FACULTAD
DEPARTAMENTO TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES (TIC)

Materia:	Algoritmos y programación II
NRC:	11939
Código:	09704
Grupo:	005
Programa/Semestre:	SIS 03, TEL 03
Periodo Académico:	202310
Intensidad Semanal:	5
Créditos:	3

PROFESORES

Domiciano Rincón

DESCRIPCIÓN

En este curso se busca que cada estudiante tenga los conocimientos en el área de programación que le permitan implementar soluciones algorítmicas para resolver problemas de nivel intermedio en complejidad. Siendo éste el tercer curso del bloque de formación en algoritmos, se introducen nuevos conceptos para modelar las entidades de la solución y técnicas para implementar algoritmos que resuelvan problemas de una complejidad mayor a la que el estudiante ha tenido hasta el momento. Asimismo, se estudian conceptos fundamentales del diseño de la experiencia de usuario. Este curso pertenece a un bloque de conocimiento incremental, por lo que se supone que el estudiante cuenta conlas competencias adquiridas en el curso previo de Algoritmos y Programación I.

OBJETIVOS

GENERALES

Resolver problemas combinando las ventajas de utilizar diferentes estrategias, enfoques y buenas prácticas de programación (orientada a objetos, concurrencia, recursividad y codificación segura) como parte de un proceso de desarrollo de la solución, consciente de la importancia de la gestión de recursos computacionales y del diseño de la experiencia de usuario, con una mayor capacidad de respuesta técnica a las necesidades concretas del contexto del problema.

RESULTADOS DE APRENDIZAJE RELACIONADOS CON EL PROGRAMA DE SISTEMAS:

•SO-1. Solución de problemas (T)
•SO-2. Diseño de ingeniería (T)
•SO-3. Comunicación efectiva (T)
•SO-4. Ética e impactos (T)
•SO-7. Aprender a aprender (T)

RESULTADOS DE APRENDIZAJE RELACIONADOS CON EL PROGRAMA DE TELEMATICA:

•SO-2. Diseño de ingeniería (T)
•SO-3. Comunicación efectiva (T)
•SO-4. Ética e impactos (T)

•SO-7. Aprender a aprender (T)

TERMINALES

Al finalizar el semestre el estudiante estará en capacidad de:

OT1. Verificar y controlar errores en los programas mediante el uso de pruebas unitarias y excepciones.
OT2. Evaluar algoritmos de búsqueda y ordenamiento clásicos en estructuras de datos lineales y no lineales.
OT3. Implementar soluciones a problemas que requieran el uso de listas enlazadas, árboles y recursión.
OT4. Desarrollar y desplegar programas que utilicen interfaces gráficas de usuario con persistencia de la información, incorporando buenas prácticas, conceptos de codificación segura y elementos básicos de gestión de la configuración.
OT5. Construir programas bajo el paradigma de programación orientado a objetos concurrente, gestionando el acceso sincronizado a recursos compartidos en un nivel básico.

ESPECÍFICOS

Unidad 1: Estructuras y Algoritmos Recursivos
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.
2. Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.
3. Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.
4. Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

Unidad 2: Estructuras Lineales Enlazadas
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Utilizar estructuras enlazadas de objetos para modelar grupos de atributos no primitivos de tamaño flexible.
2. Escribir los algoritmos necesarios para manipular estructuras lineales que almacenan sus elementos enlazándolos entre ellos.
3. Utilizar herramientas de diseño para la construcción de diagramas y la generación de código a partir de éstos.

Unidad 3: Pruebas Automáticas y Tipos de Excepción
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Reconocer el mecanismo de manejo de excepciones señalando las implicaciones de la propagación versus el control.
2. Usar e implementar distintos tipos de excepción como parte de un programa, de manera que sea posible clasificar los tipos de error que se pueden presentar y asociarles en el programa distintas manerasde recuperarse ante el problema.
3. Diseñar pruebas unitarias automáticas que permitan validar el adecuado funcionamiento de las operaciones del sistema desarrolladas para soportar los requerimientos funcionales.
4. Desarrollar las clases y los métodos necesarios para implementar las pruebas unitarias automáticas, que ayudan a comprobar el correcto funcionamiento de un programa.

Unidad 4: Persistencia, Manejo de Archivos de Texto y Algoritmos de Ordenamiento y Búsqueda
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Manipular archivos de texto para implementar requerimientos del cliente relacionados con persistencia.
2. Leer entradas e imprimir salidas de programas que interactúan directamente con archivos de texto y no con usuarios finales.
3. Hacer persistir el estado del modelo de solución del problema durante la ejecución de un programa y restaurarlo cuando se requiera usando la técnica de serialización.
4. Implementar algoritmos clásicos de ordenamiento de datos en estructuras de datos lineales y aplicarlos en la solución de un problema.
5. Implementar algoritmos clásicos de búsqueda de información en estructuras de datos lineales y

aplicarlos en la solución de un problema.
6. Hacer uso de las interfaces Comparable y Comparator para definir relaciones de orden total sobre objetos.
7. Calcular el tiempo de ejecución de un algoritmo por medio de las operaciones de tiempo del sistema
8. Implementar métodos que permitan generar muestras con datos aleatorios.

Unidad 5: Construcción de la interfaz gráfica
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Utilizar una arquitectura de tres capas para el desarrollo de un programa de computador, repartiendo de manera adecuada las responsabilidades entre la interfaz de usuario, el control de la interfaz y el modelo. El estudiante deberá poder explicar la importancia de mantener separadas las clases de estos tres dominios.
2. Construir las clases que implementan una interfaz de usuario.
3. Aplicar la técnica de descomposición de requerimientos para cumplir con la funcionalidad de un programa de computador.

Unidad 6: Concurrencia y Dibujo Básico en 2D
Al finalizar esta unidad, el estudiante estará en capacidad de:
1. Desarrollar un programa que maneje concurrencia, de manera que sea posible que ejecute más de una parte del programa de manera simultánea, utilizando hilos de ejecución (threads).
2. Construir interfaces de usuario que incluyan gráficas en 2 dimensiones como una alternativa en la presentación de información al usuario.

DE FORMACIÓN DE VALORES Y COMPETENCIAS

Durante el desarrollo de todo el curso, el estudiante desarrollará la capacidad de:
1. Utilizar convenciones de codificación como un elemento de calidad en la implementación de programas.
2. Reconocer la importancia de la medición de su proceso personal de desarrollo de software.
3. Medir sistemáticamente los tiempos utilizados en cada una de las etapas de desarrollo de software.
4. Medir sistemáticamente el tamaño de los entregables producidos.
5. Mantener un registro ordenado del proceso de medición llevado a cabo en cada uno de los proyectos de desarrollo de software.
6. Valorar y corregir las inconsistencias que se pueden presentar entre cada uno de los elementos de diseño propuestos en el diagrama de clases y su implementación en el lenguaje de programación.

METODOLOGÍA

Para los estudiantes:
De acuerdo a la metodología de aprendizaje activo de la universidad Icesi, los estudiantes deben preparar, antes de la clase, los temas asignados en la programación del curso. Esto es:

- Leer y analizar el material asignado para la sección de clase. Si no se ha asignado material, entonces investigar sobre los temas acordados en la planeación.
- Utilizar estrategias de estudio (mapas conceptuales, mapas mentales, resúmenes, etc.) que sean efectivas y que sirvan como refuerzo después del proceso de lectura.
- Contestar las preguntas que contiene el material, así como las preguntas adicionales

que el profesor entregue.

• Resolver los ejercicios propuestos en el material, así como los ejercicios adicionales que se le entreguen.

• Formular preguntas que requieran ser resueltas durante la clase.

Durante la clase, el estudiante deberá:

• Plantear las dudas que le quedaron durante el proceso de estudio del tema a tratar.

• Participar en las actividades de revisión y consolidación de conceptos que proponga el profesor.

• Trabajar en la solución de los problemas de aplicación que se propongan.

Después de la clase:

• Establecer las relaciones entre los temas tratados en la clase y el conocimiento previamente adquirido.

• Resolver los ejercicios de aplicación del tema, que tienen un nivel de complejidad mayor al de los ejercicios que resolvió previamente.

Para el desarrollo del curso:

Este curso cuenta semanalmente con dos tipos espacios que son utilizados de la siguiente forma:

• Sesiones teórico/prácticas: los estudiantes y el profesor se encontrarán en este espacio, en dos sesiones de una hora y media (una hora, 30 minutos) en las cuales se llevará a cabo la discusión de los diferentes temas y la realización de ejercicios que permitan ponerlos en práctica.

• Sesiones principalmente prácticas: los estudiantes, el profesor y el monitor del curso comparten un espacio en el cual se llevan a cabo la solución de problemas propuestos y su implementación en el lenguaje Java. Para este componente se han destinado dos (2) horas por semana.

Este curso utilizará Java como lenguaje de programación y Eclipse como entorno de desarrollo. No obstante, los objetivos de aprendizaje son independientes del lenguaje y el entorno de programación seleccionado.

ESQUEMA DE EVALUACIÓN:

En todas las evaluaciones de carácter escrito se tendrá en cuenta la GRAMÁTICA, ORTOGRAFÍA y PUNTUACIÓN con el objetivo de desarrollar y consolidar la competencia de escritura del estudiante. Las evaluaciones del curso están programadas en el cronograma del curso disponible en Moodle.

Tareas Integradoras: durante el semestre se llevarán a cabo 3 Tareas Integradoras que los estudiantes deben entregar utilizando la plataforma de e-learning (Moodle). La nota de cada tarea integradora se calcula así: primero, se revisa el trabajo entregado con base en la rúbrica de la tarea. Esa revisión deja una nota que será multiplicada por el factor de la sustentación. El factor, es un valor real entre 0 y 1 quese calcula así: se hacen, por ejemplo, 4 preguntas al estudiante, si responde a las 4 preguntas de forma correcta, entonces el valor será 1, si responde a 3 correctamente (1 la responde erróneamente) obtendrá 0.75 en el factor, si responde solo a 2 correctamente tendrá 0.5, y así. Entonces, si en la calificación obtuvo una calificación de 5.0 y en el factor de sustentación es de 0.75, su nota de la tarea será de 5.0*0.75 = 3.75. Las tareas integradoras tendrán una evaluación formativa, es decir, si después deobtener esta nota, el estudiante desea mejorarla, puede hacerlo, y el trabajo será evaluado nuevamente, por una única vez adicional. Esto último, por cada tarea integradora.

Seguimientos de Aprendizaje: al final de la semana se evaluarán los temas y competencias desarrollados a lo largo de la semana a través de una prueba breve que puede ser teórica, práctica o teórico/práctica. La prueba será calificada una vez cada dos semanas.

Nota definitiva

A continuación se especifican los porcentajes de las evaluaciones:

Tarea Integradora 1: 25%
Tarea Integradora 2: 25%
Tarea Integradora 3: 25%
Seguimientos: 25%

ESQUEMA DEL CURSO

Descripción	Comentarios	Peso
Seguimientos	N/A	25%
Tarea integradora 1	N/A	25%
Tarea integradora 2	N/A	25%
Tarea integradora 3	N/A	25%

BIBLIOGRAFÍA

TEXTOS GUÍAS

LIBROS

Deitel, Paul. Harvey Deitel (2016) . Como Programar en Java, Edición Décima edición, Pearson

Villalobos, Jorge A (2008) . Introducción a las estructuras de datos: Aprendizaje activo basado en casos, Edición Primera edición, Prentice-Hall

Weiss, Mark Allen (2010) . Estructuras de datos en Java, Edición Cuarta edición, Pearson

RECURSOS

Nombre	Descripción	Necesario para la clase
Eclipse	IDE para programar en Java	Si
Scene Builder (Gluon)	Para construir interfaces gráficas con JavaFX	Si
JDK 1.8 (Oracle)	El JDK de oracle trae consigo JavaFX	Si
Git	Herramienta de control de versiones	Si