UNIVERSIDAD AUTÓNOMA DE MANIZALES UAM VIGILADA MINEDUCACIÓN	**SYLLABUS DE LA ASIGNATURA**

1. IDENTIFICACIÓN

Nombre asignatura	Programación Orientada a Objetos
Código	103018
Departamento (área)	Ciencias Computacionales
Programa (s) en los que se ofrece	Ingeniería de Sistemas
Modalidad de la asignatura	Presencial
Número de créditos	3
Horas de trabajo presencial/sincrónico	64
Horas de trabajo dirigido/tutoría sincrónica	32
Horas de trabajo independiente/asincrónico	48
Prerrequisitos	1. Fundamentos de Programación Orientada a Objetos 2. Técnicas de Programación
Año- Periodo académico	2024-3
Docente (s)	Jorge Iván Meza Martínez
E-mail docente (s)	jimezam@autonoma.edu.co

2. JUSTIFICACIÓN

La creación de aplicaciones computacionales no es solo la programación en un lenguaje de alto nivel específico, es la intersección de diferentes ejes temáticos como: la solución de problemas, la algoritmia, la Ingeniería de Software, los lenguajes de programación y las herramientas computacionales asociadas. Cualquier proceso de desarrollo que se realice implica una serie de actividades de cada uno de estos ejes temáticos, la correcta realización de dichas actividades garantiza la calidad del producto creado.

En el curso anterior se definieron elementos fundamentales de modelamiento en el paradigma Orientado a Objetos, es importante ahora profundizar con estos elementos, adicionando conceptos avanzados como el uso de clases abstractas, interfaces, polimorfismo además de mejorar la implementación de aplicaciones en un lenguaje como Java y en una plataforma de desarrollo potente como NetBeans. Los programas desarrollados deben presentar características deseables en una aplicación como: modularidad, arquitecturas por niveles, manejo de excepciones, hilos e interfaces gráficas de usuario.

3. DESEMPEÑO ESPERADO

RESULTADOS DE APRENDIZAJE

- Diseña soluciones computacionales a problemas de la vida real a través del paradigma de programación orientado a objetos y UML.
- Implementa soluciones computacionales orientadas a objetos mediante el uso de lenguajes de programación del mismo paradigma como Java.
- Describe diferentes patrones de diseño multicapa basados en el paradigma de programación orientado a objeto, sus elementos, aplicaciones comunes y ventajas para seleccionar la más adecuada de acuerdo al problema a solucionar.
- Emplea la documentación a nivel de código y el lanzamiento y captura de excepciones para mejorar la calidad del software.
- Implementa soluciones de persistencia a partir de archivos en formatos XML y JSON para almacenar información que podrá ser usada en diferentes ejecuciones del software.

4. CONTENIDOS

- Control de versiones de código (Git)

    - Motivación
    - Historia
    - Uso local
        - Creación del repositorio
        - Consulta del estado
        - Listado de los *commits*
        - Subida al escenario (*stage*)
        - Creación de un *commit*
    - Uso remoto
        - Creación de un repositorio remoto
        - Gestión local de los repositorios remotos
        - Publicación de contenidos locales (`push`)
        - Actualización de contenidos remotos (`pull`)
        - Solución de conflictos

- Implementación de interfaces gráficas de usuario (`Swing`)

- Conceptos básicos
- Diseño de interfaces gráficas
- Creación de la ventana principal (`JFrame`)
- Manejo de eventos
- Manejo de diálogos predefindos (`JOptionPane`)
- Manejo de diálogos (`JDialog`)

- Manejo de excepciones (`Exception`)

  - Conceptos básicos
  - Tipos de excepciones
    - `Error`
    - `Exception`
    - `Runtime`
  - Lectura de una traza
  - Manejo de excepciones
    - `try`
    - `catch`
    - `finally`
    - `throws`
  - Creación de excepciones propias
    - `throw`

- Clases abstractas (`abstract`)

  - Motivación
  - Características
  - Implementación

- Uso de interfaces (`interface`)

  - Motivación
  - Características
  - Implementación

- Polimorfismo

  - Conceptos básicos
  - Objetivo principal
  - Formas de implementación

- Manejo gráfico en las interfaces gráficas de usuario (`Graphics`)

    - Conceptos básicos
    - Como dibujar sobre un componente
        - Sistema de coordenadas del componente
        - Método `paint`
        - Objeto `Graphics`
    - Primitivas de dibujo y color
    - Redibujo de la interfaz gráfica (`repaint`)

- Propuesta de modelo de clases orientado al taller final

    - Motivación
    - Elementos mínimos para el desarrollo de un juego
    - Modelo de clases propuesto
    - Implementación prpopuesta

- Manejo de Hilos (`Thread`)

    - Concepto de concurrencia
    - Formas de implementación
        - Heredando de `Thread`
        - Implementando `Runnable`
    - Primitivas principales
        - Iniciar
        - Detener
        - Pausar

- Persistencia de datos con archivos

    - Conceptos básicos
    - Tipos de archivos
        - Texto plano
        - CSV
        - XML/JSON
        - Archivos *serializados*
    - Primitivas principales
        - Abrir/cerrar
        - Leer
        - Escribir

- Pruebas unitarias (*JUnit*)

    - Motivación
    - Conceptos principales
    - Diseño de las pruebas
    - Creación de casos de prueba
    - Ejecución de los casos de prueba
    - Verificación de la cobertura

- Tipos de datos genéricos (*generics*)

    - Motivación
    - Uso de clases genéricas
    - Implementación de clases genéricas propias

- Colecciones (`Collection`)

    - Conceptos básicos
    - Primitivas principales
    - Uso de colecciones
    - Implementación de colecciones propias

- Expresiones Lambda

    - Conceptos básicos
    - Motivación
    - Implementación
    - Uso de expresiones lambda

5. ESTRATEGIAS DE ENSEÑANZA

- El curso consiste semanalmente en 4 horas de clase presencial con el profesor, 2 horas de trabajo supervisado y al menos 3 horas de trabajo individual.
- La asistencia a clase y la realización de sus actividades, tanto evaluativas como de práctica, son obligatorias.
- Antes de cada clase los estudiantes deben realizar las lecturas y actividades especificadas en el aula del curso, se harán actividades evaluativas de dichas lecturas.
- Las sesiones presenciales del curso se concentrarán en la solución de dudas de las lecturas asignadas, la presentación magistral de los temas complementarios por parte del

docente, el seguimiento de ejemplos y la realización de ejercicios prácticos que ayuden al estudiante a formalizar los temas tratados en cada módulo.

- La evaluación de los temas vistos se realizará a partir de los talleres y exámenes dispuestos en cada uno de los tres cortes del curso. Cada uno de ellos se encuentra planeado para experimentar los temas teóricos y prácticos que se espera que el estudiante domine al finalizar el curso.
- Generalmente este curso cuenta con el apoyo de un estudiante monitor para reforzar los conceptos teóricos y prácticos de los estudiantes del curso. La asistencia a las monitorias no es obligatoria, pero totalmente necesaria.

6. ESTRATEGIA DE EVALUACIÓN DEL APRENDIZAJE

Corte	Tipo de evaluación	% parcial	% total
1	Talleres	50%	30%
	Parcial	50%	
2	Talleres	50%	35%
	Parcial	50%	
3	Talleres	40%	35%
	Taller final	60%	

7. BIBLIOGRAFÍA

Requerida

[1] RUMBAUGH, James; JACOBSON, Ivar y BOOCH, Grady. UML: El Lenguaje Unificado de Modelado. Addison-Wesley, 1996. [005.1 B662]
[2] NAIR, Premchand. Java Programming Fundamentals: Problem Solving Through Object Oriented Analysis and Design. CRC Press, 2008. [005.133 N147]
[3] KRISHNA, Radha. Object oriented programming through Java. CRC Press, 2007. [005.133 K747]

Sugerida

[4] BARNES, David y KOLLING, Michael. Programación orientada a objetos con Java: una introducción práctica usando BlueJ, 3 ed. Pearson Educación, 2007 [005.133 B175]
[5] DEAN, John y DEAN Raymond. Introducción a la Programación con Java. McGraw-Hill, 2009 [005.133 D315]
[6] DEITEL, H.M. Cómo programar en Java, 7 ed. Pearson Educación, 2008. [005.13 D347]
[7] ECKEL, Bruce. Thinking in Java, 2 ed. Prentice Hall, 2000. [005.133 E243]
[8] ERIKSSON, Hans-Erik; PENKER, Magnus; LYONS, Brian y FADO, David. UML 2 Toolkit. Wiley, 2004. [005.12 E744]