1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura :	**Fundamentos de Programación**
Carrera :	**Ingeniería informática e Ingeniería en Tecnologías de la Información y Comunicaciones**
Clave de la asignatura :	**AEF-1032**
SATCA[1]	**3-2-5**

2.- PRESENTACIÓN

Caracterización de la asignatura.

La asignatura de Fundamentos de Programación aporta al perfil del egresado, los conocimientos, habilidades, metodologías, así como capacidades de análisis y síntesis, para plantear la solución de problemas susceptibles de ser computarizados, a través de diagramas de flujo, pseudocódigo, algoritmos y el paradigma de programación orientada a objetos.

Intención didáctica.

Este programa de estudios se sugiere eminentemente práctico, es decir, el profesor propone el planteamiento de un problema y el estudiante deberá resolverlos mediante diversas técnicas, por mencionar algunas: diagramas de flujo, pseudocódigo y herramientas de modelado; con el monitoreo del facilitador.

La unidad uno, introduce al estudiante, de forma teórica, en la evolución de los paradigmas de la programación, el profesor debe asegurarse de que el estudiante conoce y comprende la competencia que está en proceso de adquirir a partir de su fundamentación; al mismo tiempo que se unifica la conceptualización para que sea común y se relacione con el resto del programa de forma práctica.

En la segunda unidad, se desarrolla la parte teórico formal del modelo de las 6'D. Se describen sus etapas, así como los pasos y resultados que se deben esperar de cada una de ellas, este contenido se sugiere relacionarlo con el modelo de objetos.

La tercera unidad, se enfoca en los conceptos de la Programación Orientada a Objetos (POO), las reglas para construir diagramas, pseudocódigo y el uso de expresiones de declaración, asignación, condicionales y estructuras cíclicas.

El profesor debe hacer énfasis en los beneficios que tiene la POO combinada con la programación estructurada para obtener soluciones robustas y funcionales. Se sugiere que en esta unidad se realicen actividades integradoras, desarrollando prácticas donde se requiera involucrar los diferentes conceptos en ejercicios, utilizando editores de diagramas de flujo que permitan generar código y/o pseudocódigo en algún lenguaje en particular.

Dentro de la cuarta unidad, se plantea el enfoque en los conceptos de la Programación Orientada a Objetos (POO) en mayor profundidad, para conocer las peculiaridades de este tipo de programación; apoyándose en la programación estructurada, las características del modelo orientado a objetos, sus elementos primordiales, la representación grafica de los

[1] *Sistema de Asignación y Transferencia de Créditos Académicos*

diseños, crear objetos en prácticas y reconocer la interrelación entre ellos. A la vez se tratan los aspectos fundamentales sobre modelado, algoritmos y las herramientas de diseño conocidas como diagramas de flujo y pseudocódigo, y sobre la programación utilizando un lenguaje de POO. Sugiriéndose al profesor que utilice actividades integradoras y que se considere un avance de proyecto final, incluyendo los temas vistos dentro del contenido de esta unidad.

En la quinta unidad, se emplean la estructura de datos más simple: la variable, y se complementa con el estudio de tipos de datos primitivos, y la conversión de datos. También se aplican las estructuras lógicas. Estas estructuras son conceptos propios de la programación estructurada y son utilizadas por la POO y ofrece una forma de codificar programas con una mayor claridad y sencillez. Se sugiere presentar diferentes tipos de problemas para desarrollar las capacidades lógicas de los estudiantes y analizar las soluciones. También es importante que se realicen actividades integradoras, desarrollando prácticas donde se requiera involucrar los diferentes conceptos en ejercicios. Se sugiere hacer énfasis en el análisis, construcción y diseño de programas a través de ejercicios en clase y extra clase, se sugiere usar una herramienta integradora como Eclipse, Visual Studio o Netbeans que incluye diversos plug-ins, para que los estudiantes modelen y codifiquen las soluciones.

La sexta unidad, trata la modularidad, la cual permite la reutilización y facilita la verificación y depuración del código. En POO, los módulos están directamente relacionados con los objetos. El objetivo de esta unidad es que el estudiante aplique todos los conceptos vistos en las unidades anteriores y desarrolle la capacidad de programar en módulos/métodos. Se sugiere hacer escenarios en clase y extra clase, usar una herramienta integradora como Eclipse, Visual Studio o Netbeans, que incluye diversos plug-ins, para que los estudiantes modelen y codifiquen las soluciones.

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**	
▪ Analizar y solucionar problemas informáticos y representar su solución mediante herramientas de software orientado a objetos.	*Competencias instrumentales* • Pensamiento lógico, algorítmico, heurístico, analítico y sintético • Capacidad de análisis y síntesis. • Capacidad de diseñar modelos abstractos. • Procesar e interpretar datos. • Representar e interpretar conceptos en diferentes formas: Gráfica, escrita y verbal. • Habilidades básicas para elaborar diagramas. • Solución de problemas. • Potenciar las habilidades para el uso de lenguajes de programación • Toma de decisiones. • Lectura en idioma inglés. *Competencias interpersonales* • Capacidad crítica y autocrítica. • Trabajo en equipo. • Habilidades interpersonales. • Compromiso ético. *Competencias sistémicas* • Habilidad de planificar como un todo y diseñar nuevos sistemas. • Capacidad de aplicar los conocimientos en la práctica. • Habilidades de investigación. • Capacidad de aprender. • Creatividad. • Habilidad para trabajar en forma autónoma. • Búsqueda del logro.	

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	Participantes	Evento
Instituto Tecnológico Superior de Puerto Vallarta del 10 al 14 de agosto de 2009.	Representantes de los Institutos Tecnológicos de: Aguascalientes, Apizaco, Superior de Centla, Chetumal, Ciudad Cuauhtémoc, Ciudad Madero, Comitán, Delicias, León, Superior de Misantla, Pachuca, Pinotepa, Puebla, Superior de Puerto Vallarta, Roque, Tepic, Tijuana, Tuxtla Gutiérrez y Villahermosa.	Reunión Nacional de Diseño e Innovación Curricular para el Desarrollo y Formación de Competencias Profesionales de la Carrera de Ingeniería en Tecnologías de la Información y Comunicaciones.
Desarrollo de Programas en Competencias Profesionales por los Institutos Tecnológicos del 17 de agosto de 2009 al 21 de mayo de 2010.	Academias de Ingeniería en Tecnologías de la Información y Comunicaciones de los Institutos Tecnológicos: Comitancillo, El Salto, Los Mochis, Oaxaca y Morelia.	Elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la Carrera de Ingeniería en Tecnologías de la Información y Comunicaciones.
Instituto Tecnológico de Villahermosa del 24 al 28 de mayo de 2010.	Representantes de los Institutos Tecnológicos de: Aguascalientes, Apizaco, Superior de Centla, Chetumal, León, Pachuca, Puebla, Roque, Tepic, Tuxtla Gutiérrez y Villahermosa.	Reunión Nacional de Consolidación de los Programas en Competencias Profesionales de la Carrera de Ingeniería en Tecnologías de la Información y Comunicaciones.
Instituto Tecnológico Saltillo, del 5 al 9 de octubre de 2009.	Representantes de los Institutos Tecnológicos de: Apizaco, Cerro Azul, Chetumal, Ciudad Juárez, Ciudad Madero, Superior de Coatzacoalcos, Colima, Comitancillo, Conkal, Durango, El Llano Aguascalientes, El Salto, Superior de Fresnillo, Huejutla, Superior de Lerdo, Linares, Los Mochis, Mexicali, Morelia, Oaxaca, Superior del Occidente del Estado de Hidalgo, Ocotlán, Orizaba, Piedras Negras, Pinotepa, Saltillo, San Luis Potosí, Tapachula, Tijuana, Torreón, Tuxtepec, Superior de Valladolid, Valle del Guadiana, Superior de Zacapoaxtla y Zacatecas.	Reunión Nacional de Diseño e Innovación Curricular para el Desarrollo y Formación de Competencias Profesionales de la Carrera de Ingeniería Informática.

Lugar y fecha de elaboración o revisión	Participantes	Evento
Desarrollo de Programas en Competencias Profesionales por los Institutos Tecnológicos del 12 de octubre de 2009 al 19 de febrero de 2010.	Academias de Ingeniería Informática de los Institutos Tecnológicos: El Llano, Ciudad Juárez, Colima, Comitancillo, Los Mochis, Oaxaca y Tijuana.	Elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la Carrera de Ingeniería Informática.
Instituto Tecnológico Superior de Poza Rica, del 22 al 26 de febrero de 2010.	Representantes de los Institutos Tecnológicos de: Apizaco, Cerro Azul, Chetumal, Ciudad Juárez, Ciudad Madero, Superior de Coatzacoalcos, Colima, Comitancillo, Conkal, Durango, El Llano Aguascalientes, El Salto, Superior de Fresnillo, Huejutla, Superior de Lerdo, Los Mochis, Mexicali, Morelia, Oaxaca, Superior del Occidente del Estado de Hidalgo, Ocotlán, Orizaba, Piedras Negras, Pinotepa, Saltillo, San Luis Potosí, Tapachula, Tijuana, Torreón, Tuxtepec, Superior de Valladolid, Valle del Guadiana, Superior de Zacapoaxtla y Zacatecas.	Reunión Nacional de Consolidación de los Programas en Competencias Profesionales de la Carrera de Ingeniería Informática.
Instituto Tecnológico de Aguascalientes, del 15 al 18 de Junio de 2010.	Representantes de los Institutos Tecnológicos de: Acapulco, Aguascalientes, Altiplano de Tlaxcala, Apizaco, Boca del Río, Ciudad Cuauhtémoc, Ciudad Juárez, Ciudad Madero, Ciudad Victoria, Celaya, Chetumal, Chihuahua, Chilpancingo, Superior de Coatzacoalcos, Colima, Cuautla, Durango, Superior de El Dorado, El Llano de Aguascalientes, Huejutla, Huatabampo, Superior de Huixquilucan, Iguala, Superior de Irapuato, La Laguna, La Paz, León, Linares, Superior de Macuspana, Matamoros, Mazatlán, Mérida, Mexicali, Nuevo Laredo, Superior del Oriente del Estado de Hidalgo, Orizaba, Pachuca, Superior de Pátzcuaro, Superior de Poza Rica, Superior de Progreso,	Reunión Nacional de Implementación Curricular y Fortalecimiento Curricular de las asignaturas comunes por área de conocimiento para los planes de estudio actualizados del SNEST.

Lugar y fecha de elaboración o revisión	Participantes	Evento
	Puebla, Superior de Puerto Vallarta, Querétaro, Reynosa, Roque, Salina Cruz, Saltillo, San Luis Potosí, Superior de Tacámbaro, Superior de Tamazula de Gordiano, Tehuacán, Tijuana Tlaxiaco, Toluca, Torreón, Tuxtepec, Superior de Venustiano Carranza, Veracruz, Villahermosa, Zacatecas, Superior de Zongólica.	
Instituto Tecnológico de Aguascalientes, del 15 al 18 de Junio de 2010.	Representantes de los Institutos Tecnológicos de: Aguascalientes, Apizaco, Cd. Madero, Colima, La Paz, Toluca y Villahermosa.	Elaboración del programa de estudio equivalente en la Reunión Nacional de Implementación Curricular y Fortalecimiento Curricular de las asignaturas comunes por área de conocimiento para los planes de estudio actualizados del SNEST.

5.- OBJETIVO GENERAL DEL CURSO

Analizar y solucionar problemas informáticos y representar su solución mediante herramientas de software orientado a objetos.

6.- COMPETENCIAS PREVIAS

- Habilidad para el manejo de la computadora.
- Navegación en Internet.
- Capacidad de análisis y síntesis.
- Manejar herramientas de software mediante menús.
- Manejar comandos y funciones básicas en algún sistema operativo de computadora.
- Aplicar lógica matemática en la solución de problemas informáticos.

7.- TEMARIO

Unidad	Temas	Subtemas
1.	Fundamentos de programación orientada a objetos	1.1. Evolución de la programación. 1.2. Conceptos fundamentales de la Programación Orientada a Objetos. 1.3. Lenguajes orientados a objetos 1.4. Relaciones entre clases y objetos. 1.5. Papel de clases y objetos en el análisis y el diseño.
2.	Metodología de Solución de Problemas	2.1. Descripción del problema (enunciado). 2.2. Definición de solución (especificaciones). 2.3. Diseño de la solución (modelado). 2.4. Desarrollo de la solución (codificación). 2.5. Depuración y pruebas (pruebas). 2.6. Documentación (manuales).
3.	Herramientas de programación	3.1. Simbología. 3.2. Reglas para la construcción de diagramas. 3.3. Pseudocódigo. 3.4. Tipos de datos y expresiones. 3.5. Estructuras lógicas.
4.	Programación orientada a objetos y modelado	4.1. Características del modelo orientado a objetos. 4.2. Elementos primordiales en el modelo de objetos. 4.3. Representación gráfica del diseño. 4.4. Relación entre la programación orientada a objetos y la estructurada.

5.	Implementación Orientada a Objetos	5.1. Estructura de una clase. 5.2. Elementos de una clase. 5.3. Clase principal. 5.4. Crear objetos.
6.	Modularidad	6.1. Declaración de métodos. 6.2. Métodos de clase. 6.3. Métodos de instancia.

8.- SUGERENCIAS DIDÁCTICAS

El docente debe dominar ampliamente los contenidos de esta materia para que pueda abordar cada uno de los temas en su totalidad, además contar con la capacidad para coordinar, trabajar de forma individual y/o en equipo y, orientar el trabajo del estudiante; potenciar en él la capacidad de análisis y síntesis, el trabajo cooperativo y la toma de decisiones. Mostrar flexibilidad en el seguimiento del proceso formativo y propiciar la interacción entre los estudiantes. Tomar en cuenta el conocimiento de los estudiantes como punto de partida y no como obstáculo para la construcción de nuevos conocimientos.

- Emplear herramientas computacionales para diseño y modelado.
- Uso de un portal de Internet para apoyo didáctico de la materia.
- Definir los lineamientos de documentación que deberán contener las tareas.
- Coordinar la realización de modelos orientados a objetos a partir de entidades del mundo real, utilizando ejemplos simples del entorno del estudiante.
- Mostrar al estudiante programas completos de menor a mayor grado de dificultad y con base en cada una de las instrucciones que los componen.
- Enseñar la sintaxis de las diferentes sentencias a utilizar.
- Utilizar el aprendizaje basado en problemas, trabajando en grupos pequeños, para sintetizar y construir el conocimiento necesario para resolver problemas relacionados con situaciones reales.
- Solicitar al estudiante, la elaboración de los programas de ejemplo en la computadora.
- Solicitar al estudiante propuestas de problemas a resolver y que sean significativas para él.
- Propiciar el uso de terminología técnica apropiada.
- Propiciar que el estudiante experimente con diferentes programas encontrados en revistas, Internet y libros de la especialidad, que lo lleven a descubrir nuevos conocimientos.
- Fomentar el trabajo en equipo.
- Elaborar en coordinación con los estudiantes una guía de ejercicios para actividades extra clase.
- Realizar actividades donde se fomente el uso de la lógica y de la capacidad de análisis de datos.
- Proponer problemas que permitan al estudiante la integración de contenidos de la asignatura y entre distintas asignaturas, para su análisis y solución.
- Cuando los temas lo requieran, utilizar medios audiovisuales para una mejor comprensión del estudiante.
- Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes.
- Propiciar el uso de las nuevas tecnologías en el desarrollo de los contenidos de la asignatura.
- Propiciar la planeación y organización del proceso y modelado de programación.
- Fomentar actividades grupales que propicien la comunicación, el intercambio, el argumentado de ideas, la reflexión, la integración y la colaboración entre los estudiantes.
- Propiciar en el estudiante, el desarrollo de actividades intelectuales de inducción-deducción y análisis-síntesis, encaminadas hacia la investigación, la aplicación de conocimientos y la solución de problemas.
- Desarrollar actividades prácticas que evidencien el desarrollo de habilidades para la experimentación, tales como: observación, identificación, manejo y control de variables y datos relevantes, planteamiento de hipótesis, de trabajo en equipo, entre otros.

- Desarrollar actividades de aprendizaje que propicien la aplicación de los conceptos, modelos y metodologías que se van aprendiendo en el desarrollo de la asignatura.
- Propiciar el uso adecuado de conceptos, y de terminología científico-tecnológica de la asignatura con otras asignaturas.
- Proponer problemas que permitan al estudiante la integración de contenidos de la asignatura y entre otras asignaturas, para su análisis y solución.
- Relacionar los contenidos de la asignatura con el cuidado del medio ambiente; así como con las prácticas de una ingeniería con enfoque sustentable.
- Observar y analizar fenómenos y problemáticas propias del campo ocupacional.
- Desarrollar feria de proyectos.

9.- SUGERENCIAS DE EVALUACIÓN

La evaluación debe ser continua y cotidiana por lo que se debe considerar el desempeño en cada una de las actividades de aprendizaje, haciendo especial énfasis en:

- Elaborar mapa conceptual de fundamentos de POO.
- Construir tabla comparativa de los paradigmas de programación.
- Generar reportes de diferentes fuentes de información.
- Determinar un problema para desarrollar proyecto final a realizar.
- Desarrollar diagramas UML.
- Estructurar diagramas de flujo.
- Reportar pruebas de escritorio.
- Implementar soluciones a problemas planteados.
- Integrar los productos obtenidos de cada unidad al proyecto final.

10.- UNIDADES DE APRENDIZAJE

Unidad 1: Fundamentos de programación orientada a objetos

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Analizar la evolución de los paradigmas de programación para comprender las bases de la POO. Comprender los conceptos de la programación orientada a objetos, para conocer las peculiaridades de la POO frente a la programación estructurada, y su aplicación en situaciones del mundo real.	• Investigar en fuentes diversas de información las características principales de los diferentes paradigmas de programación y elaborar un informe. • Analizar y discutir en el aula la investigación realizada en el punto anterior, donde se resalten las diferencias identificadas. • Comparar las ventajas y desventajas de la programación estructurada y la programación orientada a objetos. • Realizar un mapa conceptual sobre los tipos de software y los conceptos básicos de programación. • Uso de un portal de Internet para apoyo didáctico de la materia. • Ejercicios en clase para aplicar objetos de la vida real en POO. • Investigar en diferentes bibliografías las características del paradigma de POO. • Desarrollar escenarios en clase para generar intercambio, discusiones y lluvias de ideas. • Identificar y hacer clasificaciones de objetos cotidianos, discutir en el aula los criterios para realizar tal clasificación y resaltar el concepto de abstracción. • Seleccionar un objeto cotidiano y representar su ciclo de vida. • Mediante casos cotidianos, identificar el concepto de herencia y polimorfismo.

Unidad 2: Metodología de solución de problemas

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Aplicar el modelo de las 6'D para definir las etapas para solución de problemas utilizando computadora.	• Uso de un portal de Internet para apoyo didáctico de la materia. • Ejercicios en clase para el modelo 6'D en la POO. • Desarrollar escenarios en clase para generar intercambio, discusiones y conclusiones.

Unidad 3: Herramientas de programación

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Diseñar diagramas UML, algoritmos y pseudocódigo empleando el paradigma de POO para la solución de problemas.	• Realizar prácticas de búsqueda de información a través de diferentes navegadores o buscadores de información. • Investigación en diversa bibliografía y tutoriales. • Emplear software para diseño y validación de diagramas de flujo. • Emplear software para generar código a partir de diagramas de flujo. • Emplear software para diseño de diagramas UML. • Trabajo en equipo para la solución de casos prácticos.

Unidad 4: Programación orientada a objetos y modelado

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Aplicar estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.	• Realizar prácticas de búsqueda de información a través de diferentes navegadores o buscadores de información. • Investigación en diversa bibliografía y tutoriales. • Emplear software para diseño y validación de diagramas de flujo. • Emplear software para generar código a partir de diagramas de flujo. • Emplear software para diseño de diagramas UML. • Trabajo en equipo para la solución de casos prácticos. • Realizar ejercicios de codificación de expresiones aritméticas y lógicas en un lenguaje de programación.

	• Buscar la información necesaria para Instalar y configurar el compilador del lenguaje de programación a utilizar • Compilar y ejecutar un programa modelo. • Realizar cambios en expresiones lógicas y algebraicas de un programa modelo y analizar los resultados obtenidos.

Unidad 5: Implementación orientada a objetos

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Aplicar estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.	• Realizar prácticas de búsqueda de información a través de diferentes navegadores o buscadores de información. • Investigación en diversa bibliografía y tutoriales. • Emplear software para diseño y validación de diagramas de flujo. • Emplear software para generar código a partir de diagramas de flujo. • Emplear software para diseño de diagramas UML. • Trabajo en equipo para la solución de casos prácticos. • Realizar ejercicios de codificación de expresiones aritméticas y lógicas en un lenguaje de programación. • Buscar la información necesaria para Instalar y configurar el compilador del lenguaje de programación a utilizar • Compilar y ejecutar un programa modelo. • Realizar cambios en expresiones lógicas y algebraicas de un programa modelo y analizar los resultados obtenidos. • Elaborar ejercicios implementando clases simples. • Desarrollar ejercicios usando las clases mediante objetos.

Unidad 6: Modularidad

Competencia específica a desarrollar	*Actividades de Aprendizaje*
Implementar métodos para diseñar módulos independientes y robustos y que correspondan a soluciones en el mundo real.	• Realizar ejemplos de clase que requieran métodos con parámetros (mensajes) • Analizar las distintas formas de paso de parámetros (mensajes). • Casos prácticos.

11.- FUENTES DE INFORMACIÓN

1. Hjalmar Jacobson, Ivar. El Lenguaje Unificado de Modelado Guía del Usuario. 2a. edición. Ed. Addison Wesley.
2. Flores Cueto, Juan José. Método de las 6'D UML – Pseudocódigo – Java Enfoque Algorítmico, Serie Textos Universitarios Facultad de Ingeniería y Arquitectura. ed. Universidad de San Martín de Porres, (http://books.google.com/).
3. Joyanes Aguilar, Luis; Fernández Azuela, Matilde y Rodríguez Baena, Luis. Fundamentos de Programación Libro de Problemas Algoritmos Estructura de Datos y Objetos. 2a. edición. Ed. McGraw Hill.
4. Ramírez, Felipe. Introducción a la Programación, Algoritmos y su Implementación en Vb.Net, C#, Java y C++. 2a. edición. Ed. Alfaomega Grupo Editor.
5. Cairo Battistutti, Osvaldo. Metodología de la Programación, Algoritmos Diagramas de Flujo y Programas. 3a. edición. Ed. Alfaomega Grupo Editor.
6. Martin Robert, C. UML for Java(TM) Programmers. Ed. Robert C. Martin Series, Pearson. 2003.
7. Grady Booch, James. Rumbaugh E., Greg Perry. Aprendiendo Principios de Programación en 24 horas. Ed. Prentice Hall.
8. Sintes, Anthony. *Aprendiendo Programación Orientada a Objetos en 21 Lecciones Avanzadas*. Ed. Pearson Educación, 2002.

12.- PRÁCTICAS PROPUESTAS

- Elaborar mapa conceptual de los paradigmas de programación.
- Determinar un problema para desarrollar proyecto final a realizar.
- Describir las características y funciones de componentes de objetos utilizados cotidianamente tales como: video, casetera, horno de microondas, teléfono público, refrigerador público expendedor de refresco, entre otros.
- Modelar clases básicas como: empleado, estudiante, cadena de caracteres, vector, entre otros.
- Elaborar un diagrama con las etapas de la solución de problemas.
- Utilizando UML obtener el diagrama de clases, el diagrama interacción y los diagramas de estado para un problema.
- Se pueden utilizar problemas presentados por el profesor o utilizar problemas del mundo real presentados por el estudiante.
- Utilizando los diagramas de clases obtenidos en clase elaborar algoritmos para representar el comportamiento de una clase.
- Elaborar ejercicios que impliquen el uso de operadores, operandos y expresiones.
- Elaborar definición de clases utilizando un lenguaje de programación a partir de problemas proporcionados por el profesor.
- Implementar aplicaciones que utilicen clases con comportamientos que impliquen el uso de estructuras secuenciales y expresiones aritméticas y lógicas.
- Implementar aplicaciones que utilicen clases con comportamientos que impliquen el uso de estructuras selectivas, haciendo uso de una herramienta de depuración de aplicaciones.
- Implementar aplicaciones que utilicen clases con comportamientos que impliquen el uso de estructuras repetitivas, haciendo uso de una herramienta de depuración de aplicaciones.
- Implementar métodos aplicando diferentes algoritmos.
- Identificar y realizar clasificaciones de objetos cotidianos.
- Utilizar software para diseño y validación de diagramas de flujo.
- Utilizar software para generar código a partir de diagramas de flujo.

- Utilizar software para diseño de diagramas UML.
- Realizar codificación de expresiones aritméticas y lógicas en lenguaje de programación.
- Desarrollar feria de proyectos.