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: 2022	**Semestre: 8**	**Horas Semestre: 60**	**Horas Semana: 4**

CONTENIDOS MÍNIMOS (Ord 33/2009-CS)

Análisis de inconvenientes de los enfoques procedurales en la resolución de problemas complejos, la reutilización de código y el mantenimiento. Conceptos básicos de POO: Tipos Abstractos de Datos. Encapsulamiento. Ocultamiento. Mensajes y Métodos. Clases e Instancias. Jerarquías de Clases. Herencia. Polimorfismo.

El lenguaje de Modelado Unificado UML. Diagramas estructurales, funcionales y de Casos de Uso. Panorama de lenguajes orientados a objetos. El lenguaje C++. Estructuras de Control. Clases y Métodos. Estructuras de datos como Objetos. Objetos contenedores. Colecciones de Objetos. Entornos y herramientas. Aplicaciones.

OBJETIVOS

Conforme al Plan de Estudios detallado en la Ord 033/2009 –CS. De la Carrera de Ingeniería en Mecatrónica, Acreditada por la Comisión Nacional de Evaluación y Acreditación Universitaria (Resolución CONEAU Número: 470/11). Ministerio de Educación Ciencia y Tecnología.

Objetivos Generales de la Carrera de Ingeniería en Mecatrónica:

- Actuar con sentido crítico e innovador en la problemática de los sistemas electromecánicos y proponga respuestas originales y alternativas pertinentes.
- Disponer de una eficiente formación teórica y formación práctica que permita iniciarse en sus actividades profesionales con idoneidad y disposición de capacitación permanente, ubicando e identificando las informaciones adecuadas.
- Poseer los suficientes recursos técnicos y metodológicos que lo habiliten conducir tareas de su especie, integrar y conducir equipos de trabajo.

Objetivos del Área de Tecnologías Aplicadas

- Aplicar el conjunto de técnicas que definen la actividad primordial del Ingeniero en Mecatrónica.
- Adquirir la capacitación metodológica específica y el pensamiento crítico y creador en el trabajo.
- Consolidar los aprendizajes para acceder a los problemas con visión de integración multidisciplinaria.

- Realizar experiencia práctica integral y directa de lo que será el futuro quehacer del graduado.
- Desarrollar la capacidad para la autoformación permanente.
- Integrar la capacidad y el esfuerzo profesional en conductas de compromiso social frente a los desafíos de la actividad contemporánea.
- Proporcionar una docencia que enfatice el aprender haciendo.

Objetivos de la asignatura PROGRAMACIÓN ORIENTADA A OBJETOS

Expectativas de logro:

- Interpretar modelos orientados a objetos sencillos representados mediante UML
- Realizar implementaciones bajo paradigma de orientación a objetos, de soluciones de complejidad variable en lenguajes OO (como C++, Java u otro/s).
- Ser capaz de incorporar funcionalidades de complejidad media o avanzada de un lenguaje (como manejo de comunicaciones, manipulación de gráficos/sonido, gestión de procesos u otro).

Objetivos específicos conceptuales:

- Comprender el paradigma de la orientación a objetos, sus características, ventajas y ámbitos de aplicación.
- Comprender básicamente el proceso de desarrollo de software orientado a objetos.
- Identificar las entidades que participan en un sistema y sus comportamientos asociados.

Objetivos específicos procedimentales:

- Definir las características de la solución de un problema bajo el paradigma de la orientación a objetos.
- Representar sistemas usando diagramas y elementos básicos de UML
- Implementar soluciones en lenguajes orientados a objetos, a partir de transcribir modelos representados mediante UML.

Objetivos específicos actitudinales:

- Cumplir en tiempo y forma con las actividades prácticas propuestas.
- Expresarse de manera técnicamente correcta en el ámbito de desarrollo de software.
- Trabajar de manera autónoma o grupal, según corresponda.
- Investigar sobre un tema específico con sentido crítico y demostrar autocrítica respecto de sus propias soluciones a un problema dado.

CONTENIDOS

UNIDAD 1: ORIENTACION A OBJETOS

1.A. Aspectos Generales

Paradigmas de programación. Conceptos generales. 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.

Diseño e Implementación. Beneficios y limitaciones de la generación automática de software. Roundtrip.

1.B. Paradigma Orientado a Objetos

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

Principios y mecanismos del paradigma orientado a objetos. Abstracción. Encapsulamiento. Jerarquías de Clases. Polimorfismo. Modularidad. Instanciación. Ocultamiento. Comunicación entre objetos: Mensajes. Herencia. Tipos de Herencia.

Ejemplos y ejercitación.

UNIDAD 2: PROCESO DE DESARROLLO

2.A. Metodología de construcción de software OO

Proceso de Modelado Orientado a Objetos. Conceptos. Fases. Generalidades. Ejemplos.

Principios de diseño. Patrones de Diseño: concepto y generalidades.

Documentación de un sistema. Relación con el Manual del Usuario.

2.B. UML

Lenguaje de Modelado Unificado. Generalidades. Aplicabilidad.

Diagrama de Casos de Uso. Actor. Caso de Uso. Relaciones.

Diagrama de Clases. Relaciones. Asociación. Agregación. Composición.

Diagrama de Secuencia. Diagrama de Estados.

Ejemplos y ejercitación.

UNIDAD 3: PROGRAMACIÓN ORIENTADA A OBJETOS

3.A. Elementos fundamentales del lenguaje

Análisis comparado de similitudes y diferencias del lenguaje C++ con el lenguaje C.

Revisión de generalidades, sintaxis, semántica y elementos básicos de los lenguajes C++ y Python. Tipos de datos simples y compuestos. Operadores. Expresiones. Conversión de tipo. Declaración de variables. Constantes. Punteros y referencias.

Flujos estándares de entrada/salida. Sentencias fundamentales de flujo secuencial y de control de flujo. Gestión de errores y excepciones. Ejemplos. Ejercitación.

3.B. Implementación del paradigma de objetos en lenguaje C++

Clases. Atributos y Métodos. Definición. Instanciación. Visibilidad.

Métodos constructores y destructores.

Clase Abstracta. Herencia simple y múltiple. Reutilización. Colecciones.

Polimorfismo. Sobrecarga de métodos. Sobrecarga de operadores. Reescritura de métodos.

Ejemplos. Ejercitación.

3.C. Implementación del paradigma de objetos en lenguaje Python

Clases y objetos nativos en Python. Tipos básicos y colecciones: cadenas, listas, tuplas y diccionarios. Funciones.

Clases personalizadas. Atributos y Métodos. Definición. Instanciación.

Métodos especiales. Métodos constructores y destructores.

Herencia simple y múltiple. Reutilización. Polimorfismo.

Consideraciones particulares de OO en Python. Sobrecarga, encapsulamiento y control de acceso.

Ejemplos. Ejercitación.

3.D. Librerías especiales y programación avanzada

Librerías complementarias. Incorporación y utilización en un proyecto.

Persistencia: concepto. Serialización. Archivos. Bases de datos.

Gestión de dispositivos de entrada/salida. Conectividad TCP/IP.

Introducción al uso de: menús de usuario en consola, aplicaciones cliente/servidor, interfaces gráficas de usuario, mecanismos RPC.

Ejemplos. Ejercitación.

TRABAJOS PRÁCTICOS

TP 1 / introducción + fundamentos:

Implementaciones de aprendizaje usando clases, objetos, mensajes, herencia, agregación y polimorfismo, a partir de consignas con modelos sencillos representados mediante UML.

TP 2 / Integrador:

Desarrollo de un proyecto integrador que incluye:

- Representaciones UML
- Implementación de una aplicación Cliente/Servidor integrando lenguajes OO (en particular: Python y C++)
- Manipulación de archivos y comunicaciones en red
- Informe técnico

Listado de Trabajos Prácticos

Las guías con las consignas del desarrollo práctico, así como los ejemplos de resolución de problemas y guías con los procedimientos de operación, se encuentran disponibles en formato digital dentro del curso de la asignatura, ubicado en https://aulaabierta.ingenieria.uncuyo.edu.ar

METODOLOGÍA DE ENSEÑANZA

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

Clases teóricas sobre pizarra, elementos multimedia y/o ambientes colaborativos en red de computadoras para la presentación de conceptos, teorías y ejemplos de aplicación (con distinto nivel de completitud y complejidad, según corresponda).

Clases prácticas destinadas a aplicar y combinar los diferentes conceptos, ya sea en aula común sobre computadoras de escritorio, o bien en aula especial haciendo uso de tableros educativos especialmente construidos, disponibles en el Laboratorio de Control.

Las prácticas se encuentran distribuidas en diferentes secciones, algunas de carácter obligatorio (se requiere de su desarrollo y presentación) y otras opcionales (sugeridas a efectos de ejercitación de los estudiantes, para la fijación de conceptos o discusión).

Las prácticas se complementan con la elaboración de informes.

Resulta de interés especial la elaboración de un informe con la propuesta de software OO, realizado de manera grupal y según la disponibilidad de recursos.

En la resolución de problemas de los temas fundamentales se requiere, en algunos casos, no sólo la aplicación de conocimientos propios de la asignatura sino también de las ciencias básicas y de otras tecnologías propias a la Carrera.

Adicionalmente, en caso que los tiempos del ciclo lectivo particular lo permitan, y según las características del alumnado y de la disponibilidad de recursos, se distribuyen casos de estudio para ser llevados adelante de manera individual y/o grupal.

Además de las calificaciones que surgen de las evaluaciones objetivas, el seguimiento de los estudiantes se realiza en base al registro de asistencia y al cumplimiento de fechas y formas.

Muchos de los conceptos de UML, principios de diseño y de programación, son desarrollados durante el cursado insertos entre otros contenidos, y no de manera secuencial como se detallan en los contenidos del programa de estudio. En las unidades 1 y 2 se plantean conceptos generales, que son trabajados de manera específica y con mayor extensión

durante el desarrollo de la unidad 3.

Se usan mecanismos compartidos sobre redes y comunicaciones tanto para la organización y distribución de los recursos de estudio, como para la realización de diversas actividades docentes complementarias a las propuestas de manera presencial.

Si algún tema quedase sin dictar y se considera que resulta necesario para aquel estudiante que se presenta rendir un examen final, al finalizar el ciclo lectivo se provee la bibliografía adecuada para abordar el mismo.

DISTRIBUCIÓN DE LA CARGA HORARIA

Actividad	Carga horaria por semestre
Teoría y resolución de ejercicios simples	20
Formación práctica	
Formación Experimental – Laboratorio	15
Formación Experimental - Trabajo de campo	0
Resolución de problemas de ingeniería	20
Proyecto y diseño	5
Total	**60**

Estimación del Tiempo del Estudiante

Los tiempos que establece el Plan de Estudio corresponden a Horas Presenciales. Las mismas son las reflejadas en el cuadro anterior.

Se considera que el estudiante debe agregar Horas no Presenciales (aproximadamente un 35% adicional) para adquirir las Competencias esperadas, mediante la resolución individual o grupal de Actividades (prácticas) no Presenciales.

Porcentaje de Horas Presenciales	67 % del Total
Porcentaje de Horas a Distancia	33 % del Total

Esta asignatura puede ser dictada totalmente bajo estrategias de Educación a Distancia. En este caso la cantidad de Horas No Presenciales se pueden transformar en Horas No Presenciales y alcanzar el 100%.

Recursos Necesarios

Espacios Físicos: aula en el edificio de clases para las actividades teóricas y sala del Laboratorio de Control ubicado en DETI I para algunas de las actividades prácticas.

Equipamiento informático personal (en caso de necesidad puede usarse el disponible en la

Facultad).

Espacios o Recursos tecnológicos: proyector multimedia como apoyo durante las explicaciones de contenidos teóricos,

Entornos integrados de desarrollo para la programación, depuración y ejecución.

Aulas virtuales y mecanismos de videoconferencia para la interacción a distancia.

Plan de Contingencia

Los cambios en la planificación que surgieran por algún imprevisto, serán informados a los estudiantes a través del aula virtual ya mencionada.

Los estudiantes podrán plantear y resolver sus dudas e inquietudes en los horarios de consulta o por mail a cesar.aranda@ingenieria.uncuyo.edu.ar

En el caso que la asignatura deba dictarse de manera completamente virtual, la modalidad a distancia adoptada dependerá de las condiciones que se presenten y de las herramientas reales disponibles. Básicamente, el dictado se realizará mediante clases pregrabadas en formato de video para los temas teóricos, así como documentos digitales y videos complementarios para los temas prácticos. Todo organizado y provisto a través de aula virtual. El trabajo de interacción con los alumnos se realizará mediante diversas herramientas disponibles en la plataforma de aulas virtuales de la Facultad. Se guiará y acompañará todo mediante reuniones por videoconferencia y chat.

BIBLIOGRAFÍA

Bibliografía básica

Autor	Título	Editorial	Año	Ejemplares en biblioteca
DEITEL H. M. Y DEITEL P. J.	Cómo Programar en C/C++ (6ª edición)	Prentice-Hall	2009	0
STROUSTRUP, B.	Programming. Principles and Practice Using C++	Pearson Education	2009	0
BRONSON, Gary	C++ para Ingeniería y Ciencias, 2da edición	Cengage Learning	2007	0
VON ROSSUM, G.	El Tutorial de Python (v3)	Python Software Foundation	2017	(1)
PEREZ CASTAÑO, A.	Python Fácil	Marcombo	2016	0
BEAZLEY, D. y JONES, B.	Python Cookbook	O'Reilly Media. 3ra edición	2013	0

Bibliografía complementaria

Autor	Título	Editorial	Año	Ejemplares en biblioteca
F.XHAFA, P.PAU VAZQUEZ, A.JORDI y otros	Programación en C++ para ingenieros	Thomson-Paraninfo	2006	0
CEBALLOS SIERRA, F.J.	C/C++ Curso de Programación, 3ª edición	Ra-Ma	2007	0
ECKEL, Bruce	Thinking In C++, Volumen 2	Mindview, Inc	2004	0
ELLIS, M.A. y	C++ Manual De Referencia Con	Ed. Diaz De	2004	0

STROUSTRUP, B.	Anotaciones	Santos		
CEBALLOS SIERRA, F.J.	Enciclopedia del lenguaje C++	AlfaOmega	2004	2
ACERA GARCIA, M.A.	C/C++ (edición revisada y actualizada 2012)	Anaya Multimedia	2012	0
ECKEL, Bruce	Aplique C++	McGraw Hill	1991	2
VON ROSSUM, G.	El Tutorial de Python (v2)	Python Software Foundation	2009	(1)
GONZÁLEZ DUQUE, R.	Python para todos	(2)	2008	(3)
B. Forouzan	Transmisión de Datos y Redes de Comunicaciones	McGraw Hill	2002	0

(1) Disponible en http://tutorial.python.org.ar/

(2) Distribuido bajo una licencia Creative Commons Reconocimiento 2.5 España

(3) Disponible en http://mundogeek.net/tutorial-python/

EVALUACIONES (S/ Ord. 108-10_CS)

Durante el período de clases, se prevé un régimen de evaluación continua.

Se proponen una serie de trabajos individuales y grupales, realizados total o parcialmente en clase.

Algunas de las actividades están destinadas a la práctica del Saber Ser, a través de la adquisición del Saber Conocer y de la ejercitación en el Saber Hacer. Pudiendo dar lugar a procesos curriculares horizontales e interdisciplinares, informados oportunamente.

Todos los trabajos prácticos deben ser realizados, pero sólo algunos de ellos poseen calificación especial.

La entrega de los trabajos al profesor por parte del estudiante no implica la revisión de los mismos, en particular aquellos referidos a programación. En general, es el estudiante quien debe lograr la habilidad de obtener software funcional y, con sentido crítico, verificar la correctitud de su solución a partir del producto final obtenido. De manera individual o grupal, según corresponda, se solicitará mostrar los resultados o las conclusiones obtenidas.

Los trabajos prácticos no se recuperan, pero el estudiante dispone de tiempo suficiente para la elaboración, las consultas y la presentación del informe correspondiente hasta la semana anterior a la de recuperación de parciales.

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 al menos 2 (dos) evaluaciones teórico-prácticas.

Al menos una de ellas consiste en un cuestionario, con una única instancia de recuperación, abordando contenidos conceptuales y de aplicación práctica que corresponden a un período indicado en clase oportunamente.

Estas evaluaciones pueden ser realizadas tanto de manera tradicional como de manera virtual utilizando herramientas de Educación a Distancia (ya sea las provistas por la plataforma de aulas virtuales como externas).

La última evaluación corresponde a un trabajo integrador de programación, que dada la modalidad de su desarrollo (presenta varias instancias de revisión). Este Trabajo Práctico será

resuelto y presentado al resto del curso de manera individual o grupal, dependiendo de la población del curso. En la evaluación de este trabajo, además de los producidos formales (individuales/grupales), se tiene en cuenta el desempeño individual durante el coloquio conceptual.

Todas las instancias de evaluación dejan una constancia documental, preferentemente en formato digital (archivos). Las observaciones y/o calificaciones realizadas, son devueltas al alumno para su conocimiento o actividad de corrección pertinente.

El criterio de evaluación a seguir, es adelantado (informado) en clase.

Para adquirir la regularidad y/o la aprobación de la asignatura se siguen los lineamientos generales fijados para la carrera, tanto académicos como administrativos.

Para la promoción directa de la asignatura se debe:

- Cumplir con la asistencia obligatoria (75% de las clases)
- Participar en el desarrollo del 100% de los trabajos prácticos.
- Aprobar los trabajos prácticos obligatorios con calificación igual o superior a 6 (seis)
- Aprobar las evaluaciones teórico-prácticas o su recuperación, con calificación igual o superior a 6 (seis).

A finalizar el cursado, cada estudiante tiene diferentes calificaciones, que se promedian de manera simple.

En el caso de hacer uso de la evaluación de recuperación, la calificación obtenida reemplaza en la ecuación a la/s evaluación/es que corresponda/n.

Cualquier estudiante que curse de manera regular puede obtener la promoción directa, ya sea que apruebe en la primera instancia o en la recuperación.

La condición de nivel que debe satisfacer el estudiante, para que ello ocurra, es obtener como resultado de la ecuación anterior, una calificación final mínima de 7 (siete).

En caso de no cumplir con la asistencia, o haber aprobado menos del 80% de los prácticos, o haber logrado una calificación final igual a 6 (seis), el estudiante obtiene la regularidad en la asignatura.

La fecha límite para obtener la promoción directa o la regularidad se corresponden con la finalización del cursado de la materia en el ciclo lectivo correspondiente.

Para adquirir la **Regularidad en la asignatura**, y por lo tanto adquirir el derecho al examen final, se siguen los lineamientos generales fijados para la carrera, tanto académicos como administrativos.

El alumno se encuentra en condición de **Libre en la asignatura**, cuando no satisface los mínimos indicados. Aunque, en casos de cumplimiento parcial, sea de la asistencia mínima, o haber aprobado sólo un porcentaje de los prácticos, o haber logrado calificaciones inferiores a los topes indicados, se puede analizar la situación particular conforme al contexto y determinar los pasos a seguir.

Para la **Aprobación de la asignatura en condición de Estudiante Regular**, se propone para el examen final la presentación y defensa individual de un producto software elaborado individualmente (puede ser la continuación del Trabajo Especial realizado durante el cursado).

Este 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 informe técnico de la aplicación que incluye

elementos conceptuales, elementos de diseño y un manual de usuario.

El “objetivo” del software y los temas abordados pueden ser propuestos por el estudiante y acordados con el profesor con una prudente anticipación a la fecha elegida para el examen final, tanto para permitir la corrección como la aprobación previa del producto resultante.

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, aunque deben cuidarse en las explicaciones el uso adecuado de la terminología y la aplicación correcta de los conceptos teóricos presentes en el programa de estudio.

Para la **Aprobación de este espacio curricular en condición de Alumno Libre**, se debe elaborar, presentar y aprobar un Trabajo Especial similar al dado en el cursado.

Sin embargo, para acceder a la presentación y coloquio correspondientes (en condiciones similares al alumno Regular), y dado que la asignatura está orientada a desarrollar habilidades de programación bajo un paradigma específico el estudiante debe demostrar las mismas, resolviendo previamente una consigna de programación de desarrollo individual sobre un producto software acotado, asignado por el profesor en el momento del examen y resuelto en un tiempo máximo de 1 hora.

El caso provisto, consiste en un breve texto descriptivo del problema a resolver acompañado de una o más representaciones UML. Implica tanto el uso de clases propias del lenguaje como de clases definidas por el programador, 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, persistencia, sobrecarga de métodos, sobrecarga de operadores, plantillas, u otros) según la consigna.

El problema se diseña e implementa en el momento del examen y no requiere de un coloquio de defensa.

Más allá que el programa deba compilar y ejecutar sin errores, es fundamental para su aprobación demostrar que se han aplicado de manera correcta los conceptos y mecanismos tanto de la OO como de el/los lenguajes indicados (Python/C++). Una vez comprobado esto último, la calificación se asigna analizando la implementación realizada.

OBSERVACIONES ESPECIALES:

a. En la asignatura no se contempla la situación de cursado como estudiante Libre, esto significa que el estudiante que desee rendir en condición de Libre puede asistir a clases, pero sólo en calidad de oyente. Para un estudiante en esta condición no se califican sus evaluaciones, no se registra su asistencia y (en principio) no puede hacer uso del equipamiento especial de laboratorio, destinado a los estudiantes que cursan en condición Regular.
b. En la asignatura, y para cualquier condición de cursado, tampoco se registran (guardan) calificaciones de manera provisoria a la espera de que se cumplan las correlativas que correspondan. Eso queda supeditado a lo que permita registrar y administrar el sistema de gestión de estudiantes y cátedras de la Facultad.
c. En caso que exista imposibilidad de evaluaciones presenciales, tanto de parciales como finales, la evaluación se podrá realizar de manera virtual. El examen en esta modalidad es similar al descrito para la presencialidad.

PROGRAMA DE EXAMEN

No aplicable (el examen consta de igual consigna base para alumnos libres y tema de coloquio abierto según desarrollo individual).

Ing. César Omar Aranda

6 de Julio de 2022

FECHA, FIRMA Y ACLARACIÓN TITULAR DE CÁTEDRA