Facultad de Ingeniería - Universidad Nacional de Cuyo P1- PROGRAMA DE ASIGNATURA			
Asignatura:	**Programación Orientada a Objetos**		
Profesor Titular:	**César Omar Aranda**		
Carrera:	**Ingeniería en Mecatrónica**		
Año: 2015	**Semestre: 2**	**Horas Semestre: 60**	**Horas Semana: 4**

OBJETIVOS

Objetivos conceptuales:

- Reconocer los conceptos fundamentales del paradigma de programación orientado a objetos.
- Discernir los tipos de aplicación y las situaciones en las que es posible y necesario aplicar el paradigma orientado a objetos.
- Comprender básicamente el proceso de desarrollo de software orientado a objetos.

Objetivos procedimentales:

- Definir las características de la solución de un problema bajo el paradigma de la orientación a objetos.
- Codificar en lenguaje C++ a partir de interpretar una representación UML.
- Realizar POO aplicando buenas prácticas de Ingeniería de Software.

Objetivos actitudinales:

- Adquirir confianza en las propias posibilidades de comprender y resolver problemas.
- Manifestar perseverancia en las tareas a desarrollar.
- Ampliar la capacidad para trabajar de manera autónoma y grupal.
- Mejorar las habilidades de investigación así como su visión crítica y autocrítica de problemas y soluciones.

CONTENIDOS

UNIDAD 1: PROGRAMACIÓN ORIENTADA A OBJETOS

1.A. Aspectos Generales

Revisión de Paradigmas de programación. Conceptos generales. Lenguajes de programación. Ejemplos.

Análisis de inconvenientes de los enfoques solamente procedurales en la resolución de problemas complejos, en la reutilización de código y en el mantenimiento.

Programación Orientada a Objetos vs Basada en Objetos. Ventajas y desventajas.

Panorama de lenguajes orientados a objetos. Actualidad y tendencias

IDE, CASE y otras herramientas de desarrollo. Construcción de Aplicaciones OO.

1.B. Paradigma Orientado a Objetos

Conceptos fundamentales de la Orientación a Objetos. Objeto y Clase. Atributos. Operaciones.

Principios del paradigma orientado a objetos. Abstracción y Proceso de abstracción.

Encapsulamiento. Jerarquías de Clases. Polimorfismo. Modularidad.

Mecanismos de la orientación a objetos. Herencia. Tipos de Herencia. Instanciación. Ocultamiento. Comunicación entre objetos: Mensajes.

Ejemplos y ejercitación.

UNIDAD 2: Modelos Orientados a Objetos

2.A. UML

El lenguaje de Modelado Unificado UML. Generalidades. Aplicabilidad.

Elementos. Versiones. Diagramas Estructurales y de Comportamiento. Generalidades

Aplicabilidad y ejemplos.
Relaciones con el Manual del Usuario y/o del Sistema.

2.B. Diagramas de un proceso de modelado simplificado

Diagrama de Casos de Uso. Actor. Caso de Uso. Relaciones. Ejemplos.
Diagrama de clases. Relaciones. Asociación. Agregación. Composición. Ejemplos y ejercitación.
Diagramas de Secuencia y de Estados. Diagrama de Actividad. Ejemplos y ejercitación.

UNIDAD 3: Lenguaje de Programación

3.A. Fundamentos del Lenguaje C++

El lenguaje C++. Generalidades. Aplicabilidad. Versiones.
Sintaxis y semántica básica.
Tipos de datos. Operadores. Expresiones. Casting. Punteros.
Declaración de variables. Constantes. Ejemplos. Ejercitación.

3.B. Instrucciones Básicas en C++

Flujos estándares de entrada/salida. Tipos.
Estructuras de Control. Sentencias if, switch, for, while, do, break, exit, [goto].
Ejemplos. Ejercitación.

3.C. Programación de Objetos en C++

Tipos de datos compuestos.
Clases. Atributos y Métodos. Definición. Instanciación.
Métodos constructores y destructores.
Ejemplos y ejercitación.

3.D. Mecanismos Orientados a Objetos en C++

Clase Abstracta. Herencia Simple y múltiple. Visibilidad.
Polimorfismo y reutilización. Sobrecarga. Sobreescritura.
Ejemplos y ejercitación.

UNIDAD 4: Elementos Complementarios

4.A. Tipos de Datos Especiales en C++

Estructuras de datos como Objetos. Tipos Abstractos de Datos.
Objetos contenedores. Concepto. Tipos.
Colecciones de Objetos. Array. Vector. List. Stack. Queue.
Plantilla. Concepto. Utilidad.
Ejemplos. Ejercitación.

4.B. Depuración del software

Gestión de errores y excepciones.
Técnicas de depuración. Breaking. Debugging. Stepping. Tracing.
Ejemplos. Ejercitación.
Depuración vs. Prueba de Software.

4.C. Persistencia en C++

Persistencia: concepto y mecanismos.
Streams asociados. File. Acceso a dispositivos.
Ejemplos.

4.D. Extensiones

Librerías complementarias. Incorporación al proyecto y utilización.
Características básicas e introducción al uso de: gráficos, sonidos, bases de datos.
Gestión de dispositivos de entrada/salida, conectividad en red.
Ejemplos.

METODOLOGÍA DE ENSEÑANZA

Se considera que cada clase es eminentemente teórico-práctica, aplicando en forma inmediata los conceptos expresados teóricamente, por lo que el trabajo, en general, responderá al de un aula-taller.

Las clases teóricas se presentan sobre pizarra o con elementos multimediales, incluyendo ejemplos de aplicación de distinto nivel de completitud y complejidad.

Se plantean trabajos prácticos, casos de estudio, investigación y/o de extensión.

Algunos de ellos tienen carácter obligatorio (se solicita y califica tanto su desarrollo como su presentación) y otros opcionales (son sólo a efectos de ejercitación de los alumnos, de fijación de conceptos o discusión). Según las características del alumnado y del ciclo lectivo, los trabajos se distribuyen para ser llevados adelante de manera individual y/o grupal.

El seguimiento de alumnos se realiza registrando asistencia, cumplimiento y participación.

En principio el desarrollo de cada clase cuenta con tres momentos sucesivos:

- La introducción: Donde se expone teóricamente un tema nuevo o se realiza el análisis del código de un ejemplo simple ya resuelto.
- La elaboración: Donde se construye la solución a un problema acorde al tema explicado anteriormente y que permite se pongan de manifiesto las principales dudas conceptuales y prácticas.
- El cierre: Donde se analizan críticamente diferentes soluciones obtenidas y/o se incluyen conclusiones pertinentes al tema.

Muchos de los conceptos de UML, uso del IDE y principios de diseño y programación, son desarrollados durante el cursado insertos entre otros contenidos, de manera no secuencial (según el programa de contenidos). En las unidad 1 y 2 se plantean conceptos generales, que son trabajados de manera más profunda durante el desarrollo de las unidades 3 y 4.

Uso de mecanismos compartidos sobre redes y comunicaciones para la organización/distribución de los recursos de estudio así como para la realización de diversas actividades docentes complementarias a las propuestas de manera presencial.

En caso de que quedase algún tema sin dictar, se provee la bibliografía adecuada.

Actividad	Carga horaria por semestre
Teoría y resolución de ejercicios simples	30
Formación práctica	
Formación Experimental – Laboratorio	10
Formación Experimental - Trabajo de campo	0
Resolución de problemas de ingeniería	10
Proyecto y diseño	10
Total	**60**

BIBLIOGRAFÍA

Bibliografía básica

Autor	Título	Editorial	Año	Ejemplares en biblioteca
Deitel H. M. Y Deitel P. J.	Cómo Programar en C/C++ 6ª ed.	Prentice-Hall	2009	0
Stroustrup, B.	Programming. Principles and Practice Using C++	Pearson Education	2009	0
Ceballos Sierra, F.J.	C/C++ Curso de Programación, 3ª ed.	Ra-Ma	2007	0

Bibliografía complementaria

Autor	Título	Editorial	Año	Ejemplares en biblioteca
F. XHAFA, P. PAU VÁZQUEZ, A.J ORDI y otros	Programación en C++ para ingenieros	Thomson-Paraninfo	2006	0
Eckel, Bruce	Thinking In C++, Volumen 2	Mindview	2004	0
Ellis, M.A. y Stroustrup, B.	C++ Manual De Referencia Con Anotaciones	Díaz De Santos	2004	0
Ceballos Sierra, F. J.	Enciclopedia del lenguaje C++	AlfaOmega	2004	2

Acera García, M. A.	C/C++ (edición revisada y actualizada 2012)	Anaya Multim.	2012	0
Eckel, Bruce	Aplique C++	McGraw-Hill	1991	2
Schildt, H.	C/C++ Manual De Referencia	Anaya Multim.	1996	0

EVALUACIONES

Durante el período de clases, se prevé un sistema de evaluación continua consistente en un registro de los trabajos individuales y grupales realizados en clase.

Esto no implica la entrega de todos los trabajos al profesor por parte del alumno para su revisión. En el caso de la elaboración de programas propuestos, es el alumno quien debe lograr la habilidad de obtener software funcional y verificar la correctitud de su solución a partir del producto final obtenido. Mostrando finalmente sus resultados o conclusiones.

En base a las características del grupo y tiempos disponibles se agregará la elaboración de trabajos monográficos para su entrega y/o exposición.

Se prevé realizar 1 (una) evaluación de carácter global (esto es, abordando contenidos conceptuales y prácticos que corresponden a un período indicado en clase oportunamente), con una única instancia de recuperación, también global y acumulativa en contenidos.

Se proponen además una serie de actividades de desarrollo obligatorio para seguimiento y control. Entre ellas un Trabajo Práctico Especial de desarrollo y presentación individual o grupal según la población del curso.

Todas las evaluaciones deben dejar una constancia documental, es decir almacenadas en un archivo o escritas en papel. Esta evaluación (o una copia) es devuelta al alumno con las correcciones pertinentes. El criterio de evaluación a seguir, es adelantado (informado) en clase.

Para obtener la regularidad o la promoción directa en la asignatura se tienen en cuenta las calificaciones obtenidas tanto en la evaluación ya indicada como en las obtenidas del seguimiento continuo.

Para la aprobación de la asignatura, en condición de alumno regular, se propone la presentación y defensa individual de un producto software elaborado individualmente.

Éste producto consiste en la implementación de una aplicación que debe ejecutar libre de fallas (jamás terminar anormalmente), aplicando la totalidad de los conceptos fundamentales aprendidos en clase y en la confección de un manual de usuario conteniendo anexa la documentación de la aplicación.

El “tema/objetivo” del producto software es propuesto por el alumno y acordado con el profesor con una prudente anticipación a la fecha elegida para el examen final.

La defensa consiste en una exposición coloquial sobre los conceptos fundamentales y estrategias utilizadas en la confección del producto. La finalidad de esta exposición no es otra que la de corroborar autoría.

Para la aprobación de la asignatura, en condición de alumno libre, se propone la resolución (equivale a la presentación/defensa de la condición anterior) individual de un producto software acotado, asignado por el profesor en el momento del examen.

La resolución se debe hacer en un tiempo acotado máximo de 1 hora y media. La misma incluye clases de usuario, y precisa aplicar algunos de los conceptos fundamentales de este programa de estudio (como mecanismos de herencia y polimorfismo, almacenamiento de información usando clases de colección de C++, persistencia, sobrecarga de operadores, plantillas, u otros) según la consigna.

Como el problema se diseña e implementa en el momento no hace falta un coloquio de defensa, salvo alguna pregunta para aclarar porqué se adoptó una solución y no otra.

El programa debe compilar y ejecutar sin errores. Una vez comprobado esto último, la calificación se asigna analizando el diseño de la implementación realizada.

Con respecto a otros detalles relacionados con adquirir la regularidad y/o la aprobación de la asignatura se siguen los lineamientos generales fijados para la carrera, tanto de tipo académico como administrativos.

Programa de examen

No Aplicable