando manejo de archivos
 5. A partir de la implementación de una aplicación con interfaz amigable, el alumno ejercitará el conocimiento adquirido acerca de las librerías visuales de Java para coadyuvar a la correcta interacción entre usuario y equipo

6. El alumno participará en el desarrollo de un proyecto por equipo que involucre investigación, análisis, modelado y codificación de un método numérico asignado por el profesor

SABERES NECESARIOS

1. El paradigma orientado a objetos (horas)
 - 1.1 Introducción a los objetos
 - 1.2 POO vs PE
 - 1.3 Características de los LOO
2. Fundamentos de Java (horas)
 - 2.1 Origen y características de Java
 - 2.2 Programación básica: a) comentarios, b) identificadores, c) tipos de datos, d) operadores
 - 2.3 Control de flujo: a) if, b) switch, c) for, d) do/while
 - 2.4 Arreglos de tipos de primitivos
3. Estructura de un programa en Java - clases
 - 3.1 Encabezado : import, package
 - 3.2 Declaración de una clase
 - 3.3 Definición de atributos
 - 3.4 Introducción a los métodos : a) invocación, b) return, c) main, d) constructor, e) sobrecarga
 - 3.5 Ámbito de una variable
 - 3.6 Representación gráfica de las clases usando UML
4. Introducción a los objetos
 - 4.1 Creación
 - 4.2 Visibilidad de atributos / métodos (private y public)
 - 4.3 Uso de atributos y métodos
 - 4.4 Alcance de un objeto
 - 4.5 Pase por valor / referencia
5. Modificadores
 - 5.1 Para atributos : final, static
 - 5.2 Para métodos : static
6. Herencia y Polimorfismo. Ventajas: reutilización de código y flexibilidad
 - 6.1 Especialización. Introducción. Presentar modelo UML
 - Clase base
 - Visibilidad de atributos / métodos (protected)
 - Clase derivada, transmisión de atributos *protected* y objetos de clases derivadas
 - Instrucción *super*
 - Sobreescritura de métodos
 - 6.2 Generalización. Introducción. Presentar modelo UML
 - Clase abstracta, métodos abstractos
 - Clases derivadas y objetos de clases derivadas
 - Enlace dinámico y estático
 - Polimorfismo en acción
 - 6.3 Herencia múltiple. Introducción. Presentar modelo UML
 - Interfaces
 - Clase implementadora
 - Polimorfismo en acción
7. Almacenamiento de datos
 - 7.1 Arreglos (de objetos)
 - Características, ventajas vs. vectores
 - 7.2 Vectores
 - Características, ventajas vs. Arreglos
 - Uso de métodos para: obtener tamaño, agregar, eliminar, sustituir y obtener datos
 - 7.3 Implementación de agregación de clases: 1..1, 1..n, 1..*
 - 7.4 Manejo del try y catch
 - 7.5 Gestión de Archivos
8. Librerías visuales de Java (Swing, AWT)

SITUACIONES DE APRENDIZAJE				
PROPÓSITOS	CONOCIMIENTOS – HABILIDADES	ACTIVIDADES	ESCENARIOS	INDICADORES DE EVALUACIÓN
Que el alumno comprenda el objetivo y los conceptos de la programación orientada a objetos.	Ninguna.	Lecturas de texto. Ejemplificar conceptos durante la clase.	Trabajo en clase.	Exámenes.
Que el alumno sea capaz de codificar un programa en Java utilizando las herramientas disponibles (IDEs) y la sintaxis básica.	Que el alumno tenga nociones de diseño de algoritmos para la resolución de problemas y sintaxis del lenguaje C.	Prácticas en clase. Ejercicios de tarea. Practicar en casa ejemplos obtenidos de libros.	Trabajo en laboratorio.	Ejercicios resueltos en clase Ejercicios de Tarea. Exámenes.
Que el alumno implemente los conceptos de OO utilizando Java, a partir de modelos en notación UML.	Conceptos de OO. Sintaxis básica de Java. Manejo de las herramientas para compilación y ejecución. Notación UML.	Lecturas de texto. Prácticas en clase. Ejercicios de tarea. Practicar en casa ejemplos obtenidos de libros.	Trabajo en clase. Trabajo en laboratorio.	Ejercicios resueltos en clase Ejercicios de Tarea. Exámenes.
Que el alumno desarrolle aplicaciones para la gestión de archivos.	Que el alumno pueda implementar los conceptos de OO utilizando Java.	Lecturas de texto. Prácticas en clase. Desarrollo de un proyecto.	Trabajo en laboratorio.	Ejercicios resueltos en clase Proyecto.
Que el alumno conozca e implemente un método numérico para el cálculo de raíces de funciones.	Conocimientos básicos de funciones matemáticas. Implementación de los conceptos OO con Java.	Lecturas de texto. Estudio de ejemplos Exposición en clase. Desarrollo de un proyecto.	Trabajo en clase. Investigación extraclase. Trabajo en laboratorio.	Exposición del método numérico Proyecto final.
Que el alumno se familiarice con las librerías visuales de Java para el desarrollo de aplicaciones amigables.	Implementación de los conceptos OO con Java.	Prácticas en clase. Desarrollo de un proyecto.	Trabajo en laboratorio.	Ejercicios resueltos en clase. Proyecto.

EVALUACIONES
Tres Exámenes Parciales 15 % cada uno Un Examen Final Departamental 15 % Tareas y Prácticas 20 % Proyecto 20 % En esta materia se aplicará Examen Extraordinario, de acuerdo a lo contemplado en los reglamentos del ITESO.
<ul style="list-style-type: none"> • El proyecto estará relacionado con la implementación de un método numérico. • Se debe promediar 60 (calificación sobre 100 puntos) entre los cuatro exámenes para acreditar la materia.

BIBLIOGRAFÍA / RECURSOS
Deitel, Java. Como Programar, Quinta Edition, Pearson, 2004 Humphrey, Watts; "Introducción al Proceso Software Personal (PSP), Addison Wesley, 2001 Bronson, C++ para Ingeniería y Ciencias, Capítulo 13, Thomson, 1999 Tremblay, Jean Paul; "Data Structures and Software Development in an Object-Oriented Domain. Java Edition". Pearson 2003 Watt, David A.; "Java Collections. An introduction to abstract data types, data structures and algorithms", John Wiley & Sons, 2003 Kit, Edward; "Software Testing in the Real World", Addison Wesley, 1995