FCEFyN
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS y NATURALES

UNC Universidad Nacional de Córdoba

Asignatura: **Informática Avanzada**

Código: 10-09700	RTF	6
Semestre: Tercero	Carga Horaria	72
Bloque: Tecnologías Aplicadas	Horas de Práctica	48
Departamento: Computación		

Correlativas:
- Informática y Cálculo Numérico

Contenido Sintético:
- Programación avanzada.
- Interfaces de software.
- Librerías.
- Herramientas para desarrollo de programas.
- Diseño de aplicaciones de software.
- Pruebas y corrección de errores.
- Performance.
- Componentes de un sistema operativo.

Competencias Genéricas:
- CG4: Utilizar de manera efectiva las técnicas y herramientas de aplicación en ingeniería.
- CG5: Contribuir a la generación de desarrollos tecnológicos y/o innovaciones tecnológicas.

Aprobado por HCD: 1054-HCD-2013 RES: Fecha: 29/11/23

Competencias Específicas:

CE1.1: Diseñar, proyectar y calcular sistemas, equipos y dispositivos de generación, transmisión y/o procesamiento de campos y señales analógicos y digitales; circuitos integrados; hardware de cómputo de propósito general y/o específico y el software a él asociado; hardware y software de sistemas embebidos y dispositivos lógicos programables; sistemas de automatización y control; sistemas de procesamiento y de comunicación de datos y sistemas irradiantes, para brindar soluciones óptimas de acuerdo a las condiciones técnicas, legales, económicas, humanas y ambientales.

CE1.2.1: Modelar matemáticamente problemas de ingeniería, hallar soluciones específicas empleando algoritmos matemáticos y herramientas informáticas, y generalizar las soluciones para resolver situaciones reales de ingeniería.
CE1.2.2: Manejar herramientas informáticas que permitan el desarrollo de soluciones en el ámbito de la ingeniería.

CE1.4.3: Analizar, diseñar, programar, implementar, probar, depurar y evaluar hardware y software para sistemas de computación de propósitos específicos.
CE1.4.4: Analizar, diseñar, programar, implementar, probar, depurar y evaluar hardware y software para sistemas de computación de propósitos generales.
CE1.4.6: Analizar, diseñar, programar, implementar, probar, depurar y evaluar sistemas de procesamiento de datos (hardware/software).

CE3.1: Validar y certificar el funcionamiento, condición de uso o estado de los sistemas mencionados anteriormente.

Presentación

La Programación Orientada a Objetos se inicia con el sistema de simulación denominado Simula en los años 60 y posteriormente recibe un gran impulso teórico de sus fundamentos con el grupo de investigación del Xerox Parc Place y su desarrollo del lenguaje Smalltalk en los 70. La pureza del paradigma implementado en Smalltalk se ha convertido en su punto débil ya que se debe abandonar completamente el paradigma procedural y también la sintaxis de los lenguajes descendientes del Algol como Pascal y particularmente la familia del C, constituida por el C mismo, C++, C# y el Java. Un caso particular es el de Eiffel, más cercano a las ideas del Pascal y Algol, pero implementa el paradigma de objetos en forma pura, lo que lo lleva ser de difícil distribución masiva, no obstante, muchas de sus características han sido implementadas actualmente en Java y C#.

Otro aspecto que ha colaborado en la difusión del paradigma orientado a objetos han sido los Sistemas Operativos con interfaces gráficas de usuario en las cuales la metáfora de los objetos y los eventos permite elaborar una interacción con el usuario en forma natural y flexible.

Simultáneamente con el crecimiento de la Programación Orientada a Objetos se da el Análisis y Diseño Orientados a Objetos hasta desembocar en el Proceso Unificado de Desarrollo y el Lenguaje de Modelado (UML) que permite actualmente disponer de una Metodología que guíe el proceso de diseño e implementación. Al igual que en los proyectos de ingeniería, se ha descubierto que existen patrones de diseño a partir de los cuales se puede construir el software sin necesidad de tener que reinventarlo y, además, se ha consolidado la construcción de software a partir de componentes reusables durante los 90 y principios de este siglo. Actualmente, el desarrollo profesional de software se somete a estos principios que denominamos Ingeniería de Software.

La asignatura se dicta en el tercer semestre (segundo año) de la carrera Ingeniería Electrónica, y tiene como objetivo principal brindar las herramientas y capacidades básicas para que el estudiante, luego de aprobar la materia, pueda aprender en forma autónoma la programación en cualquier lenguaje avanzado de alto nivel.

Se estudian los fundamentos de la programación de alto nivel en lenguajes avanzados y actuales, los paradigmas más utilizados y las nuevas tecnologías para desarrollo de software. Además de la práctica de programación de software, se estudian conceptos y fundamentos involucrados en el desarrollo de sistemas informáticos: sistemas operativos; sus características, tipos y funciones; testing; ensayos y pruebas de performance; certificación de software; herramientas para desarrollo de software; etc. Todos los contenidos se abordan desde la perspectiva de ingeniería de software.

En cuanto a la pedagogía, se centra en el estudiante mediante un enfoque por competencias y se trabaja principalmente desde la práctica.

Contenidos

Sistemas Operativos

Historia de los Sistemas Operativos. Funciones del Sistema Operativo. Interfaces con el usuario: intérprete de comandos (shell) y entornos gráficos. Interfaces con las aplicaciones: llamadas al sistema. Descomposición en capas del Sistema

Operativo. El núcleo del Sistema Operativo. Descripción y funciones básicas. Ingeniería de Software. Enfoque desde la Ingeniería de Software.

Introducción a los lenguajes de programación
Historia de los lenguajes de programación y los distintos paradigmas. Comparación entre intérpretes y compiladores; fases de la traducción de lenguajes; aspectos independientes y dependientes de la máquina. El concepto de máquina virtual, jerarquía de máquinas virtuales, lenguajes intermediarios

Objetos y Clases
Definición de clases y de objetos. Llamado de métodos y sus parámetros. Tipos de datos. Instancia de objetos y estado. Interacción básica entre objetos. Objetos como parámetros de métodos. Declaración y definición de clases: Campos, constructores y métodos.

Interacción entre objetos
Abstracción y modularización de software. Diagramas de clases y de objetos. Tipos primitivos y tipos de objetos. Creación de objetos y constructores múltiples. Llamado interno y externo de métodos. La autorreferencia a un objeto.

Agrupamiento de objetos
Agrupamiento de objetos en colecciones de tamaño flexible. Elementos básicos de una biblioteca de clases. Clases genéricas. Enumeración y gestión de colecciones. Iteradores. Colecciones fijas de objetos. Información y ocultamiento de campos y métodos. Variables de clase y constantes. Documentación de bibliotecas de clases estándar. Interfaces o implementación de métodos. Lectura y redacción de documentación de clases parametrizadas. Programación defensiva. Manejo de excepciones. Reporte de errores.

Prueba, depuración y mantenimiento de software
Pruebas de unidad. Inspectores. Pruebas positivas y negativas. Automatización de las pruebas. Pruebas de regresión. Escenarios y registro de pruebas. Uso de depuradores. Uso de aserciones. Performance. Validación y certificación de software.

Diseño de clases
Introducción al acoplamiento y la cohesión. Duplicación y extensión de código. Uso de encapsulado. Diseño basado en responsabilidades. Refactorización. Guías de diseño.

Herencia de clases
Jerarquías de herencia de campos y métodos de clases. Derechos de acceso a la herencia. Inicialización de instancias con herencia. Superclase y tipo de dato. Subtipos y subclases. Variables polimórficas. Conversión de tipo (Casting). Tipos estáticos y dinámicos. Sobrecarga de métodos. Búsqueda dinámica de métodos.

Llamado a la superclase en los métodos. Polimorfismo de métodos. Acceso protegido. Clases abstractas. Métodos abstractos. Interfaces. Interfaces como tipos. Interfaces como especificaciones.

Metodología de enseñanza

Cada nuevo tema se aborda mediante clases expositivas y exposición dialogada, a fin de introducir a los estudiantes en la temática.
Los estudiantes deben afrontar diversas actividades prácticas con los saberes conceptuales y procedimentales adquiridos previamente. El docente sigue el proceso y orienta al estudiante mediante preguntas guías, interviniendo en los casos que observe que el rumbo tomado por el equipo de trabajo se desvía de los caminos que permiten arribar a la consecución del mismo.
La metodología se basa fuertemente en la práctica y en el aprender haciendo. Se estimula en los estudiantes el estudio de nuevas tecnologías.

Evaluación

Los estudiantes deben demostrar los conocimientos y habilidades adquiridas en tres tipos de instancias de evaluación:

Evaluación conceptual (EC): Consiste en una serie de evaluaciones breves de ejercicios que permitan demostrar comprensión de temas puntuales desarrollados en las clases anteriores. La evaluación conceptual consiste en ejercicios de programación que serán evaluados objetivamente mediante pruebas de software basadas en test unitarios, verificando el cumplimiento de los requerimientos, de forma automática a través del aula virtual. Para aprobar el conjunto de todas las Evaluaciones Conceptuales, se debe alcanzar un rendimiento igual o superior al 60% del puntaje total. Se prevé un recuperatorio del 30% del puntaje total que se considera como puntaje adicional y se suma a los ya obtenidos, no pudiendo pasar la suma del 100% del puntaje.

Trabajo Práctico (TP): Consiste en una serie de actividades de desarrollo de la solución algorítmica a problemas propuestos por la Cátedra. Las soluciones deben estar libres de errores sintácticos que impidan su compilación y generación de código ejecutable. La solución propuesta por el estudiante se evalúa objetivamente mediante pruebas de software basadas en test unitarios, verificando el cumplimiento de los requerimientos, de forma automática a través del aula virtual. Para aprobar cada trabajo práctico, se debe alcanzar individualmente un rendimiento igual o superior al 60%. Estas actividades son de carácter extra-áulico, para las cuales los alumnos disponen de un mínimo de 72 horas para su resolución y envío. Si bien cada trabajo puede favorecer el desarrollo de una determinada competencia en particular, y es de esperar la evidencia de

esto hacia la conclusión de dicha actividad, la evaluación es contínua a lo largo de todas las actividades propuestas.

Examen de Promoción (EP): en la última clase, se realiza una ejercitación consistente en el desarrollo de un programa correspondiente al enunciado de un algoritmo. La implementación debe estar libre de errores sintácticos que impidan su ejecución por parte del compilador, como requisito para su evaluación. La solución propuesta por el estudiante se evalúa objetivamente mediante pruebas de software basadas en test unitarios, verificando el cumplimiento de los requerimientos, de forma automática a través del aula virtual. Para ser aprobado, se debe alcanzar un rendimiento igual o superior al 60%.

Al final del semestre, cada estudiante debe haber demostrado un nivel de desarrollo mínimo de las competencias propuestas a través de los resultados de aprendizaje propuestos.

Condiciones de aprobación

Condiciones de regularización

El estudiante alcanza la condición de regular al cumplir las siguientes condiciones:

1. Asistir al 80% de las clases. La asistencia es registrada en el aula virtual mediante la presentación del 60% de las EC y del 60% de los TP.
2. Alcanzar un rendimiento global igual o superior al 60% de los puntos en las EC. La presentación a la EC confiere asistencia a clase.
3. Aprobar el 60% de los TP propuestos.

Condiciones de promoción

El estudiante alcanzará la condición de promocionado al cumplir las siguientes condiciones:

1. Alcanzar la condición de alumno regular para lo cual se deben cumplir las condiciones indicadas anteriormente.
2. Aprobar la actividad EP.

Cumplidas estas condiciones, la nota final de promoción se calcula según la siguiente fórmula:

Nota Final = redondear((0,3 * EC + 0,7 * EP)/10)

Examen final

Los estudiantes regulares rendirán un examen equivalente a la actividad Examen de Promoción (EP), la cual será calificada del mismo modo que para los alumnos promocionados, considerando para la variable EC el rendimiento alcanzado durante la cursada, y para la variable EP el rendimiento alcanzado en el examen final.

Los estudiantes libres rendirán un examen que constará de dos partes:

1. Una prueba de competencias con la misma metodología y objetivos que la Evaluación Conceptual (EC) que serán evaluados automáticamente en el aula virtual. La aprobación de esta primera parte es requisito excluyente para la prosecución del examen, y se deberá obtener un rendimiento igual o superior al 60%.
2. Un examen equivalente al Examen de Promoción (EP), con la misma modalidad y objetivos que el requerido a los alumnos regulares.

La Nota Final final de examen para los casos 1 y 2 se obtendrá por la siguiente expresión:

Nota Final = redondear ((0,3 * EC + 0,7 * EP)/10)

Actividades prácticas y de laboratorio

Las clases son de carácter teórico-práctico en laboratorio de computación, en las cuales se presentan los temas teóricos, se dara lugar al diálogo e intercambio de ideas y se resuelven actividades específicas a la temática abordada cada clase.

Resultados de aprendizaje

CE1.1: Diseñar, proyectar y calcular sistemas, equipos y dispositivos de generación, transmisión y/o procesamiento de campos y señales analógicos y digitales; circuitos integrados; hardware de cómputo de propósito general y/o específico y el software a él asociado; hardware y software de sistemas embebidos y dispositivos lógicos programables; sistemas de automatización y control; sistemas de procesamiento y de comunicación de datos y sistemas irradiantes, para brindar soluciones óptimas de acuerdo a las condiciones técnicas, legales, económicas, humanas y ambientales.

- Expresar los fundamentos de los sistemas operativos y su interacción con el usuario y con programas del usuario..
- Aplicar los fundamentos del paradigma orientado a objetos mediante un lenguaje de programación orientado a objetos utilizando un entorno de desarrollo.

CE1.2.1: Modelar matemáticamente problemas de ingeniería, hallar soluciones específicas empleando algoritmos matemáticos y herramientas informáticas, y generalizar las soluciones para resolver situaciones reales de ingeniería.

- Expresar el concepto de tipo de dato y tipo abstracto de dato y ser capaz de identificar las características principales de un sistema de tipos.
- Desarrollar aplicaciones informáticas utilizando diferentes estructuras de datos, aplicando algoritmos de ordenación y búsqueda sobre ellas.
- Conocer lenguajes de programación actuales.

CE1.2.2: Manejar herramientas informáticas que permitan el desarrollo de soluciones en el ámbito de la ingeniería.

- Seleccionar y utilizar herramientas tipo Entorno Integrado de Desarrollo de software (IDE del Inglés *Integrated Development Environment*).
- Conocer herramientas para desarrollo de software (generación de código, depuración, pruebas y ensayos, etc.).

CE1.4.3: Analizar, diseñar, programar, implementar, probar, depurar y evaluar hardware y software para sistemas de computación de propósitos específicos.
CE1.4.4: Analizar, diseñar, programar, implementar, probar, depurar y evaluar hardware y software para sistemas de computación de propósitos generales.
CE1.4.6: Analizar, diseñar, programar, implementar, probar, depurar y evaluar sistemas de procesamiento de datos (hardware/software).

- Interpretar el diseño modular y los conceptos de cohesión y acoplamiento.
- Explicar los fundamentos de la orientación a objetos y ser capaz de identificar las diferencias entre la representación basada en objetos y los modelos de flujo de datos.
- Crear soluciones algorítmicas a problemas y ser capaz de representarlas como un software orientado a objetos.
- Dominio de conceptos y fundamentos sobre programación de aplicaciones en sistemas operativos.

CE3.1: Validar y certificar el funcionamiento, condición de uso o estado de los sistemas mencionados anteriormente.

- Evaluar mediante prueba y validación los requerimientos especificados para un software.
- Conocer conceptos de validación y certificación de software.

Bibliografía

- David J. Barnes y Michael Kölling (2017), "*Programación orientada a objetos con Java usando BlueJ*" (Sexta edición), Pearson https://efn.biblio.unc.edu.ar/cgi-bin/koha/opac-detail.pl?biblionumber=17531
- William Stallings (2011), "*Sistemas operativos: aspectos internos y principios de diseño*" (Quinta edición), Pearson https://efn.biblio.unc.edu.ar/cgi-bin/koha/opac-detail.pl?biblionumber=9033