

**Nombre de la asignatura:** Programación Orientada a Objetos

**Créditos:** 2- 3- 5

**Aportación al perfil**

- Desarrollar, implementar y administrar software de sistemas o de aplicación que cumpla con los estándares de calidad con el fin de apoyar la productividad y competitividad de las organizaciones.

**Objetivo de aprendizaje**

- Analizar, diseñar, desarrollar e implementar soluciones orientadas a objetos utilizando encapsulamiento, herencia, polimorfismo y archivos.

**Competencias previas**

- Analizar, diseñar, desarrollar e implementar soluciones de problemas utilizando estructuras condicionales, repetitivas y arreglos en un lenguaje de programación.

## Temario

- Introducción al paradigma orientado a objetos
  - Elementos del modelo de objetos.
- Clases y objetos
  - Declaración de clase.
  - Instanciación de una clase.
  - Invocación de métodos (mensajes).
  - Constructores y destructores.
  - Sobrecarga de métodos.
  - Sobrecarga de operadores.
- Herencia
  - Definición.
  - Clasificación.
  - Implementación.
  - Reutilización de miembros heredados
  - Referencia al objeto de la clase base.
  - Constructores y destructores en clases derivadas.
  - Redefinición de métodos en las clases derivadas.
- Polimorfismo
  - Definición.
  - Clases abstractas.
  - Interfaces.
  - Variables polimórficas.
- Excepciones
  - Definición.
  - Tipos de Excepción.
  - Excepciones comunes.
  - Propagación de excepciones.
  - Gestión de excepciones.
  - Excepción definidas por el usuario.
- Flujos y archivos
  - Definición.
  - Clasificación.
  - Operaciones básicas.

## **Definición de las competencias específicas**

- Comprender y describir los conceptos principales del paradigma orientado a objetos.
- Crear clases y objetos aplicando encapsulamiento para proteger los datos.
- Creación de métodos para enviar mensajes a un objeto.
- Implementar constructores para inicializar atributos y liberar recursos.
- Sobrecargar métodos y operadores para optimizar el código de una clase.
- Implementar la herencia en una clase derivada para reutilizar los miembros de una clase base.
- Redefinir un método en una clase derivada para modificar el comportamiento de una clase base.
- Implementar clases abstractas e interfaces para generar diferentes comportamientos en clases derivadas.
- Manejar y gestionar excepciones para prevenir la interrupción de la ejecución de un programa.
- Crear y gestionar excepciones personalizadas para manejar errores definidos por el usuario.
- Crear y manipular archivos para guardar y recuperar información en memoria secundaria.

### **Sugerencias didácticas transversales para el desarrollo de competencias profesionales**

- 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.
- Fomentar actividades grupales que propicien la comunicación, el intercambio argumentado de ideas, la reflexión, la integración y la colaboración de y entre los estudiantes.
- 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.

## Prácticas

- 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.
- Analizar objetos de la vida real para abstraer y modelar sus atributos y comportamientos.
- Implementar aplicaciones que utilicen clases con métodos que usen tipos de datos simples como parámetros y valores de retorno, así como métodos sin valores de retorno.
- Desarrollar aplicaciones que impliquen clases con métodos que usen objetos como atributos, parámetros y valores de retorno.
- Crear clases que definan varios constructores para inicializar sus atributos.
- Implementar destructores en una clase para liberar recursos de una clase.
- Implementar aplicaciones que utilicen clases con métodos sobrecargados.
- Desarrollar aplicaciones que requieran el uso de operaciones aritméticas y comparaciones entre objetos de la misma clase, para sobrecargar operadores.
- Implementar la herencia de una clase base en una clase derivada para reutilizar sus miembros.
- Sobrescribir un método en una clase derivada para cambiar el comportamiento de una clase base.
- Diseñar clases abstractas e interfaces para dar flexibilidad a los programas al modificar los comportamientos en clases derivadas.
- Manejar y gestionar excepciones predefinida o definida por el usuario para proteger las ejecuciones de un programa.
- Implementar aplicaciones que almacenen y recuperen información de diferentes tipos de datos simples y objetos a través de un archivo de texto para persistir la información.

## **Criterios de evaluación**

La evaluación de la asignatura se hará con base en siguiente desempeño:

- Elaboración de aplicaciones orientadas a objetos que requieran las herramientas de encapsulamiento, herencia, polimorfismo y archivo.