UNIVERSIDAD DEL BÍO-BÍO
La Libertad del Conocimiento

UNIVERSIDAD DEL BÍO-BÍO
VICERRECTORÍA ACADÉMICA – DIRECCION DE DOCENCIA

PROGRAMA DE ASIGNATURA

I. IDENTIFICACIÓN

Nombre asignatura: Programación Orientada a Objeto	**Período de Vigencia**: 2013-2014
Código:	
Tipo de Curso: Obligatorio, Formación de Especialidad	

Carrera: Ingeniería Civil en Informática	**Departamento**: Sistemas de Información, Ciencias de la Computación y Tecnologías de Información	**Facultad**: Ciencias Empresariales
Nº Créditos SCT: 8	**Total de horas** Cronológicas: 240 Pedagógicas: 360	**Año / semestre:** 1/ 2

Horas presenciales: 144 **HT:** 4 **HP:** 2 **HL:** 2	**Horas trabajo autónomo:** 216 **HT:** 4 **HP:** 4 **HL:** 4
Prerrequisitos: Asignatura: Introducción a la Programación Código:	**Correquisitos:** No Tiene

II.- DESCRIPCIÓN

II.1 Presentación: Relación de la Asignatura con las Competencias del Perfil de Egreso

Esta asignatura permitirá al estudiante resolver problemas simples utilizando un lenguaje de programación orientado a objeto. Se imparte en el segundo semestre de la carrera.

Tributa a las subcompetencias específicas:

- Analizar las problemáticas de las organizaciones y de los individuos con el objeto de determinar requerimientos de software usando técnicas definidas para este propósito.
- Resolver problemas de programación utilizando lenguajes de programación y modelado de acuerdo a reglas y estándares existentes, y aplicando estrategias que aseguren la generación de soluciones eficientes.
- Construir aplicaciones de software, probando su funcionalidad y eficiencia, mediante el uso de arquitecturas, modelos, patrones, técnicas y herramientas de programación pertinentes para distintas plataformas.

Y la competencia específica:

- Construir bases de datos que permitan satisfacer las necesidades de información de las organizaciones o individuos, mediante el uso de diversas técnicas de modelado.

Además, contribuye a la competencia genérica:

- Manifestar una actitud permanente de búsqueda y actualización de sus aprendizajes, incorporando los cambios sociales, científicos y tecnológicos en el ejercicio y desarrollo de su profesión.

II.2 Descriptor de competencias

Resolver problemas simples de programación orientado a objeto usando un lenguaje que se ajuste a dicho enfoque apoyándose en un entorno de programación.

Resultados de Aprendizaje:

1. Identifica los conceptos del enfoque orientado a objeto para posibilitar la construcción de software bajo dicho enfoque.
2. Interpreta diagramas de clases representados mediante un lenguaje de modelado para construir software que implemente dichos diagramas.
3. Construye software constituido por una única clase que implementa elementos básicos del enfoque orientado a objeto, apoyado por un entorno de programación, para comprender el uso de clases y objetos.
4. Construye software constituido por una única clase que implementa persistencia, recursividad e interfaces gráficas bajo el enfoque orientado a objeto, apoyado por un entorno de programación, para enfrentar la forma de resolver problemas de programación.
5. Construye software orientado a objeto constituido por múltiples clases usando un lenguaje de programación para resolver problemas simples, a partir de diagramas de clase, utilizando un entorno de programación.

II.3 Aprendizajes Previos

Construye programas usando un lenguaje de programación procedimental para resolver problemas simples.

I. RESULTADOS DE APRENDIZAJE

Resultados de Aprendizaje	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales
1. Identifica los conceptos del enfoque orientado a objeto para posibilitar la construcción de software bajo dicho enfoque.	1.1. Describe los conceptos del enfoque orientado a objeto. 1.2. Relaciona entre sí los conceptos del enfoque orientado a objeto. 1.3. Ejemplifica los conceptos del enfoque orientado a objeto y sus relaciones valorando el aporte de dicho enfoque.	<u>Conceptual</u> • Concepto de abstracción. • Concepto de clase y objeto y su relación con la realidad que representan. • Estructura de una clase: atributos y operaciones. • Relaciones entre clases: herencia y polimorfismo, agregación, asociación. <u>Procedimental</u> Analogías de objetos y sus

Resultados de Aprendizaje	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales
		relaciones en el mundo real. Actitudinal • Valora el aporte del enfoque orientado a objeto.
2. Interpreta diagramas de clases representados mediante un lenguaje de modelado para construir software que implemente dichos diagramas.	2.1. Indica el propósito de los diagramas de clases. 2.2. Describe el significado de los símbolos del lenguaje de modelado presentes en los diagramas de clases. 2.3. Describe la situación representada en un diagrama de clases interpretando los símbolos usados en él.	Conceptual • Propósito de los diagramas de clases. • Símbolos del diagrama de clases de un lenguaje de modelado y su significado: clase, atributo, operación, asociación (o uso), agregación, multiplicidad, rol, navegabilidad, herencia, clase abstracta. Procedimental • Reglas para interpretar diagramas de clases.
3. Construye software constituido por una única clase que implementa elementos básicos del enfoque orientado a objeto, apoyado por un entorno de programación, para comprender el uso de clases y objetos.	3.1. Identifica los elementos básicos de un lenguaje de programación orientado a objeto en un programa dado. 3.2. Distingue los elementos básicos presentes en una clase implementada mediante un lenguaje de programación orientado a objeto. 3.3. Implementa los elementos básicos de una clase, según las reglas sintácticas, semánticas y de codificación correspondientes, apoyado por un entorno de programación. 3.4. Verifica el funcionamiento de la clase implementada mediante el desarrollo y ejecución de una clase de prueba, apoyado por un entorno de programación.	Conceptual • Concepto de lenguaje de programación orientado a objeto. • Lenguajes de programación orientados a objeto existentes. • Elementos básicos presentes en un lenguaje de programación orientado a objeto, su sintaxis y semántica: clase, constante, atributo, constructor, método, tipo de acceso, atributo y método estático, arreglo, entrada/salida estándar, principales clases de la API del lenguaje (operaciones matemáticas, de cadenas de caracteres, de fechas, clases contenedoras de objetos y las que posibilitan el uso de los elementos antes mencionados). Procedimental • Instructivo de uso de un entorno de programación. • Implementación de una clase básica. • Verificación del

Resultados de Aprendizaje	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales
		funcionamiento de una clase a través de la creación de una clase de control.
4. Construye software constituido por una única clase que implementa persistencia, recursividad e interfaces gráficas simples bajo el enfoque orientado a objeto, apoyado por un entorno de programación, para enfrentar la forma de resolver problemas de programación.	4.1. Define los conceptos de archivo y recursividad como parte de los recursos y estrategias de desarrollo de software. 4.2. Identifica los elementos que permiten la implementación de persistencia, mediante archivos, recursividad e interfaces gráficas en un programa dado. 4.3. Implementa clases que usan archivos, recursividad y proveen interfaces gráficas según las reglas sintácticas, semánticas y de codificación correspondientes, con el apoyo de un entorno de programación. 4.4. Verifica el funcionamiento de la clase implementada mediante el desarrollo y ejecución de una clase de prueba, apoyado por un entorno de programación.	Conceptual • Conceptos de archivo, método recursivo y su estructura. • Elementos del lenguaje que permiten el uso de archivos y la construcción de interfaces gráficas. Procedimental • Estrategias de implementación de clases que manejen archivos, usen recursividad y provean de interfaz gráfica. • Técnica de prueba funcional unitaria bajo el enfoque orientado a objeto, implementada en una clase de control.
5. Construye software constituido por múltiples clases usando un lenguaje de programación orientado a objeto para resolver problemas simples, a partir de diagramas de clase, utilizando un entorno de programación.	5.1. Implementa la captura y manejo de errores producidos en tiempo de ejecución en una clase. 5.2. Implementa múltiples clases relacionadas entre sí utilizando las reglas sintácticas, semánticas y de codificación de un software orientado a objeto con el apoyo de un entorno de programación. 5.3. Verifica el correcto funcionamiento de un software orientado a objeto usando un entorno de programación.	Conceptual • Elementos que permiten capturar y manejar errores producidos durante ejecución. • Elementos que permiten implementar relaciones de asociación (o uso), agregación y herencia, así como polimorfismo, clases y métodos abstractos. Procedimental • Estrategias de implementación de clases relacionadas entre sí con el apoyo de un entorno de programación. • Técnica de prueba funcional de integración bajo el enfoque orientado a objeto. Actitudinal • Comprueba el correcto funcionamiento del software que desarrolla.

IV. ACTIVIDADES DE APRENDIZAJE Y EVALUACIÓN

Resultados de Aprendizaje	Actividades de Aprendizaje	Actividades de Evaluación	Tiempo Estimado
1. Identifica los conceptos del enfoque orientado a objeto para posibilitar la construcción de software bajo dicho enfoque.	1.1 Profesor: • Presenta apunte con los contenidos requeridos. • Entrega guía de ejercicios. • Sugiere lectura de apoyo. • Expone los distintos enfoques de programación y lenguajes de programación asociados. • Plantea preguntas para focalizar comprensión lectora y las publica en plataforma. 1.2 Estudiante: • Lee apunte recibido. • Resuelve guía de ejercicios. • Lee material de apoyo. • Responde preguntas en la plataforma	• Desarrollo de test de diagnóstico. • Construcción de mapa conceptual. • Resolución de ejercicios.	Horas presenciales: HT: 2 HP: 5 HL: 2 Horas de trabajo autónomo: HT: 2 HP: 5 HL: 2
2. Interpreta diagramas de clases representados mediante un lenguaje de modelado para construir software que implemente dichos diagramas.	2.1 Profesor: • Sugiere lectura en textos que expliquen los diagramas de clases y sus símbolos en un lenguaje de modelado. • Entrega apuntes con reglas básicas para la interpretación de diagramas de clases. • Entrega diagramas de clases de diversas temáticas usando al menos una vez cada símbolo. 2.2 Estudiante: • Lee textos y apuntes indicados por el profesor. • Revisa los diagramas de clases recibidos y aplica reglas para interpretarlos. • Compara su interpretación de los símbolos y de los diagramas de clases con sus pares y recibe	• Contesta test sobre conceptos asociados a los diagramas de clases y sus símbolos (pareo símbolos a significado, respuestas a preguntas). • Retroalimentación a la interpretación de diagramas de clases. • Interpretación de un diagrama de clases recibido en un enunciado usando las reglas trabajadas.	Horas presenciales: HT: 2 HP: 5 HL: 2 Horas de trabajo autónomo: HT: 4 HP: 10 HL: 4

Resultados de Aprendizaje	Actividades de Aprendizaje	Actividades de Evaluación	Tiempo Estimado
	retroalimentación.		
3. Construye software constituido por una única clase que implementa elementos básicos del enfoque orientado a objeto, apoyado por un entorno de programación, para comprender el uso de clases y objetos.	3.1 Profesor: • Indica fuente bibliográfica para lectura sobre lenguajes de programación. • Presenta apuntes de cada tema relacionado con la implementación de una clase (sin persistencia, interfaces gráficas, ni recursividad), incluyendo ejemplos en que se muestra la representación de una clase en un diagrama de clases y su implementación correspondiente, relacionando símbolos y elementos del lenguaje, así como la confección de las correspondientes clases de prueba para crear y usar objetos. • Entrega listado de clases con y sin errores. • Entrega listados de ejercicios que involucren distintos elementos del lenguaje de programación para construir una clase. • Conduce sesión de laboratorio para conocer entorno de programación que se usará como herramienta de apoyo. • Conduce laboratorios para construir y probar clases usando diversos aspectos. • Publica tests en línea previos a clase. 3.2 Estudiante: • Lee los diversos materiales y apuntes que clases a clase va dando el profesor. • Responde tests en línea. • Realiza los ejercicios	• Pareo de símbolos de una clase representada mediante el lenguaje de modelado y una clase implementada en un lenguaje de programación. • Identificación de elementos del lenguaje de programación en una clase dada. • Corrección de errores, si los hay, en una clase dada. • Confección de una clase a partir de un enunciado. • Confección de una clase de prueba que cree objetos de alguna clase e invoque sus métodos, a partir de un enunciado. • Codificación de una clase y su clase de prueba en un entorno de programación en laboratorio. • Retroalimentación y apoyo. Nota: Todo lo anterior considerando distintos elementos o características que puede incluir una clase.	Horas presenciales: HT: 12 HP: 30 HL: 12 Horas de trabajo autónomo: HT: 14 HP: 56 HL: 14

Resultados de Aprendizaje	Actividades de Aprendizaje	Actividades de Evaluación	Tiempo Estimado
	propuestos que se relacionan con el tema tratado en cada apunte con y sin apoyo del profesor. • Contrasta sus soluciones con las dadas por el profesor y/o sus pares e identifica errores, si los hay. • Desarrolla laboratorios propuestos.		
4. Construye software constituido por una única clase que implementa persistencia, recursividad e interfaces gráficas simples bajo el enfoque orientado a objeto, apoyado por un entorno de programación, para acercarse a la resolución de problemas de programación.	4.1 Profesor: • Indica fuentes bibliográficas para lecturas sobre los conceptos de archivo, recursividad e interfaces gráficas. • Presenta apuntes de archivo, recursividad e interfaces gráficas y su implementación en un lenguaje orientado a objeto incluyendo ejemplos en que se muestra su uso, así como la confección de las correspondientes clases de prueba para crear y usar objetos que implementan estos conceptos. • Entrega listado de clases con y sin errores. • Entrega listados de ejercicios que involucren archivos, recursividad e interfaces gráficas. • Conduce laboratorios para construir y probar clases que usen archivos, recursividad e interface gráficas. • Publica tests en línea previos a clase. 4.2 Estudiante: • Lee los diversos materiales y apuntes que clases a clase va dando el profesor. • Responde tests en línea. • Realiza los ejercicios	• Identificación de elementos del lenguaje de programación que permiten la implementación y uso de archivos, recursividad e interfaces gráficas simples. • Corrección de errores, si los hay, en una clase dada. • Confección de una clase a partir de un enunciado. • Confección de una clase de prueba que cree objetos de alguna clase e invoque sus métodos, a partir de un enunciado. • Codificación de una clase y su clase de prueba en un entorno de programación en laboratorio. • Retroalimentación y apoyo. Nota: Todo lo anterior considerando los tópicos de archivos, recursividad e interfaces gráficas simples.	Horas presenciales: HT: 8 HP: 20 HL: 8 Horas de trabajo autónomo: HT: 10 HP: 40 HL: 10

Resultados de Aprendizaje	Actividades de Aprendizaje	Actividades de Evaluación	Tiempo Estimado
	propuestos que se relacionan con el tema tratado en cada apunte con y sin apoyo del profesor. • Contrasta sus soluciones con las dadas por el profesor y/o sus pares e identifica errores, si los hay. • Desarrolla laboratorios propuestos.		
5. Construye software constituido por múltiples clases usando un lenguaje de programación orientado a objeto para resolver problemas simples, a partir de diagramas de clase, utilizando un entorno de programación.	5.1 Profesor: • Entrega fuentes bibliográficas para lecturas sobre la implementación de relaciones entre clases en el lenguaje de programación utilizado en el curso. • Presenta apuntes sobre conceptos básicos de la arquitectura de capas para la construcción de software. • Entrega listado de clases relacionadas con y sin errores. • Entrega listados de ejercicios que involucren múltiples clases relacionadas. • Conduce laboratorios para construir y probar clases relacionadas. • Publica tests en línea previos a clases. • Presenta tarea del curso. 5.2 Estudiante: • Lee los diversos materiales y apuntes que clase a clase va dando el profesor. • Responde tests en línea. • Realiza los ejercicios propuestos que se relacionan con el tema tratado en cada apunte con y sin apoyo del profesor. • Contrasta sus soluciones	• Identificación de elementos del lenguaje de programación que permiten la implementación de relaciones entre clases. • Corrección de errores, si los hay, en una solución dada. • Confección de un conjunto de clases relacionadas a partir de un enunciado. • Definición y aplicación de casos de prueba en un entorno de programación en laboratorio. • Retroalimentación y apoyo.	Horas presenciales: HT: 12 HP: 30 HL: 12 Horas de trabajo autónomo: HT: 12 HP: 42 HL: 18

Resultados de Aprendizaje	Actividades de Aprendizaje	Actividades de Evaluación	Tiempo Estimado
	con las dadas por el profesor y/o sus pares e identifica errores, si los hay. • Desarrolla laboratorios propuestos. • Desarrolla tarea del curso.		

V. SISTEMA DE EVALUACIÓN

Tests grupales	10%
Test individuales	15%
Certámenes	45%
Laboratorios	20%
Tarea del curso	10%

Los tests grupales e individuales incluyen el desarrollo de ejercicios, mapas conceptuales, y elaboración de respuestas a preguntas basadas en lectura de apuntes, textos u otras fuentes en medio escrito o en línea.

VI. BIBLIOGRAFÍA

Fundamental

Joyanes, L. (2011). *Programación en Java 6*: Mc Graw Hill.

Shildt, H. (2011). *Java. A Beginner's Guide* (5ª ed.): McGraw Hill Osborne Media.

Complementaria

Deitel, P. y Deitel, H. (2009). *Java How to Program: Late Objects Version* (8ª ed.): Prentice Hall.