Universidad Católica del Norte

PROGRAMA DE ASIGNATURA

I. IDENTIFICACIÓN

Carrera o programa: Ingeniería en Tecnologías de Información				
Unidad responsable: Escuela de Ingeniería				
Nombre de la asignatura: Programación Orientada a Objetos				
Código: ECIN-00361				
Semestre en la malla[1]: 3				
Créditos SCT – Chile: 5				
Ciclo de Formación	Básico	X	Profesional	
Tipo de Asignatura	Obligatoria	X	Electiva	
Clasificación de área de conocimiento[2]				
Área: Ingeniería y Tecnología		**Subárea:** Ingeniería Eléctrica, Electrónica e Informática		
Requisitos:				
Prerrequisitos:		**Requisitos para:**		
• Programación		• Fundamentos de Ciencias de la Computación • Estructuras de Datos • Arquitectura y Organización de Computadores		

II. ORGANIZACIÓN SEMESTRAL

Horas Dedicación Semanal (Cronológicas)	Docencia Directa	6	Trabajo Autónomo	2	Total	8	
Detalle Horas Directas	Cátedra	Ayudantía	Laboratorio	Taller	Terreno	Exp. Clínica	Supervisión
	3	1.5		1.5			

III. APORTE AL PERFIL DE EGRESO

La asignatura contribuye al dominio 1 del perfil de egreso, “Conocimiento científico y disciplinario”. Además, contribuye al dominio 2 “Habilidades y Actitudes Personales y Profesionales”. También contribuye al dominio 3 “Habilidades Interpersonales”. Por último, contribuye al dominio 4 “Habilidades para la Práctica de la Ingeniería”. Al finalizar la asignatura las y los estudiantes serán capaces de describir conceptos, formular algoritmos y soluciones a problemas aplicando el paradigma de la programación orientada al objeto.

IV. COMPETENCIAS

[1] Este campo sólo se completa en caso de carreras con programas semestrales.

[2] Clasificación del curso de acuerdo a la OCDE

La carrera declara las siguientes habilidades:
2.1. Identificar y resolver problemas con un razonamiento analítico.
2.4 Demostrar habilidades personales que contribuyen a una práctica exitosa de la ingeniería: iniciativa, toma de decisiones, perseverancia, pensamiento crítico, aprendizaje continuo, pensamiento creativo, orientación al logro, flexibilidad, autoevaluación, gestión del tiempo y recursos.
3.1. Liderar y trabajar en equipos multidisciplinarios y multiculturales.
3.2. Comunicar comprensivamente información técnica en español, en forma oral, escrita, y gráfica, a nivel avanzado.
4.3. Concebir soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.
4.4. Diseñar soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.
4.5. Implementar soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.

V. RESULTADOS DE APRENDIZAJE

1. Aplicar técnicas de ingeniería de software en la creación de software legible, mantenible y testeable.
2. Aplicar técnicas de programación orientada al objeto en la resolución de problemas.
3. Crear tipos de datos abstractos con bajo acoplamiento entre la implementación y su comportamiento que permitan la resolución problemas.
4. Analizar las relaciones causa efecto de los algoritmos creados.
5. Identificar los objetivos y requerimientos de las soluciones TIC
6. Seleccionar los procesos, técnicas y herramientas adecuados de acuerdo a los requerimientos.
7. Desarrollar la solución tecnológica más adecuada en base a las características del problema y los recursos disponibles.
8. Seleccionar información útil para la generación de nuevos saberes vinculados a contextos académicos – profesionales.

VI. ÁREAS TEMÁTICAS

1. Lenguaje de Programación Java
1.1 Sintaxis y semántica.
1.2 Compilación e interpretación en una máquina virtual del lenguaje orientado a objetos.
1.3 Usos de IDE para programar en el lenguaje.

2. Conceptos y Prácticas de Ingeniería de Software
2.1 Correctitud de un programa: Conceptos de especificación de requisitos, Programación defensiva (manejo de excepciones, codificación segura, generics), Revisión de código.
2.2 El rol y uso de contratos, incluyendo pre y pos-condiciones.
2.3 Documentación y estilo de programación.
2.4 Estrategias de Debugging.

3. Concepto de Clase y Objeto basado en Clases
3.1 Descomposición, abstracción, encapsulación en Programación Orientada a Objetos.
3.2 Descomposición en objetos conteniendo estado y comportamiento.
3.3 Diseño de jerarquía de clases como base de modelado.

3.4 Definición de objetos: atributos, métodos y constructores.
3.5 Modificadores de encapsulación de orientación a objetos: private, protected, public.
3.6 Definición de clases: atributos y métodos (estáticos).
3.7 Definiciones de packages y modificadores: private-package.
3.8 Uso de colecciones en orientación a objetos: List, ArrayList, LinkedList, vectors, generics.

4. Herencia y jerarquía de clases.
4.1 Subclases, herencia y sobreescritura de métodos.
4.2 Enlace dinámico: definición de llamada de métodos.
4.3 Polimorfismo.
4.4 Clases abstractas.
4.5 Relación entre subtipo y herencia.
4.6 Interfaces.
4.7 Distintos tipos de herencias.

5. UML y Patrones de Diseño
5.1 Diagrama de casos de uso.
5.2 Diagrama de clases: asociación, agregación, composición, herencia, interface.
5.3 Diagrama de secuencias.
5.4 Patrones de diseño: Singleton, Façade, Strategy, Builder.

6. Interfaces Gráficas de Usuario (GUI) [Ejemplo de OOP]
6.1 Diseño de Interfaces Gráficas de Usuarios
6.2 Implementación de Interfaces Gráficas de Usuarios
6.3 Uso de librerías graficas de GUI, por ejemplo, creación de ventana principal, uso de menús, formularios, botones, listas, e imágenes.
6.4 Incorporación de interfaces gráficas en el desarrollo de aplicaciones.

VII. ORIENTACIONES METODOLÓGICAS

1. La metodología a desarrollar en esta asignatura es principalmente práctica, por lo que se debe favorecer la interacción entre los y las estudiantes, a través de trabajos prácticos colaborativos.
2. Se sugiere trabajar la teoría a través de metodologías activas como clase invertida, ABP, entre otras, generando instancias de presentación oral individual y/o grupal, favoreciendo el aprendizaje contextualizado.
3. Las experiencias de cátedra y taller deben ser realizadas por medio de la utilización de software moderno aplicable a la asignatura.
4. Se recomienda que las y los estudiantes realicen presentaciones periódicas sobre el trabajo realizado en taller.

VIII. ORIENTACIONES Y CRITERIOS PARA EVALUACIÓN

1. Se recomienda la aplicación de una evaluación diagnóstica al inicio de la asignatura.
2. En cátedra, se sugiere implementar estrategias de evaluación sumativas, con una ponderación del 70% de la nota final de la asignatura.
3. En taller se sugiere implementar estrategias de evaluación sumativas, con una ponderación del 30% de la nota final de la asignatura.
4. Las actividades podrán ser individuales o grupales.

5. La asistencia mínima exigida para las actividades de cátedra es del 70%.
6. Se exigirá un 60% de logro de objetivos para aprobar las actividades de evaluación.
7. Se recomienda realizar evaluaciones de carácter formativo con retroalimentación de carácter personal.

IX. RECURSOS BIBLIOGRÁFICOS

Bibliografía Mínima

– Ross, E. (2023). Programación Orientada a Objetos, Libro Guía.

Bibliografía Complementaria

– Barnes, D. J., & Kölling, M. (2012). Objects First with Java (5th ed.). Prentice Hall / Pearson Education.
– Weiss, M. A. (2013). Data Structures and Problem Solving Using Java: Pearson New International Edition. Reino Unido: Pearson Education.