NOMBRE DE LA INSTITUCIÓN	Universidad de Sonora
UNIDAD ACADÉMICA	Unidad Regional Centro
DIVISIÓN ACADÉMICA	División Ciencias Exactas y Naturales
DEPARTAMENTO ACADÉMICO QUE IMPARTE SERVICIO	Departamento de Matemáticas
LICENCIATURAS USUARIAS	Licenciatura en Ciencias de la Computación
NOMBRE DE LA MATERIA	**Programación avanzada**
CLAVE	**9448**
EJE FORMATIVO	Básico
REQUISITOS	Programación de computadoras
CARÁCTER	Obligatorio
VALOR EN CRÉDITOS	8 (3 teoría/2 laboratorio)
CLAVE	

Introducción:
El rápido desarrollo de las tecnologías de la información tiene un impacto profundo en la sociedad actual. De aquí la importancia que tiene la computación y en particular el análisis, diseño y construcción de software bien estructurado y eficiente, así como las estructuras de datos utilizadas y la forma de pensar para conseguir soluciones a problemas de manera clara, eficaz y fácil de implementar en una computadora.

Este es un curso introductorio al paradigma orientado a objetos, que ofrece los fundamentos de proceso y la metodología de la programación utilizando el lenguaje java. Los principios abordados en este contenido permitirán desarrollar las habilidades de abstracción necesarias para crear modelos de clases en el dominio del problema y en el de solución, con los cuales podrá construir software con componentes.

Objetivo General
El alumno será capaz de comprender y aplicar las técnicas básicas para el desarrollo de soluciones a problemas elementales de procesamiento de información utilizando el paradigma orientado a objetos en el proceso de elaboración del software.

Objetivos Específicos
El alumno será capaz de:

- Reconocer y aplicar los principios básicos que sustentan el desarrollo de software con base en el paradigma orientado a objetos.
- Aplicar el proceso de desarrollo de software para la elaboración de los programas que resuelvan los problemas que se planteen en el curso.
- Enunciar soluciones con clases y objetos.
- Estudiar y ejercitar la semántica y la sintaxis del lenguaje de programación Java.
- Destacar elementos principales de la programación en Java.

Contenido

1. **Estructuras de datos básicas.**
   1.1. Definición de estructuras de datos básicas
   1.2. Uso de estructuras de datos básicas.
   1.3. Implantaciones contiguas y ligadas.

2. **Tipos de datos abstractos.**
   2.1. Propósito de ADT.
   2.2. Implantación de un ADT en un lenguaje de alto nivel.

3. Algoritmos recursivos.

3.1. Introducción a los algoritmos recursivos.
3.2. Relación con inducción matemática.
3.3. Comparación de algoritmos recursivos e interactivos.

4. Análisis de complejidad

4.1. Análisis asintótico (O, ι) del límite mayor y promedio de algoritmos recursivos e iterativos
4.2. Tiempo vs espacio en algoritmos

5. Ordenación y búsqueda.

5.1. Algritmos de ordenación O (n^2) (eq. inserción y selección).
5.2. Algoritmos de ordenación O(n long n) (eq. quickosrt. heapsort, insergesort).
5.3. Otros algoritmos de ordenación (eq. shell, bracket, radix).
5.4. Comparación de algoritmos.
5.5. Búsqueda serial, búsqueda binaria y búsqueda en árboles binarios.
5.6. Resolución de colisiones.

6. Estrategias en solución de problemas.

6.1. Algoritmos greedy.
6.2. Algoritmos divide y vencerás.
6.3. Algoritmos backtracking

7. Representación de números, errores y portabilidad.

7.1. Precisión finita en la representación de números reales y enteros.
7.2. Errores en la aritmética de las computadoras y relación con la portabilidad.
7.3. Problemas de condición Well y de condición III.

8. Secuencias de control.

8.1. Expresiones, orden de evaluación y efectos laterales.
8.2. Declaraciones simples y compuestas.
8.3. Subprogramas y corrutinas como abstracciones y declaraciones.
8.4. Manejo de excepciones.

9. Conceptos fundamentales de solución de problemas.

9.1. Abstracción procedural
9.2. Estructuras de control (selección, iteración, recursión).
9.3. Tipos de datos.
9.4. El proceso de diseño de software, desde la especificación hasta la implantación, el paso de refinamiento y representación gráfica.
9.5. Métodos de aproximación iterativa.
9.6. Aproximación iterativa en problemas matemáticas (Newton).
9.7. Clasificación de errores.
9.8. Revisión de las aplicaciones en ciencias e ingeniería.

Estrategias Didácticas

- Promover la participación activa de los estudiantes en todas las sesiones del curso fomentando la discusión de los conceptos elementales que sustentan a la programación avanzada y motivando el interés en la investigación en los distintos paradigmas de programación.
- Promover la consulta bibliográfica y en línea de los distintos temas que se abordarán en el curso, así como la presentación de informes y exposiciones orales.

- Para cada una de las unidades, plantear problemas típicos del tema en exposición y desarrollar la solución de éste a través de un proceso de desarrollo de software sistemático.
- Asegurarse del dominio de los temas, a través de la práctica en el desarrollo de algoritmos que solucionen diversos problemas planteados por el profesor en las sesiones de clase y, en particular, en tareas extraclase.
- Diseñar prácticas de laboratorio apropiadas para cada unidad con la finalidad de que el alumno demuestre el dominio de los conceptos abordados en clase.
- Comparar las distintas soluciones que puede tener un problema, de acuerdo al trabajo realizado por el grupo.
- Motivar discusiones sobre eficiencia de algoritmos.

Estrategias de Evaluación

Para la evaluación de los estudiantes, se sugiere que el profesor tome en cuenta:

- Resultados de los exámenes parciales aplicados (se sugiere que sean al menos tres).
- Tareas, trabajos de investigación, presentaciones.
- Participación individual y colectiva en las actividades del curso.
- Elaboración de prácticas de laboratorio.
- Elaboración de un proyecto final.

Los porcentajes de cada indicador serán establecidos al inicio del semestre.

Bibliografía:

- Sintes, Anthony. Aprendiendo programación orientada a objetos en 21 lecciones avanzadas. Pearson Educación, México, 2002.
- Jacobson, I., Booch, G. y Rumbaugh, J. El proceso unificado de desarrollo de software. Pearson Educación, Madrid, 2000.
- Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall, México, 1999.
- Fowler, Martin. UML gota a gota. Addison Wesley Longman de México, México, 1999.
- Sintes, Anthony. Aprendiendo programación orientada a objetos en 21 lecciones avanzadas. Pearson Educación, México, 2002.
- Deitel, H. M. y Deitel, P. J. *C++ Cómo programar*. 2da. ed., Prentice Hall. México, 1999.
- Joyanes A., L. *Programación en C++. Algoritmos, estructuras de datos y objetos*. McGraw Hill.

Perfil académico deseable del profesor

Se recomienda que el profesor tenga el siguiente perfil:

- Formación profesional en el área de ciencias de la computación o área afín.
- Conocimiento y aplicación de los principios de desarrollo de software, en particular los paradigmas de diseño estructurado descendente y el orientado a objetos.