VICERRECTORADO ACADÉMICO GENERAL

PROGRAMA DE ASIGNATURA – SÍLABO PRESENCIAL

1. DATOS INFORMATIVOS

MODALIDAD: Presencial	**DEPARTAMENTO:** Eléctrica y Electrónica		**AREA DE CONOCIMIENTO:** Programación	
CARRERA: Electrónica	**NOMBRE ASIGNATURA:** Programación II		**PERÍODO ACADÉMICO:**	
PRE-REQUISITOS: Fundamentos de Programación II	**CÓDIGO:** 15083	**NRC:**	**No. CRÉDITOS:** 6	**NIVEL:** 2
CO-REQUISITOS:	**FECHA ELABORACIÓ**	**SESIONES/SEMANA:**		**EJE DE FORMACIÓN:** Básico
		TEÓRICAS: 2	**LABORATORIOS:** 4	
DOCENTE:				

DESCRIPCIÓN DE LA ASIGNATURA:

Programación II es una asignatura básica que estudia el paradigma de programación orientado a objetos, para lo cual se estudia los fundamentos teóricos de este nuevo enfoque junto con la manejo del entorno gráfico para la creación de aplicaciones.

Esta asignatura pretende crear las competencias necesarias del futuro profesional para que resuelva problemas en el ámbito de la programación básica y computación, aplicando el enfoque de programación orientada a objetos haciendo uso del lenguaje de programación Java.

CONTRIBUCIÓN DE LA ASIGNATURA A LA FORMACIÓN PROFESIONAL:

Esta asignatura corresponde a la primera etapa formación profesional, proporciona al futuro profesional las bases para la resolución de problemas mediante el uso de un computador con un lenguaje orientado a objetos.

RESULTADO DE APRENDIZAJE DE LA CARRERA: (UNIDAD DE COMPETENCIA)

Analiza problemas, desarrolla la lógica de programación e implementa el software específico para la solución del mismo, así como el análisis y desarrollo de las redes básicas de computadoras y sus servicios y aplicaciones.

OBJETIVO DE LA ASIGNATURA:

Estudiar el paradigma de programación orientado a objetos, basándose en los fundamentos teóricos de este nuevo enfoque y las características principales de la orientación a objetos como son la creación de nuevos tipos de datos, herencia, polimorfismo, aplicaciones gráficas y manejo de concurrencia para la resolución de problemas en el ámbito de la ingeniería utilizando un lenguaje de programación open source.

RESULTADO DE APRENDIZAJE DE LA ASIGNATURA: (ELEMENTO DE COMPETENCIA)

Desarrolla programas aplicando fundamentos, conceptos y características de paradigmas orientado a objetos utilizando herramientas computacionales a fin de resolver problemas prácticos de la ingeniería.

1. Aplicar correctamente la lógica de programación utilizando el enfoque orientado a objetos, en la resolución de problemas.
2. Reconocer la estructura de objeto, componentes y características que permita dar inicialización a nuevos tipos de datos conocidos como objetos.
3. Desarrolla aplicaciones empleando mecanismos de herencia y polimorfismo como características fundamentales de la programación orientada a objetos.
4. Desarrolla aplicaciones utilizando herramientas Open Source para la resolución de casos prácticos utilizando interfaces gráficas.
5. Reconoce la estructura de objetos componentes y características para efectuar multiprocesos.
6. Aplica conocimientos generales para elaborar un proyecto final en el que involucre todo el paradigma orientada a objetos, dando énfasis en el conocimiento de archivos y encadenamiento en Java.

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

No.	UNIDADES DE CONTENIDOS	RESULTADOS DEL APRENDIZAJE Y SISTEMA DE TAREAS
1	**UNIDAD 1: FUNDAMENTOS DE PROGRAMACIÓN Y PRINCIPIOS DE DISEÑO ORIENTADO A OBJETOS.**	**Resultados de Aprendizaje de la Unidad 1:** **1.** Aplica correctamente la lógica de programación utilizando el enfoque orientado a objetos, en la resolución de problemas. **2.** Reconocer la estructura de objeto, componentes y características que permita dar inicialización a nuevos tipos de datos conocidos como objetos.
	Contenidos: 1.1. Conceptos Generales 1.1.1. Principios fundamentales de la programación orientado a objetos 1.1.2. Definición de clase, objeto, atributos y métodos 1.2. Plataforma de Desarrollo 1.2.1. Definición de java 1.2.2. Características 1.2.3. Entorno de aplicación: JDK y JRE 1.2.4. Instalación, configuración y uso básico del IDE de desarrollo. 1.3. Tipos de datos, variables, constantes y conversión de tipo. 1.4. Estructura general de un programa 1.4.1. Uso de IDE 1.5. Revisión de Sentencias de control 1.5.1. Secuenciales 1.5.2. Selección 1.5.3. Repetición 1.6. Clases y objetos 1.6.1. Implementación de clases 1.6.2. Declaración e instanciación de objetos 1.6.3. Encapsulamiento: Atributos, métodos y niveles de visibilidad 1.7. Paso de parámetros a Métodos 1.7.1. Paso de parámetros en funciones 1.7.2. Paso de parámetros entre métodos 1.7.3. Paso de parámetros entre clases de diferentes paquetes 1.7.4. Generación e incorporación de librerías jar. 1.8. Constructores 1.8.1. Definición 1.8.2. Tipos de constructores 1.8.3. Instanciación 1.9. Métodos getters, setters (get / set). 1.9.1. Definición y Aplicación. 1.10. Excepciones 1.10.1. Excepciones y errors 1.10.2. Clases de excepción 1.10.3. Tipos de excepciones: marcadas y no marcadas. 1.10.4. Captura de excepciones: bloques try… catch … finally. Propagación de excepciones. 1.10.5. Lanzamiento de excepciones 1.10.6. Métodos para el control de excepciones 1.10.7. Clases de excepciones personalizadas 1.10.8. Aserciones : Formato. Habilitar aserciones: Compilar y Ejecutar. 1.11. Clases de entrada / salida por consola en lenguaje orientado a objetos.	Tarea 1: Identificación y representación de clases sus atributos y métodos. Tarea 2: Implementaciones de clases en Java. Resolución de Problemas con Constructores. Tarea 3: Resolución de Problemas aplicando entrada y salida básicas. Tarea 4: Aplicación del puntero This y de modelamiento con modelos de clases y casos de uso.

	1.12. Manejo de clases generales: String, Math, Time, Random, Calendar. 1.13. Lenguaje Unificado de Modelado (UML) 1.13.1. Uso básico de modelos de: clases y casos de uso.	
2	**UNIDAD 2: MÉTODOS, HERENCIA Y POLIMORFISMO**	**Resultados de Aprendizaje de la Unidad 2:** **3.** Desarrolla aplicaciones empleando mecanismos de herencia y polimorfismo como características fundamentales de la programación orientada a objetos. **4.** Desarrolla aplicaciones utilizando herramientas Open Source para la resolución de casos prácticos utilizando interfaces gráficas.
	Contenidos: 2.1. Arreglos 2.2. Agregación / Composición 2.2.1. Colecciones: Arreglos de objetos, hashtable. 2.3. Herencia 2.3.1. Definición, ventajas, nomenclatura y reglas. Clase Object 2.3.2. Creación de herencia en Java. 2.3.3. Ejecución de constructores. 2.3.4. Métodos y atributos protegidos. 2.3.5. Clases finales. 2.3.6. Métodos: Clone, equals, toString, getClass. 2.4. Clases y métodos abstractos 2.4.1. Atributos y métodos estáticos. 2.4.2. Clases y métodos virtuales. 2.5. Polimorfismo 2.5.1. Definición y ventajas. 2.5.2. Asignación de objetos a variables de su superclase. 2.6. Interfaces 2.6.1. Definición e implementación de interfaz. 2.6.2. Interfaces y polimorfismo. 2.7. Encapsulamiento 2.7.1. Definición 2.7.2. Clases internas 2.7.3. Paquetes • Declaración • Paquetes incorporados • Acceso a los elementos de un paquete • Importación de paquetes • Control de Acceso a paquetes 2.8. Inferfaces Gráficas de usuario (awt , applet, y Swing) 2.8.1. AWT • Principales clases de AWT • Contenedores • Creación y personalización de ventanas 2.8.2. Modelo de gestión de eventos en Java • Interfaces de escucha y escuchadores • Proceso de gestión de eventos. • Clases de evento • Adaptadores • Gestores de organización AWT. 2.8.3. Swing • Creación de una interfaz grafica swing. • Principales clases de swing. jTextComponent, JLabel, JButton, Canvas, Choice,Scrollbar.	Tarea 1: Resolución de Problemas sobre herencia: - Clases abstractas vs interfaces. Tarea 2: Resolución de Problemas sobre herencia: - Polimorfismo Tarea 3: Resolución de problemas utilizando AWT y applets Tarea 4: Resolución de problemas utilizando interfaces gráficas de usuario (Swing)

	• Listas y tablas swing: jList, jCombo Box, jTable. 2.8.4. Applets	
3	**UNIDAD 3: TÉCNICAS AVANZADAS ORIENTACIÓN A OBJETOS.**	**Resultados de Aprendizaje de la Unidad 3:** **5.** Reconoce la estructura de objetos componentes y características para efectuar multiprocesos. **6.** Aplica conocimientos generales para elaborar un proyecto final en el que involucre todo el paradigma orientada a objetos, dando énfasis en el conocimiento de archivos y encadenamiento en Java.
	Contenidos: 3.1. Aplicaciones Multitarea en Java 3.1.1. Creación de hilos • Derivación de clases clase Thread. • Implementación de interfaces: interfaz Runnable. • Sobreescritura del método run(). 3.1.2. Ciclo de vida de hilos 3.1.2.1. Métodos para control de threads. • Ejecución de nuevos hilos • Ejecución pausada de hilos • Finalización 3.1.3. Sincronización 3.1.4. Prioridades 3.1.5. Grupos de hilos 3.2. Modelos básicos de persistencia (Archivos planos) 3.2.1. Información sobre archivos y directorios. La clase File. 3.2.2. Lectura y escritura de archivos de texto: FileReader-BufferedReader, FileWriter-PrintWriter. 3.2.3. Lectura y escritura de datos primitivos en archivos: FileOutputStream-DataOutputStream, FileInput Stream-DataInputStream. 3.2.4. Lectura y escritura de objetos. 3.2.4.1. Serialización. 3.2.4.2. Creación de un objeto ObjectOutputStream 3.2.4.3. Creación de un objeto ObjectInputStream. 3.2.4.4. Deserialización de objetos.	Tarea 1: Aplicaciones multitarea con el manejo de hilos: - runnable Tarea 2: Aplicaciones multitarea con el manejo de hilos: - threads Tarea 3: Lectura y escritura de objetos como una forma básica de un modelo de persistencia. Tarea 4: Aplicaciones donde se evidencie el manejo de archivos.

3. PROYECCIÓN METODOLÓGICA Y ORGANIZATIVA PARA EL DESARROLLO DE LA ASIGNATURA

Se emplearán variados métodos de enseñanza para generar un aprendizaje de constante actividad, para lo que se propone la siguiente estructura:

- Se diagnosticará conocimientos y habilidades adquiridas al iniciar el periodo académico.
- Con la ayuda del diagnóstico se indagará lo que conoce el estudiante, como lo relaciona, que puede hacer con la ayuda de otros, qué puede hacer solo, qué ha logrado y qué le falta para alcanzar su aprendizaje significativo.
- A través de preguntas y participación de los estudiantes el docente recuerda los requisitos de aprendizaje previos que permite al docente conocer cuál es la línea de base a partir del cual incorporará nuevos elementos de competencia, en caso de encontrar deficiencias enviará tareas para atender los problemas individuales.
- Plantear interrogantes a los estudiantes para que den sus criterios y puedan asimilar la situación problemática.
- Se iniciará con explicaciones orientadoras del contenido de estudio, donde el docente plantea los aspectos más significativos, los conceptos, leyes y principios y métodos esenciales; y propone la secuencia de trabajo en cada