Facultad de Ingeniería
Escuela Profesional de Ingeniería Informática

SÍLABO 2024-II

I. DATOS ADMINISTRATIVOS

1. Asignatura	**: TALLER DE DESARROLLO DE SOFTWARE I**
2. Código	: **II 0201**
3. Naturaleza	: Teórico-práctico.
4. Condición	: Obligatorio.
5. Requisitos	: Introducción a la Ingeniería Informática
6. Nro. Créditos	: 4
7. Nro. de horas	: 6 (Teoría = 2 Practica = 4)
8. Semestre Académico	: 2
9. Docente(s)	Juana Segura, Julio Valverde, Edgard De Olazábal
10. Correo Institucional	

II. SUMILLA

Asignatura de teoría y taller. Tiene como propósito capacitar al estudiante para la resolución de problemas medianamente complejos a través de programas de computadora. Aporta a las competencias específicas: Solución de problemas en Ingeniería Informática, Aplica el diseño en ingeniería, Adquiere y aplica nuevos conocimientos.

Al término del curso el estudiante conocerá, comprenderá y aplicará los conceptos básicos de Programación orientada a objetos y resolverá problemas de programación medianamente complejos, planteando casos y desarrollando un proyecto que los involucre.

Síntesis del contenido: (1) POO: clasificación y encapsulamiento. (2) Herencia y polimorfismo. (3) Interfaces. (4) Arreglos de objetos y de otros tipos. (5) Excepciones. (6) Archivos secuenciales. (7) Proyecto de aplicación.

III. COMPETENCIAS

III.1 COMPETENCIAS GENÉRICAS A LAS QUE CONTRIBUYE LA ASIGNATURA

- Pensamiento crítico y creativo
- Autoaprendizaje

III.2 COMPETENCIAS ESPECÍFICAS A LAS QUE CONTRIBUYE LA ASIGNATURA

- Soluciona problemas de Ingeniería
- Práctica moderna de la ingeniería

IV. DESARROLLA EL COMPONENTE DE:

Investigación formativa ( )
Responsabilidad social ( )

V. LOGRO DE LA ASIGNATURA

Al finalizar la asignatura el estudiante:

- Conocerá, comprenderá y aplicará los conceptos básicos de Programación orientada a objetos y resolverá problemas de programación medianamente complejos, planteando casos y desarrollando un proyecto que los involucre.

- Será capaz de desarrollar y mantener de manera económica sistemas de software confiables capaces de satisfacer los requisitos definidos por los clientes, llevando a cabo un proyecto colaborativo con aplicación de buenas prácticas como la depuración de código y ejecución de pruebas unitarias, uso de entornos distribuidos y aplicación eficiente del lenguaje de modelado UML.

VI. PROGRAMACIÓN DE CONTENIDOS

UNIDAD I: Programación Orientada a Objetos: Clasificación y Encapsulamiento		
LOGRO DE APRENDIZAJE: Al finalizar la unidad, el estudiante conoce y comprende el concepto de "Objeto", Clase, Atributos y Métodos. Utiliza los objetos para la resolución de problemas computacionales. Conoce la Notación UML para el diagrama de clases.		
Semana	**Contenido**	
1	Clasificación. Clases y objetos. Métodos. Encapsulamiento. Modificadores de acceso. Propiedades de la Programación Orientada a Objetos: Encapsulamiento, herencia y polimorfismo.	
2	Sobrecarga de métodos. Miembros de instancia, miembros de clase (estáticos). Lenguaje de Modelado UML: Diagrama de clases.	
3	Encapsulamiento. Modificadores de acceso. Sobrecarga de métodos. Miembros de instancia, miembros de clase (estáticos). Clases abstractas, métodos abstractos. Interfaces.	

UNIDAD II: Herencia y polimorfismo		
LOGRO DE APRENDIZAJE: Al finalizar la unidad, el estudiante utiliza la herencia para definir una jerarquía de Clases. Resuelve problemas más elaborados con las propiedades de herencia avanzadas. Entiende y aplica el polimorfismo para la resolución de problemas con códigos fuente más compactos y claros. Aplica el UML para refinar el diagrama de clases.		
Semana	**Contenido**	
4	Herencia: Conceptos y aplicaciones. Superclases, Subclases, Jerarquías de clases Subclases. Herencia: Tipos. Herencia Simple. Herencia múltiple. Interfaces para herencia múltiple.	
5	Polimorfismo estático: Sobrecarga de métodos, sobrecarga de constructores, sobreposición de métodos. Métodos virtuales. Clases Abstractas. Interfaces. El IDE. Herramientas de Depuración de códigos, uso del Trace, Watch, Warnings, Sugerencias y Refactoring.	
6	Polimorfismo dinámico a partir de la sobreescritura de métodos. Las interfaces y el polimorfismo. Implementación de interfaces. Principio de sustitución de Liskov. Trabajo colaborativo. Respaldo de código en la nube. Gestión de repositorios.	

UNIDAD III: Interfaces y pruebas unitarias		
LOGRO DE APRENDIZAJE: Al finalizar la unidad, el estudiante utiliza las interfaces para la modularización del código en la resolución de problemas más elaborados. Utiliza las pruebas Unitarias y el desarrollo rojo-verde en un esquema Test Driven Development.		
Semana	**Contenido**	
7	Polimorfismo dinámico a partir de la sobre posición de métodos. Las interfaces y el polimorfismo. Elaboración de pruebas unitarias.	
8	**Semana de Exámenes Parciales**	

UNIDAD IV: Arreglos de objetos y de otros tipos		
LOGRO DE APRENDIZAJE: Al finalizar la unidad, el estudiante resuelve problemas de arreglos de tipos básicos y de objetos.		
Semana	**Contenido**	
9	Arreglos Unidimensionales: el Vector. Arreglos Bidimensionales: Definición y resolución de problemas utilizando matrices. Aplicaciones. Pruebas unitarias con arreglos.	

10	Arreglos de objetos. Arreglos de objetos. Resolución de problemas con vectores y matrices de objetos. Aplicaciones. . Desarrollo colaborativo de proyecto de software.
11	Cadenas de caracteres. Uso de librerías predefinidas para el manejo de cadenas de caracteres. Aplicaciones. Desarrollo colaborativo de proyecto de software.

UNIDAD V: Excepciones y Archivos secuenciales	
LOGRO DE APRENDIZAJE: Al finalizar la unidad, el estudiante emplea el manejo de excepciones para la creación correcta de librerías. Resuelve problemas con archivos secuenciales.	
Semana	**Contenido**
12	Manejo de Excepciones: Generalidades, Excepciones predefinidas y definidas por el usuario. Depuración de errores en el IDE, depuración de excepciones. Teoría sobre las excepciones. Tipos de excepciones. Aplicaciones. Entornos de desarrollo en la nube
13	Concepto de archivos secuenciales. Archivos de texto y binarios. Serialización. Entornos de desarrollo en la nube.
14	Uso de librerías predefinidas para el manejo de flujos de caracteres con archivos secuenciales. Uso de librerías predefinidas para el manejo de flujos de bytes con archivos secuenciales.
15	Proyecto de aplicación. Demostración del cumplimiento de requerimientos ágiles. Historial de trabajo colaborativo. Casos de prueba. Pruebas en la nube. Documentación de entregables.
16	**Semana de Exámenes Finales**
17	**Entrega de Notas**

VII. ESTRATEGIAS DIDÁCTICAS

Aprendizaje Basado en Proyectos, Problemas. Aprendizaje Colaborativo, Aprendizaje Basado en Investigación, Talleres, Laboratorios de programación.

Se podrán desarrollar actividades sincrónicas (que los estudiantes realizarán al mismo tiempo con el docente) y asincrónicas (que los estudiantes realizarán independientemente fortaleciendo su aprendizaje autónomo. La planificación y ejecución de las sesiones de aprendizaje deberán considerar actividades que se organizarán de la siguiente manera:

Exploración: preguntas de reflexión vinculada con el contexto, otros.
Problematización: conflicto cognitivo de la unidad, otros.
Motivación: bienvenida y presentación del curso, otros.
Presentación: PPT, otros.
Práctica: resolución individual de un problema, resolución colectiva de un problema, otros.
Evaluación de la unidad: presentación del resultado o producto.
Extensión / Transferencia: presentación de la resolución individual de un problema.

VIII. RECURSOS

- Aula con recursos Audiovisuales
- Laboratorio con computadoras personales
- Software de los temas desarrollados
- Visual Studio 2022 u otro IDE similar.
- Software de Oficina
- Aula virtual

IX. EVALUACIÓN

Las evaluaciones se realizarán a lo largo del semestre con el propósito de determinar en qué medida el estudiante va logrando las competencias de la asignatura.

Las actividades de enseñanza se complementarán con actividades de evaluación continua (AEC) tales como: talleres, proyectos, trabajos, simulaciones, exposiciones, casos, participaciones en las sesiones de clases, entre otras, para las cuales se podrán seleccionar los instrumentos que el docente estime conveniente, además cuando menos de una rúbrica como recurso educativo.

1.- Responsabilidad del alumno respecto a la asistencia a clases. De acuerdo con las normas establecidas, se tendrá en cuenta este criterio en la evaluación final. Los alumnos que hayan alcanzado el 30% de inasistencias, no podrán rendir sus evaluaciones y su calificación será de cero (0).
2.- Se tendrá en cuenta la participación activa de los alumnos durante las clases, estudio de casos, talleres evaluativos y presentaciones.
3.- Se aplicarán las evaluaciones escritas en las fechas establecidas en el calendario de clases.
El promedio final de la asignatura se obtendrá de la manera siguiente: 4 Laboratorios calificados obligatorios, y un Proyecto Final de curso. Ninguna de las notas se elimina.

Fórmula: PF = ((LB1 + LB2 + LB3 + LB4)/4) * 0.6 + PRO1 * 0.4

En donde:

PF	: Promedio Final
LB1-4	: Laboratorio calificado
PRO1	: Proyecto final

IX. REFERENCIAS

Bibliografía Básica.

1. JOYANES AGUILAR, Luis. **Fundamentos de Programación – Algoritmos, Estructuras de Datos y Objetos**. 2003. Mc Graw Hill, España.
2. CEBALLOS, Javier C# Curso de Programación 2011 2da. Edición Editorial Rama S. A.
3. CAIRÓ, Osvaldo. **Metodología de la Programación**. © 2005 ALFAOMEGA GRUPOEDITOR, S.A. de C.
4. DEITEL, P.J/H,M, **C# How to Program.** 2012. Fifth Edition Pearson.
5. Martin, Robert. **Código Limpio.** 2009. XcUIDI. Traducción de Gómez Celador.
6. BOBADILLA, Jesús. **Java a través de ejemplos**. 2006. México. Editorial Ra-Ma.
7. OVIEDO, Efraín. **Lógica de Programación.** 2004. Colombia. ECO Ediciones.
8. Beck, K., & Andres, C. (2004). **Extreme Programming Explained: Embrace Change** (2nd ed.). Addison-Wesley.
9. Sutherland, J. Scrum: **The Art of Doing Twice the Work in Half the Time**. 2014. Crown Business.
10. Spinellis, D. **Code Quality: The Open Source Perspective**. 2006. Addison-Wesley.
11. Fowler, M. **UML Distilled:** A Brief Guide to the Standard Object Modeling Language (3rd ed.). 2004. Addison-Wesley.
12. Lewis, J., & Loftus, W. Java Software Solutions: Foundations of Program Design (10th ed.). 2019. Pearson.

Referencias Web

1. https://www.w3schools.com/
   Tutoriales de los principales lenguajes de programación: C#, Java, Python con Programación Orientada a Objetos.
2. https://visualstudio.microsoft.com/es/vs/getting-started/
3. https://netbeans.apache.org/tutorial/main/kb/docs/
4. https://www.youtube.com/playlist?list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B
   Curso completo de C# en español