Universidad
Católica del Norte

Programa de Asignatura

I. IDENTIFICACIÓN				
Carrera o programa: Ingeniería Civil en Computación e Informática				
Unidad responsable: Escuela de Ingeniería				
Nombre de la asignatura: Programación Orientada a Objetos				
Código: ECIN-00307				
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		**Sub área:** Ingeniería Informática		
Requisitos:				
Pre-requisitos:		**Requisito para:**		
• ECIN-00201 Programación		• ECIN-00407 Estructuras de Datos • ECIN-00506 Fundamentos de la Computación		

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	Supervision
	3	1.5		1.5			

[1] Este campo

[2] Clasificación del curso de acuerdo a la OCDE

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 4 “Habilidades para la Práctica de la Ingeniería”. También contribuye al dominio 3 “Habilidades Interpersonales”. 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. HABILIDADES PERFIL DE EGRESO (RELACIÓN)

1.3 Aplicar conocimientos, métodos y herramientas de la especialidad para resolver problemas complejos de Ingeniería de Software, Plataformas y Gestión de Tecnologías.

2.1 Identificación, formulación, modelación y resolución de problemas complejos de ingeniería considerando las interacciones y la dinámica de las variables.

3.1 Liderar y trabajar en equipos de trabajo multidisciplinarios

3.2 Comunicar comprensivamente información técnica en español, en forma oral, escrita, y gráfica, a nivel avanzado

4.3 Concebir las soluciones TIC requeridas en las organizaciones haciendo uso eficiente de los recursos como personas, información, y procesos. Estas soluciones consideran las arquitecturas TI junto a sus modelos de servicios y modelos operativos; los sistemas de software; y las plataformas de cómputo y comunicaciones junto a sus servicios asociados.

4.4 Diseñar las soluciones TIC para la industria intensiva en procesamiento de la información. Estas soluciones consideran las arquitecturas TI junto a sus modelos de servicios y modelos operativos; los sistemas de software; y las plataformas de cómputo y comunicaciones junto a sus servicios asociados.

4.5 Implementar las soluciones TIC. Estas soluciones consideran las arquitecturas TI junto a sus modelos de servicios y modelos operativos; los sistemas de software; y las plataformas de cómputo y comunicaciones junto a sus servicios asociados.

4.6 Operar arquitecturas TI junto a sus modelos de servicios y modelos operativos; los sistemas de software; y las plataformas de cómputo y comunicaciones junto a sus servicios aso-

ciados.

CG1 Capacidad de Autoaprendizaje: habilidades de tipo metacognitivo que permiten descubrir y aprovechar constantemente oportunidades de aprendizaje, teniendo una permanente actitud de superación y espíritu inquisitivo, logrando con ellos la motivación necesaria para construir nuevos conocimientos y transferirlos a nuevas situaciones. El egresado de la UCN se hace cargo de construir su destino académico con Libertad, valiéndose de las oportunidades de espacio, contexto y relaciones que se generan en el entorno educativo.

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 procesos en estudio.
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. Introducción al lenguaje
   - 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, genericos.
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.

VII. ORIENTACIONES Y CRITERIOS PARA LA 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

– David J. Barnes & Michael Kölling. (2012). Objects First with Java (Fifth edition). Prentice Hall / Pearson Education.

Bibliografía Complementaria

– Weiss, Mark Allen. (2009). Data Structures and Problem Solving Using Java (Fourth edition). Addison-Wesley.
– Texto Guía, disponibles en la Plataforma Campus Virtual UCN