PROGRAMA DE ASIGNATURA - SÍLABO

1. DATOS GENERALES

Modalidad: PRESENCIAL ESPE MATRIZ SANGOLQUI	**Departamento:** CIENCIAS DE LA COMPUTACION		**Área de Conocimiento:** PROGRAMACION	
Nombre Asignatura: PROG. ORIENTADA A OBJETOS	**Período Académico:** PREGRADO S-II OCT 24 - MAR 25			
Fecha Elaboración: 30/11/20 22:03	**Código:** A0J08	**NRC:** 1973		**Nivel:** PREGRADO
Docente: LASCANO JORGE EDISON jelascano@espe.edu.ec				

Unidad de Organización	BÁSICA
Campo de Formación:	FUNDAMENTOS TEÓRICA
Núcleos Básicos de	Fundamentos de computación, Ingeniería y gestión de software Infraestructura, seguridad y gestión tecnológica e Investigación Desarrollo profesional.

CARGA HORARIA POR COMPONENTES DE APRENDIZAJE			**SESIONES SEMANALES**
DOCENCIA	**PRACTICAS DE APLICACIÓN Y EXPERIMENTACIÓN**	**APRENDIZAJE AUTÓNOMO**	4
64	64	64	

Fecha Elaboración	**Fecha de Actualización**	**Fecha de Ejecución**
15/05/2020	15/05/2020	30/11/2020

Descripción de la Asignatura:

La materia Programación Orientada a Objetos, es una asignatura del eje de formación profesional, que se caracteriza por contribuir a la formación de los elementos de competencia y fortalecer las unidades de competencia en análisis, diseño, y construcción de aplicaciones de software, basado en el paradigma orientado a objetos, sus fundamentos y principios, como el encapsulamiento, la abstracción, la herencia, el polimorfismo apoyados, por el lenguaje de programación Java. Esta asignatura se enfoca principalmente en la resolución de problemas complejos del mundo real, y en producir aplicaciones de calidad, empleando principios y prácticas de la Ingeniería de Software, tales como pruebas de unidad, patrones de diseño y los “SOLID Principales”. Se fortalece también con el uso de interfaces gráficas de usuario, y conexión a bases de datos que permiten la adecuada interacción entre el usuario y el computador.

Contribución de la Asignatura:

La asignatura contribuye al resultado de aprendizaje del nivel y es parte sustancial de la formación profesional, los componentes son la solución a problemas orientados a la integración de diferentes aplicaciones e infraestructura tecnológica existente en las organizaciones, bajo el sustento de la programación de computadores.

Resultado de Aprendizaje de la Carrera: (Unidad de Competencia)

Aplica el paradigma de programación orientado a objetos para implementar algoritmos en lenguajes de programación que solucionan problemas básicos en diferentes dominios

Objetivo de la Asignatura: (Unidad de Competencia)

Brindar lo conocimientos esenciales al estudiante para que aprenda a analizar, diseñar, implementar y probar software usando el paradigma de Orientación a Objetos, y de esta manera sea capaz de emplear métodos, técnicas y herramientas ingenieriles para la construcción de aplicaciones robustas y mantenibles mediante el uso de este paradigma.

Resultado de Aprendizaje de la Asignatura: (Elemento de Competencia)

Conceptuales:
Conoce el paradigma basado en objetos y sus diferentes componentes.

Procedimentales:
Resuelve aplicaciones computacionales enfocadas al paradigma basado en objetos. Programación de aplicaciones utilizando el Paradigma Orientado a Objetos

Actitudinales:
Lidera el Trabajo en Equipo con pro-actividad, para el desarrollo de aplicaciones computacionales.

Proyecto Integrador

considerando las características de las acciones prácticas en esta unidad no se han considerado prácticas preprofesionales ni proyectos integradores, sino
redes de aprendizaje concentradas en laboratorios de Ingeniería de Software de la Universidad, donde se desarrollarán algoritmos, código, pruebas de escritorio en el paradigma orientado a objetos; a partir de una especificación de requisitos de un problema.

PERFIL SUGERIDO DEL DOCENTE

TÍTULO Y DENOMINACIÓN

GRADO: Ingeniero en Sistemas e Informática/ Ingeniero en Software/Ing. en Tecnologías de Información / Ing. Ciencias de la computación.

POSGRADO: Master en Ingeniería de Software / Master en Gerencia de Sistemas o afines.

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

CONTENIDOS	
Unidad 1 **Horas/Min:** CONCEPTOS BÁSICOS DEL PARADIGMA DE PROGRAMACIÓN ORIENTADA A OBJETOS Y PRINCIPIOS DE DISEÑO.	**HORAS DE TRABAJO AUTÓNOMO** Prácticas de Aplicación y Experimentación
1. Sistemas de control de versionamiento (VCS) 1.1. Concepto de los SCV 1.2. Terminología utilizada en los SCV 1.3. Características 1.4. Clasificación 1.5. Herramientas o Sw : Git, GitHub 1.6. Funciones comunes de GitHub 1.7. Registro en GitHub **2. Paradigmas de programación** 2.1. Concepto de paradigmas de programación 2.2. Principales paradigmas de programación 2.3. Transición de paradigmas 2.4. Ejemplo de paradigmas de programación **3. Entorno de Desarrollo** 3.1. Conceptos básicos 3.2. Tipos de IDE 3.3. NetBeans 3.4. Características de Instalación 3.5. Consola **4. Revisión de conceptos generales de la POO.** Principios Generales de la Programación Orientada a Objetos. 4.1. Concepto de clase 4.2. Concepto de objeto 4.3. Atributos 4.4. Métodos. 4.5. Ejemplo **5. Modelamiento de clases y Objetos** 5.1. Concepto de UML 5.2. Tipos de UML	

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

5.3. UML: Diagramas de Casos de Uso (Concepto, caractisticas, ejmplo) 5.4. UML: Diagramas de Clases(Concepto, caractisticas, ejmplo 5.5. Identificación de clases de un sistema, uso correcto de identificadores. 5.6. Modificadores de Acceso 5.7. Administración y configuración del UML en NetBeans 5.8. Ejemplo de una Implementación de clases **6. Código limpio** 6.1. Concepto de código limpio 6.2. Características 6.3. Estándares de implementación 6.4. Buenas prácticas de programación. **7. Estructura general de un programa** 7.1. Estructura general de un programa 7.2. Creación de un programa básico POO 7.3. Tipos de datos (PRIMITIVOS Y REFERENCIADOS.) **8. Lectura escritura de datos por consola** 8.1 Entrada de datos 8.2 Salida de datos 8.3. Otras formas de lectura y escritura de datos **9. Excepciones** 9.1. Concepto de excepciones 9.2. Excepciones y errores 9.3. Clases o Tipos de excepción 9.4. Excepciones personalizadas. **10. Encapsulamiento** 10.1. Concepto de encapsulamiento 10.2. Niveles de encapsulamiento 10.3. Modificadores de acceso 10.4. Beneficios del encapsulamiento (POO) 10.5. Desventajas de utilizar encapsulación 10.6. Clases 10.7. Paquetes 10.8. Librerías/bibliotecas, métodos static 10.9. Ejemplo **11. Constructores** 11.1. Concepto 11.2. Tipos de constructores 11.3. Instanciación 11.4. Ejemplo **12 métodos getters, setters.** 12.1. Concepto 12.2. Ejemplo implementación **13. Persistencia de datos** 13.1. Concepto de Archivos	

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

13.2. Manipulación de archivos 13.3. Lectura y escritura de datos en Archivos 13.4. Lectura y escritura de objetos 13.5. Formatos de datos : csv, json 13.6. Colecciones List/Arraylist/LinkedList **14. Arreglos y colecciones** 14.1. Concepto de arreglos 14.2. Ventajas de los arreglos 14.3. Clasificación de los arreglos (Con datos Primitivos y objetos) 14.4. Ejercicio de arreglos 14.5. Concepto de colecciones **15. Relaciones entre clases** 15.1. Concepto de relación 15.2. Tipos de relaciones 15.3. Asociación 15.4. Dependencia 14.5. Agregación 15.5. Composición: Concepto, sintaxis, modelado e Implementación 15.6. Generalización/Especialización (Concepto, ventajas, nomenclatura, reglas, modelado. implementación **16. Revisiones de Código** 16.1. Conceptos de revisión de código 16.2. Revisiones de Código	

ACTIVIDADES DE APRENDIZAJE / HORAS CLASE	
COMPONENTES DE DOCENCIA	22
PRÁCTICAS DE APLICACIÓN Y EXPERIMENTACIÓN	21
HORAS DE TRABAJO AUTONOMO	21
TOTAL HORAS POR UNIDAD	**64**

CONTENIDOS	
Unidad 2 **Horas/Min:**	**HORAS DE TRABAJO AUTÓNOMO**
PRINCIPIOS AVANZADOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS E INTEGRACIÓN DE OBJETOS GRAFICOS Y CONEXIÓN A BASES DE DATOS NO SQL	Prácticas de Aplicación y Experimentación
16. REVISIONES DE CÓDIGO (Continuación). 16.3. Gestión de Defectos (testing). 16.4. Verificación y Validación 16.5. Pruebas Vs Depuración 16.6. Pruebas de unidad **17. Polimorfismo** 17.1. Concepto 17.2. Ventajas 17.3. Sobrecarga de métodos 17.4. Sobre escritura de métodos 17.5. Asignación de objetos a variables de su superclase. 17.6. Ejercicio **18. Interfaces de programación y clases Abstractas**	

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

18.1. Concepto de métodos y clases abstractas	
18.2. Sintaxis de un método y una clase abstracta	
18.3. Ventajas de la abstracción	
18.4. Modelado de una clase abstracta	
18.5. Declaración e implementación de clases abstractas.	
18.6. Interfaces	
18.7. Sintaxis de una interfaz	
18.8. Ventajas de la interfaz	
18.9. Modelado de la interfaz	
18.10. Declaración e Implementación de Interfaces	
18.11. Ejercicio de interface	
18.12. Diferencias entre una interfaz y una clase abstracta	
19. Modelo Vista Controlador	
19.1. Concepto	
19.2. Ventajas	
19.3. Características	
19.4. Arquitectura	
19.5. Implementación	
20. Bases de Datos no SQL	
20.1. Concepto	
20.2. Acceso a base de datos	
20.3. Drivers y Conexión	
20.4. Operaciones CRUD	
21. Componentes y objetos gráficos	
21.1. Widgets (componentes gráficos)	
21.2. Introducción a los componentes gráficos	
21.3. Formularios	
21.4. Menús	
21.5. Tablas	
21.6. Gestión de eventos	
21.7. Integración de componentes gráficos y clases	

ACTIVIDADES DE APRENDIZAJE / HORAS CLASE	
COMPONENTES DE DOCENCIA	21
PRÁCTICAS DE APLICACIÓN Y EXPERIMENTACIÓN	22
HORAS DE TRABAJO AUTONOMO	21
TOTAL HORAS POR UNIDAD	**64**

CONTENIDOS	
Unidad 3 **Horas/Min:**	**HORAS DE TRABAJO AUTÓNOMO**
TECNICA PARA RESOLVER PROBLEMAS EN EL DESARROLLO DE SOFTWARE	Prácticas de Aplicación y Experimentación
22. Principios SOLID	
22.1 Single Responsibility	
22.2 Open/Closed	
22.3 Liskov Substitution	
22.4 Interface Segregation	

2. SISTEMA DE CONTENIDOS Y RESULTADOS DEL APRENDIZAJE

22.5 Dependency Inversion	
22.6 Código entendible, flexible y mantenible	
23. Modularidad	
23.1. Localización de decisiones de diseño	
23.2. Alta cohesión	
23.3. Bajo acoplamiento	
24.Introducción a Patrones de Diseño	
24.1. Concepto generales	
24.2. Importancia de los patrones de diseño	
24.3. Tipos de patrones	
24.3.1. Patrones de creación	
1. Singleton	
2. Abstract Factory	
24.3.2. Patrones de estructura	
1. Composite	
2. Template	
24.3.3. Patrones de comportamiento	
1. Observer	
2. Strategy	
2.5. PROYECTOS DE POO	
2.5.1. PROYECTOS DE POO	

ACTIVIDADES DE APRENDIZAJE / HORAS CLASE	
COMPONENTES DE DOCENCIA	21
PRÁCTICAS DE APLICACIÓN Y EXPERIMENTACIÓN	21
HORAS DE TRABAJO AUTONOMO	22
TOTAL HORAS POR UNIDAD	**64**

7. BIBLIOGRAFÍA BÁSICA/ TEXTO GUÍA DE LA ASIGNATURA

Titulo	Autor	Edición	Año	Idioma	Editorial
Introducción a la programación orientada a objetos	Deitel, Paul J	-	2010	spa	México : Pearson
Programación orientada a objetos y programación estructurada	Pérez, María	-	2014	spa	Estados Unidos de América : [s.n.]

FIRMAS DE LEGALIZACIÓN

NO APROBADO

JORGE EDISON LASCANO
DOCENTE

RUBEN DARIO ARROYO CHANGO
COORDINADOR DE AREA DE CONOCIMIENTO

SONIA ELIZABETH CARDENAS DELGADO
DIRECTOR DE DEPARTAMENTO