UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

COORDINACIÓN GENERAL DE FORMACIÓN BÁSICA
COORDINACIÓN GENERAL DE FORMACIÓN PROFESIONAL Y VINCULACIÓN UNIVERSITARIA
PROGRAMA DE UNIDAD DE APRENDIZAJE

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de Ingeniería, Mexicali; Facultad de Ingeniería, Arquitectura y Diseño, Ensenada y Facultad de Ciencias Químicas e Ingeniería, Tijuana.

2. Programa Educativo: Ingeniero en Computación

3. Plan de Estudios:

4. Nombre de la Unidad de Aprendizaje: Programación Orientada a Objetos

5. Clave:

6. HC: 01 **HL:** 02 **HT:** 02 **HPC:** 00 **HCL:** 00 **HE:** 01 **CR:** 06

7. Etapa de Formación a la que Pertenece: Disciplinaria

8. Carácter de la Unidad de Aprendizaje: Obligatoria

9. Requisitos para Cursar la Unidad de Aprendizaje: Ninguno

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
REGISTRADO
17 OCT 2019
REGISTRADO
COORDINACIÓN GENERAL DE FORMACIÓN BÁSICA

Equipo de diseño de PUA

J. Reyes Juárez Ramírez
Manuel Castañón Puga
Sergio Omar Infante Prieto

Vo.Bo. de Subdirectores de Unidades Académicas

Alejandro Mungaray Moctezuma
Humberto Cervantes de Ávila
Rocío Alejandra Chávez Santoscoy

Fecha: 17 de octubre de 2019

II. PROPÓSITO DE LA UNIDAD DE APRENDIZAJE

La programación orientada a objetos permite al profesionista resolver problemas del mundo real, mediante la abstracción utilizando el lenguaje de programación de actualidad, el cual es de carácter universal que permite desarrollar aplicaciones que corren en cualquier arquitectura de hardware y que es compatible con muchas plataformas de software y con la Web. El Paradigma orientado a objetos es uno de los más utilizados debido a su potencial para crear arquitecturas robustas, fáciles de mantener y con alto nivel de reusabilidad. Esta unidad de aprendizaje de carácter teórico práctico, puede ser impartida en cualquier lenguaje orientado a objetos. Permite al participante que ya tiene conocimientos de programación, ver la programación desde otra perspectiva y forma de pensar, además, le proporciona las bases necesarias para comprender los ambientes de programación visual.

Se ubica en la etapa disciplinaria con carácter obligatorio y pertenece al área de conocimiento Ciencias de la Ingeniería.

III. COMPETENCIA DE LA UNIDAD DE APRENDIZAJE

Aplicar el paradigma de programación orientada a objetos en la solución de problemas de procesamiento de información, empleando un lenguaje de modelado y un lenguaje de programación orientado a objetos, para construir componentes de software reutilizables y fáciles de mantener, con actitud analítica y creativa.

IV. EVIDENCIA(S) DE DESEMPEÑO

1. Prototipo de proyecto de tamaño mediano que incluya códigos fuentes y ejecutables y diseño arquitectónico de clases en un lenguaje de modelado a objetos.
2. Reporte integrado con la documentación técnica del desarrollo de una biblioteca de clases basada en un análisis y diseño orientado a objetos.

V. DESARROLLO POR UNIDADES

UNIDAD I. Abstracción y encapsulamiento

Competencia:
Aplicar los principales elementos de una clase como entidad básica de un programa orientado a objetos, distinguiendo entre la función y declaración de los atributos y la declaración y función de los métodos, para reproducir el estado y el comportamiento de una entidad del mundo real, con actitud analítica, meticulosa y orden.

Contenido: **Duración:** 4 horas

1.1. Clases
- 1.1.1. Definición de clase
- 1.1.2. Miembros de una clase: variables de clase, métodos
- 1.1.3. Declaración de atributos o datos y su uso
- 1.1.4. Modificadores de acceso para atributos
- 1.1.5. Protocolo de un método
- 1.1.6. Modificadores de acceso para métodos
- 1.1.7. Modificadores aplicables en la declaración de clases

1.2. Manejo de métodos
- 1.2.1. Tipos de métodos: métodos de instancia y métodos de clase
- 1.2.2. Invocación de métodos y pase de parámetros a métodos
- 1.2.3. Métodos constructores: creación de constructores, pase de parámetros a constructores
- 1.2.5. Sobrecarga de métodos

1.3. Objetos
- 1.3.1. Definición de objeto
- 1.3.2. Declaración e inicialización de objetos
- 1.4.3. Tipos de objetos

UNIDAD II. Modularidad y jerarquía

Competencia:
Aplicar conjuntos de clases en una arquitectura modular y con estructura jerárquica, con base a los principios de relaciones de herencia, composición, dependencia y asociación, para reproducir las relaciones entre varias entidades del mundo real, con creatividad y trabajo colaborativo.

Contenido: **Duración:** 4 horas

2.1 Herencia simple: clases derivadas
- 2.1.1. Herencia de los miembros de la clase padre
- 2.1.2. Agregación de comportamiento en la clase derivada
- 2.1.3. Sobre escritura de atributos y métodos
- 2.1.4. Invocación del constructor de la clase padre

2.2. Tipos de clases
- 2.2.1. Clases abstractas
- 2.2.2. Interfaces
- 2.2.3. Enumeraciones

2.3. Herencia múltiple
- 2.3.1. Herencia múltiple entre clases
- 2.3.2. Herencia múltiple entre clases e interfaces
- 2.3.3. Herencia múltiple entre interfaces
- 2.3.4. Jerarquía de herencia múltiple

2.4. Composición de clases
- 2.4.1. Composición fuerte
- 2.4.2. Composición débil
- 2.4.3. Clases anidadas

2.5. Manejo de paquetes
- 2.5.1. La clase como unidad mínima de modularidad.
- 2.5.2. Creación de un paquete
- 2.5.3. Invocación y uso de paquetes

UNIDAD III. Polimorfismo y reutilización de código

Competencia:
Aplicar el polimorfismo de objetos que faciliten la reutilización de código, empleando los principios y estrategias de diseño, para reproducir el polimorfismo y reorganizar la disposición de los elementos que representan las entidades del mundo real, con respeto a los principios de manejo de información según el lenguaje de modelado empleado y actitud colaborativa.

Contenido: **Duración:** 4 horas

3.1. Polimorfismo
- 3.1.1. Definición de polimorfismo
- 3.1.2. Sobrecarga de métodos
- 3.1.3. Sobre-escritura de miembros
- 3.1.4. Enlace dinámico

3.2. Cohesión
- 3.2.1. El concepto de cohesión de clases y entre clases
- 3.2.2. Niveles de cohesión: coincidencia, temporal, procedural, secuencial, comunicacional, informativa, funcional, asociación lógica
- 3.2.3. Convergencia de los métodos de una clase hacia una funcionalidad
- 3.2.4. Divergencia de los métodos de una clase
- 3.2.5. Convergencia de las clases de un módulo/paquete
- 3.2.6. Divergencia de las clases de un módulo/paquete
- 3.2.7. Principio de alta cohesión: Especialización/enfoque
- 3.2.8. Baja cohesión: dificultad para entendimiento, mantenimiento, reutilización

3.3. Acoplamiento
- 3.3.1. Acoplamiento entre clases
- 3.3.2. Niveles de acoplamiento (dependencia): de contenido, de acceso común, de control, de sello, de datos
- 3.3.3. Dependencia entre clases
- 3.3.4. La dependencia de datos
- 3.3.5. La dependencia de comportamiento
- 3.3.6. El principio de la separación
- 3.3.7. Principio de bajo acoplamiento: Independencia
- 3.3.8. Alto acoplamiento: dificultad para entendimiento, mantenimiento, reutilización

UNIDAD IV. Persistencia y concurrencia

Competencia:
Aplicar la persistencia y concurrencia de objetos, empleando los medios y mecanismos de lectura, escritura y encriptación, cuidando el acceso concurrente, para reproducir entidades del mundo real que perduren a lo largo del tiempo y los comportamientos que pueden superponerse en el tiempo entre varias entidades del mundo real, con respeto a los principios de manejo de información según el lenguaje de modelado empleado y actitud sistemática.

Contenido: **Duración:** 4 horas

4.1 Persistencia
- 4.1.1 Definición de persistencia
- 4.1.2 Persistencia de en memoria
- 4.1.3 Persistencia en el tiempo
- 4.1.4 Persistencia de objetos

4.2 Diseño de estructuras persistentes
- 4.2.1 Estructuras de datos persistentes
- 4.2.2 Encriptación de datos
- 4.2.3. Persistencia en medios de almacenamiento
- 4.2.4 Bases de datos orientadas a objetos

4.3 Concurrencia
- 4.3.1 Definición de concurrencia
- 4.3.2 Objetos sincrónicos versus asincrónicos
- 4.3.3 Compartición de datos
- 4.3.4 Compartición de bloques de código
- 4.3.5 Sincronización

4.4 Patrones de diseño concurrente
- 4.4.1 Aceptor-Conector
- 4.4.2 Reactor-Proactor
- 4.4.3 Objetos activos
- 4.4.4 Monitores
- 4.4.5 Líder y seguidores

VI. ESTRUCTURA DE LAS PRÁCTICAS DE TALLER				
No. de Práctica	**Competencia**	**Descripción**	**Material de Apoyo**	**Duración**
1	Construir el diagrama y el programa de computo orientado a objetos correspondiente, de una clase que represente la abstracción de un grupo de objetos, para reproducir el estado de una entidad del mundo real, de acuerdo con los principios de la orientación objetos y las capacidades del lenguaje de modelado y programación, con una actitud crítica, propositiva y creativa.	El docente: 1. Explica el concepto de abstracción, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. Explica cómo analizar un concepto, diagramar una clase que represente su abstracción en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir el estado de una entidad del mundo real, a través de una clase y sus atributos. 3. Proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. El alumno: 4. Lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. Analiza un concepto, diagrama una clase que represente su abstracción en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir el estado de una entidad del mundo real, a través de una clase y sus atributos. 7. Presenta el resultado del ejercicio	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		al profesor. El docente valida que el estudiante haya analizado un concepto, diagramado una clase que represente su abstracción en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir el estado de una entidad del mundo real, a través de una clase y sus atributos.		
2	Construir un diagrama de clases, y el programa de computo orientado a objetos correspondiente, con las propiedades que representen la inicialización del estado de los objetos a través de sus métodos constructores, para reproducir el comportamiento de una entidad del mundo real, de acuerdo con los principios de la orientación objetos y las capacidades del lenguaje de modelado y programación, con una actitud crítica, propositiva y creativa.	El docente: 1. Explica el concepto de encapsulamiento, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. Explica cómo analizar un concepto, diagramar las propiedades de una clase que representen la inicialización del estado de un objeto en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos constructores de una clase. 3. Proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. El alumno: 4. Lee la descripción del ejercicio para entender el problema a tratar y	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		los requerimientos específicos. 5. Analiza un concepto, diagrama las propiedades de una clase que representen la inicialización del estado de un objeto en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos constructores de una clase. 6. Presenta el resultado del ejercicio al profesor. El docente valida que el estudiante haya analizado un concepto, diagramado las propiedades de una clase que representen la inicialización del estado de un objeto en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos constructores de una clase.		
3	Construir un modelo orientado a objetos con su diagrama de clases y programas de cómputo correspondientes en los cuales se maneje el encapsulamiento de información, para reproducir el comportamiento de una entidad del mundo real, de acuerdo con los principios de la orientación objetos y las capacidades del lenguaje de modelado y programación, con una actitud crítica, propositiva y creativa.	1. El docente explica el concepto de encapsulamiento, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar un concepto, diagramar las propiedades de una clase que representen el encapsulamiento de información en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir el	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		comportamiento de una entidad del mundo real, a través de métodos de la interfaz de un objeto. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza un concepto, diagrama las propiedades de una clase que representen el encapsulamiento de información en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos de la interfaz de un objeto. 6. El alumno presenta el resultado del ejercicio al profesor. 7. El profesor valida que el estudiante haya analizado un concepto, diagramado las propiedades de una clase que representen el encapsulamiento de información en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos de la interfaz de un objeto.		
4	Construir un modelo orientado a objetos con su diagrama de clases y programas de cómputo	1. El docente explica el concepto de encapsulamiento, su representación en un lenguaje de modelado y su	Material: Apuntes del curso, literatura a consultar	2 horas

	correspondientes en los cuales se maneje el encapsulamiento de información, para reproducir el comportamiento de una entidad del mundo real, respetando los principios de la orientación objetos y las capacidades del lenguaje de modelado y programación, con una actitud crítica, propositiva y creativa.	implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar un concepto, diagramar las propiedades de una clase que representen el encapsulamiento de información en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos de la interfaz de una clase. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza un concepto, diagrama las propiedades de una clase que representen el encapsulamiento de información en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos de la interfaz de una clase. 6. El alumno presenta el resultado del ejercicio al profesor. 7. El profesor valida que el estudiante haya analizado un concepto, diagramado las propiedades de una clase que representen el encapsulamiento de	Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	

		información en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir el comportamiento de una entidad del mundo real, a través de métodos de la interfaz de una clase.		
5	Construir modelos orientados a objetos con sus diagramas de clases y programas de computo respectivos en los cuales se represente la jerarquía entre clases, para reproducir las relaciones entre varias entidades del mundo real, a través de la aplicación de los principios de *Generalización* y *Asociación* y de acuerdo con los principios de la orientación objetos y las capacidades del lenguaje de modelado y programación, con una actitud crítica, propositiva y creativa.	1. El docente explica el concepto de jerarquías entre clases derivadas a partir de una clase padre, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación entre dos conceptos, diagramar las jerarquías entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de una clase padre. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación entre dos conceptos, diagrama las jerarquías entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de una clase padre. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado la relación entre dos conceptos, diagramado las jerarquías entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de una clase padre.		
6		1. El docente explica el concepto de jerarquías entre clases parcialmente completas y completamente abstractas, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación entre dos conceptos, diagramar las jerarquías entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases parcialmente completas y completamente	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

	abstractas. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación entre dos conceptos, diagrama las jerarquías entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases parcialmente completas y completamente abstractas. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado la relación entre dos conceptos, diagramado las jerarquías entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases parcialmente completas y completamente abstractas.		
7	1. El docente explica el concepto de jerarquías entre clases derivadas a	Material: Apuntes del curso, literatura a	2 horas

	partir de varias clases padre, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación entre dos conceptos, diagramar las jerarquías entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de varias clases padre. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación entre dos conceptos, diagrama las jerarquías entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de varias clases padre. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el	consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	

		estudiante haya analizado la relación entre dos conceptos, diagramado las jerarquías entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases derivadas a partir de varias clases padre.		
8		1. El docente explica el concepto de jerarquías entre clases compuestas a partir de clases componentes, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación entre dos conceptos, diagramar las jerarquías entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases compuestas a partir de clases componentes. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación entre	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		dos conceptos, diagrama las jerarquías entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases compuestas a partir de clases componentes. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado la relación entre dos conceptos, diagramado las jerarquías entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases compuestas a partir de clases componentes.		
9		1. El docente explica el concepto de jerarquías entre clases anidadas en clases contenedores, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación entre dos conceptos, diagramar las jerarquías entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases anidadas en clases contenedores. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación entre dos conceptos, diagrama las jerarquías entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases anidadas en clases contenedores. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado la relación entre dos conceptos, diagramado las jerarquías entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir las relaciones entre varias entidades del mundo real, a través de las relaciones de *Generalización* y *Asociación* entre clases anidadas en clases contenedores.		
10	Desarrollar modelos orientados a	1. El docente explica el concepto de	Material: Apuntes del	2 horas

	objetos en los que se representa su modularidad con un diagrama estructural que represente la modularidad del modelo, a través de paquetes cohesivos de clases, para modularizar la disposición de los elementos que representan las entidades del mundo real, de acuerdo con los principios de la orientación objetos y capacidades del lenguaje de modelado y programación, y con una actitud crítica, propositiva y creatividad.	modularidad, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar los componentes de un modelo, diagramar la modularidad en un diagrama estructural, utilizar el modelo y construir un programa de cómputo orientado a objetos, para modularizar la disposición de los elementos que representan las entidades del mundo real, a través de paquetes cohesivos de clases. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza los componentes de un modelo, diagrama la modularidad en un diagrama estructural, utiliza el modelo y construye un programa de cómputo orientado a objetos, para modularizar la disposición de los elementos que representan las entidades del mundo real, a través de paquetes cohesivos de clases. 6. El alumno presenta el resultado del ejercicio al profesor. 7. El profesor valida que el estudiante haya analizado los componentes de un modelo, diagramado la modularidad en un	curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	

		diagrama estructural, utilizado el modelo y construido un programa de cómputo orientado a objetos, para modularizar la disposición de los elementos que representan las entidades del mundo real, a través de paquetes cohesivos de clases.		
11	Construir modelos orientados a objetos con sus respectivos diagramas de clases y programas de cómputo, analizando la relación polimórfica entre conceptos y de acuerdo con los principios de la orientación a objetos, para representar el polimorfismo en un conjunto de entidades a través de las relaciones entre clases, con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de polimorfismo, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar la relación polimórfica entre conceptos, diagramar una jerarquía entre clases en un diagrama de clases, utilizar el modelo y construir un programa de cómputo orientado a objetos, para reproducir el polimorfismo entre varias entidades del mundo real, a través de las relaciones entre clases. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza la relación polimórfica entre conceptos, diagrama una jerarquía entre clases en un diagrama de clases, utiliza el modelo y construye un programa de cómputo orientado a objetos, para reproducir el polimorfismo entre varias entidades	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		del mundo real, a través de las relaciones entre clases. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado la relación polimórfica entre conceptos, diagramado una jerarquía entre clases en un diagrama de clases, utilizado el modelo y construido un programa de cómputo orientado a objetos, para reproducir el polimorfismo entre varias entidades del mundo real, a través de las relaciones entre clases.		
12	Construir un modelo orientado a objetos e implementarlo en una biblioteca de clases, aplicando los principios de la cohesión en la reutilización y de acuerdo con los principios de la orientación objetos, para organizar la disposición de los elementos que representan las entidades del mundo real; con una actitud crítica, propositiva y con creatividad.	1. El docente explica los principios de la cohesión en la reutilización, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar los principios de la cohesión en la reutilización, diagramar los componentes de un modelo en un diagrama estructural, utilizar el modelo y construir una biblioteca de clases orientada a objetos, para reorganizar la disposición de los elementos que representan las entidades del mundo real. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		específicos. 5. El alumno analiza los principios de la cohesión en la reutilización, diagrama los componentes de un modelo en un diagrama estructural, utiliza el modelo y construye una biblioteca de clases orientada a objetos, para reorganizar la disposición de los elementos que representan las entidades del mundo real. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado los principios de la cohesión en la reutilización, diagramado los componentes de un modelo en un diagrama estructural, utilizado el modelo y construido una biblioteca de clases orientada a objetos, para reorganizar la disposición de los elementos que representan las entidades del mundo real.		
13	Construir un modelo orientado a objetos e implementarlo en una biblioteca de clases, aplicando los principios del acoplamiento en la reutilización y de acuerdo con los principios de la orientación objetos, para organizar la disposición de los elementos que representan las entidades del mundo real; con una actitud crítica, propositiva y con creatividad.	1. El docente explica los principios de acoplamiento en la reutilización, su representación en un lenguaje de modelado y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar los principios de acoplamiento en la reutilización, diagramar los componentes de un modelo en un diagrama estructural, utilizar el modelo y construir una biblioteca de clases orientada a objetos, para reorganizar la disposición de los	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	2 horas

		elementos que representan las entidades del mundo real, a través de bibliotecas de clases. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza los principios de acoplamiento en la reutilización, diagrama los componentes de un modelo en un diagrama estructural, utiliza el modelo y construye una biblioteca de clases orientada a objetos, para reorganizar la disposición de los elementos que representan las entidades del mundo real, a través de bibliotecas de clases. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado los principios de acoplamiento en la reutilización, diagramado los componentes de un modelo en un diagrama estructural, utilizado el modelo y construido una biblioteca de clases orientada a objetos, para reorganizar la disposición de los elementos que representan las entidades del mundo real, a través de bibliotecas de clases.		
14	Construir modelos orientados a objetos con sus respectivos diagramas de	1. El docente explica el concepto de	Material: Apuntes del curso, literatura a	3 horas

	clases y diagramas de objetos en los cuales se represente la persistencia de objetos, e implementarlos en bibliotecas de clases, utilizando las decoraciones de persistencia y de acuerdo con los principios de la orientación a objetos, para reproducir entidades del mundo real que perduren a lo largo del tiempo, con una actitud crítica, propositiva y con creatividad.	persistencia y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar elementos que perduran en el tiempo, diagramar la persistencia en diagramas de objetos y clases, utilizar el modelo y construir una biblioteca de clases orientada a objetos utilizando las decoraciones de persistencia en modelos de objetos y clases, para reproducir entidades del mundo real que perduren a lo largo del tiempo, a través de clases y objetos persistentes. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos específicos. 5. El alumno analiza elementos que perduran en el tiempo, diagrama la persistencia en diagramas de objetos y clases, utiliza el modelo y construye una biblioteca de clases orientada a objetos utilizando las decoraciones de persistencia en modelos de objetos y clases, para reproducir entidades del mundo real que perduren a lo largo del tiempo, a través de clases y objetos persistentes. 7. El alumno presenta el resultado del ejercicio al profesor.	consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	

		8. El profesor valida que el estudiante haya analizado elementos que perduran en el tiempo, diagramado la persistencia en diagramas de objetos y clases, utilizado el modelo y construido una biblioteca de clases orientada a objetos utilizando las decoraciones de persistencia en modelos de objetos y clases, para reproducir entidades del mundo real que perduren a lo largo del tiempo, a través de clases y objetos persistentes.		
15	Construir modelos orientados a objetos con sus respectivos diagramas de clases, actividades, secuencia y maquinas de estados e implementarlos en bibliotecas de clases, representado su comportamiento a través del tiempo y de acuerdo con los principios de la orientación a objetos, para crear objetos concurrentes que reproduzcan los comportamientos que pueden superponerse en el tiempo entre varias entidades del mundo real, con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de concurrencia y su implementación en un lenguaje de programación orientado a objetos. 2. El docente explica cómo analizar los comportamientos a través del tiempo, diagramar la concurrencia en diagramas de actividades, secuencia o máquina de estados, utilizar el modelo y construir una biblioteca de clases orientada a objetos, para reproducir los comportamientos que pueden superponerse en el tiempo entre varias entidades del mundo real, a través de objetos concurrentes. 3. El docente proporciona la descripción del ejercicio a realizar, precisando el problema a tratar y los requerimientos específicos. 4. El alumno lee la descripción del ejercicio para entender el problema a tratar y los requerimientos	Material: Apuntes del curso, literatura a consultar Equipo: Computadora, Conexión a Internet. Herramientas software: Lenguaje de programación, Editor de texto, Compilador.	3 horas

		específicos. 5. El alumno analiza los comportamientos a través del tiempo, diagrama la concurrencia en diagramas de actividades, secuencia o máquina de estados, utiliza el modelo y construye una biblioteca de clases orientada a objetos, para reproducir los comportamientos que pueden superponerse en el tiempo entre varias entidades del mundo real, a través de objetos concurrentes. 7. El alumno presenta el resultado del ejercicio al profesor. 8. El profesor valida que el estudiante haya analizado los comportamientos a través del tiempo, diagramado la concurrencia en diagramas de actividades, secuencia o máquina de estados, utilizado el modelo y construido una biblioteca de clases orientada a objetos, para reproducir los comportamientos que pueden superponerse en el tiempo entre varias entidades del mundo real, a través de objetos concurrentes.		

VI. ESTRUCTURA DE LAS PRÁCTICAS DE LABORATORIO				
No. de Práctica	**Competencia**	**Descripción**	**Material de Apoyo**	**Duración**
1	Diseñar clases para modelar el estado y comportamiento de entidades del mundo real, a través de variables de	1. El docente explica el concepto de abstracción. 2. El docente explica la estructura	Apuntes del curso, literatura a consultar, computadora, conexión a Internet,	2 horas

	clase y métodos, respectivamente, respetando los principios de la orientación objetos y las capacidades del lenguaje de programación, con una actitud crítica, propositiva y con creatividad.	de una clase a nivel de sus miembros. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de diseño de clases. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño de las clases a crear. 5. El alumno analiza el problema planteado, identificando las clases a diseñar. 6. El alumno diseña las clases requeridas, respetando los principios de creación de variables de clase y métodos. 7. El alumno codifica las clases requeridas, respetando los principios de creación de clases y las capacidades del lenguaje de programación orientado a objetos empleado. 8. El alumno compila y ejecuta el programa que contiene las clases diseñadas. 9. El alumno presenta el reporte de la práctica.	herramientas software, lenguaje de programación, editor de texto, compilador.	
2	Diseñar métodos constructores, para modelar la inicialización del estado de entidades del mundo real encapsulado en clases, precisando el protocolo con pase de parámetros y la asignación de valores, con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de estado de una entidad expresada en variables de clase. 2. El docente explica el concepto de método constructor. 3. El docente explica la estructura del protocolo de un método constructor.	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de diseño de constructores. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño de los constructores a crear. 6. El alumno analiza el problema planteado, identificando los constructores a diseñar. 7. El alumno diseña los constructores requeridos, respetando los principios de creación de métodos constructores. 8. El alumno codifica los métodos constructores requeridos, respetando los principios de creación de métodos y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene los constructores diseñados. 10. El alumno presenta el reporte de la práctica.		
3	Diseñar métodos de instancia, para modelar el comportamiento de entidades del mundo real con multiplicidad y con estado específico, precisando su protocolo de tal manera que puedan ser invocados a través de	1. El docente explica el contexto de entidades del mundo real con multiplicidad mayor a uno. 2. El docente explica el concepto de instancia de clase (objeto).	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

	instancias de la clase, con una actitud crítica, propositiva y con creatividad.	3. El docente explica la estructura del protocolo de un método de instancia. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de diseño de métodos de instancia. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño de de métodos de instancia. 6. El alumno analiza el problema planteado, identificando de métodos de instancia a diseñar. 7. El alumno diseña de métodos de instancia requeridos, respetando los principios de creación de métodos. 8. El alumno codifica los métodos de instancia requeridos, respetando los principios de creación de métodos y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene de métodos de instancia diseñados. 10. El alumno presenta el reporte de la práctica.		
4	Diseñar métodos de clase , para modelar el comportamiento de entidades únicas del mundo real con comportamiento general, precisando	1. El docente explica el contexto de entidades del mundo real con comportamiento general (comportamiento de	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software,	2 horas

	su protocolo de tal manera que puedan ser invocados a través de la clase, sin creación de instancias, con una actitud crítica, propositiva y con creatividad.	procesamiento). 2. El docente explica la estructura del protocolo de un método de clase. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de diseño de métodos de clase. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño de de métodos de clase. 5. El alumno analiza el problema planteado, identificando de métodos de clase a diseñar. 6. El alumno diseña los métodos de clase requeridos, respetando los principios de creación de métodos. 7. El alumno codifica los métodos de clase requeridos, respetando los principios de creación de métodos y las capacidades del lenguaje de programación orientado a objetos empleado. 8. El alumno compila y ejecuta el programa que contiene de métodos de clase diseñados. 9. El alumno presenta el reporte de la práctica.	lenguaje de programación, editor de texto, compilador.	
5	Diseñar clases derivadas, a partir de una clase padre, para expresar la abstracción en forma jerárquica de entidades del mundo real, aplicando los principios de la herencia simple y	1. El docente explica el concepto de jerarquía en su expresión de herencia simple. 2. El docente explica el concepto de clase derivada.	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación,	2 horas

	las capacidades de un lenguaje de programación orientado a objetos, con una actitud analítica, propositiva y con creatividad.	3. El docente explica la propiedades de la clase padre y las características de la clase derivada. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de clases derivadas. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de las clases derivadas. 6. El alumno analiza el problema planteado, identificando las clases padre y las clases derivadas a crear. 7. El alumno diseña las clases padre y las clases derivadas requeridas, respetando los principios de herencia simple. 8. El alumno codifica las clases requeridas, respetando los principios herencia simple y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene las clases padre y las clases derivadas. 10. El alumno presenta el reporte de la práctica.	editor de texto, compilador.	
6	Diseñar clases parcialmente completas y completamente abstractas para modelar entidades y situaciones del mundo real que tengan esas	1. El docente explica el concepto de clases abstractas e interfaces. 2. El docente explica el contexto del mundo real susceptible de	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software,	2 horas

	características genericidad del comportamiento, creando métodos abstractos y habilitándolas para hacer extensiones de las mismas y poder crear objetos; con una actitud analítica, propositiva y con creatividad.	modelar mediante métodos abstractos. 3. El docente explica la propiedades de una clase abstracta y una interface. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de clases abstractas e interfaces. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de las clases abstractas e interfaces. 6. El alumno analiza el problema planteado, identificando las clases abstractas e interfaces a crear. 7. El alumno diseña clases abstractas e interfaces requeridas, respetando los principios de diseño de métodos abstractos. 8. El alumno codifica clases abstractas e interfaces requeridas, respetando los principios de diseño de métodos abstractos y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene clases abstractas e interfaces. 10. El alumno presenta el reporte de la práctica.	lenguaje de programación, editor de texto, compilador.	
7	Diseñar clases derivadas a partir de	1. El docente explica el concepto	Apuntes del curso, literatura	2 horas

	varias clases padre, para expresar la abstracción en forma jerárquica y multiplicidad horizontal de entidades del mundo real, aplicando los principios de la herencia múltiple y las capacidades de un lenguaje de programación orientado a objetos para tal fin; con una actitud analítica, propositiva y con creatividad.	de jerarquía en su expresión de herencia múltiple. 2. El docente explica el concepto de clase derivada con múltiples padres. 3. El docente explica la propiedades de las clases padre y las características de la clase derivada. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de clases derivadas a partir de múltiples clases padre. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de las clases derivadas con múltiples clases padre. 6. El alumno analiza el problema planteado, identificando las clases padre y las clases derivadas a crear. 7. El alumno diseña las clases padre y las clases derivadas requeridas, respetando los principios de herencia múltiple. 8. El alumno codifica las clases requeridas, respetando los principios herencia múltiple y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene las clases	a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	

		padre y las clases derivadas. 10. El alumno presenta el reporte de la práctica.		
8	Diseñar clases compuestas a partir de clases componentes, para modelar entidades del mundo real con estructura jerárquica de composición, mediante una abstracción adecuada e identificando la asociación fuerte y asociación débil y la multiplicidad; con una actitud crítica, propositiva y creatividad.	1. El docente explica el concepto de jerarquía de composición. 2. El docente explica el concepto de clase compuesta y clase componente. 3. El docente explica la propiedades de una clase compuesta y las características de las clases componentes. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de clases compuestas a partir de clases componentes. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de las clases compuestas a partir de clases componentes. 6. El alumno analiza el problema planteado, identificando las clases compuestas y clases componente a crear. 7. El alumno diseña la clase compuesta y clases componente requeridas, respetando los principios de jerarquía de composición y multiplicidad. 8. El alumno codifica clase compuesta y clases componente requeridas, respetando los principios de jerarquía de	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		composición y multiplicidad y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene clase compuesta y clases componentes y la expresión de multiplicidad. 10. El alumno presenta el reporte de la práctica.		
9	Diseñar clases anidadas en clases contenedores, para modelar entidades del mundo real con una estructura de composición por entidades internas funcionales con uso local a su contexto, enfatizando el encapsulamiento y delimitando las capacidades, responsabilidades del contenedor y de los componentes y sus dependencias; con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de composición con anidamiento. 2. El docente explica el concepto de clase anidada. 3. El docente explica la propiedades de una clase anidada y sus diferentes expresiones. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de clases anidadas. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de las clases anidadas. 6. El alumno analiza el problema planteado, identificando las clases anidadas a crear. 7. El alumno diseña la clase contenedor y las clases anidadas requeridas, respetando los principios de encapsulamiento. 8. El alumno codifica clase contenedor y clases anidadas	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		requeridas, respetando los principios de jerarquía de encapsulamiento y ámbito de las clases anidadas, así como las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene clase contenedor y clases anidadas. 10. El alumno presenta el reporte de la práctica.		
10	Crear paquetes cohesivos para organizar conjuntos de clases agrupadas en base a su naturaleza y convergencia en la solución de un problema, aplicando los principios de asociación de clases y las directivas de creación de paquetes; con una actitud crítica, reflexiva y con creatividad.	1. El docente explica el concepto de paquete de clases.. 2. El docente explica la estructura de un paquete y su expresión en almacenamiento.. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la creación de paquetes. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño de paquetes. 5. El alumno analiza el problema planteado, identificando paquetes a diseñar. 6. El alumno diseña los paquetes requeridos, respetando los principios de de agrupación de clases. 7. El alumno codifica las clases requeridas y genera los paquetes	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		requeridos, respetando los principios de agrupación de clases y las capacidades del lenguaje de programación orientado a objetos empleado. 8. El alumno compila y ejecuta el programa que contiene las clases y las directivas de creación de paquetes. 9. El alumno presenta el reporte de la práctica.		
11	Diseñar objetos polimórficos, con el fin de potenciar las capacidades de las entidades modeladas, mediante la creación de clases con sobrecarga y sobreescritura de métodos, respetando los principios de la herencia entre clases; con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de polimorfismo. 2. El docente explica las diferentes expresiones de polimorfismo. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de las formas de polimorfismo a implementar. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de diseño objetos polimórficos. 5. El alumno analiza el problema planteado, identificando las formas del polimorfismo a implementar. 6. El alumno diseña el polimorfismo requerido, respetando las reglas de expresión de polimorfismo. 7. El alumno codifica las clases requeridas con el polimorfismo implementado, respetando las	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		reglas de expresión de polimorfismo y las capacidades del lenguaje de programación orientado a objetos empleado. 8. El alumno compila y ejecuta el programa que contiene el polimorfismo implementado. 9. El alumno presenta el reporte de la práctica.		
12	Aplicar los principios de la cohesión, para lograr estructuras de clases y arquitecturas con alta convergencia al problema a modelar, mediante el empleo de las capacidades para tal fin de un lenguaje de modelado y programación orientado a objetos; con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto de cohesión. 2. El docente explica los principios y reglas para lograr la cohesión. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos de la cohesión a lograr. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos de la cohesión a lograr. 5. El alumno analiza el problema planteado, identificando las formas de la cohesión a lograr. 6. El alumno diseña la cohesión requerida, respetando las reglas y principios correspondientes. 7. El alumno codifica las clases requeridas con la cohesión implementada, respetando las reglas y principios correspondientes y las capacidades del lenguaje de programación orientado a objetos	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		empleado. 8. El alumno compila y ejecuta el programa que contiene las clases con la alta cohesión implementada. 9. El alumno presenta el reporte de la práctica.		
13	Aplicar los principios de acoplamiento, para lograr estructuras de clases y arquitecturas con bajo nivel de dependencias, mediante el empleo de las capacidades para tal fin de un lenguaje de modelado y programación orientado a objetos; con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto del acoplamiento. 2. El docente explica los principios y reglas para lograr el bajo acoplamiento. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos del acoplamiento a lograr. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos del acoplamiento a lograr. 5. El alumno analiza el problema planteado, identificando las formas del acoplamiento a lograr. 6. El alumno diseña el acoplamiento requerido, respetando las reglas y principios correspondientes. 7. El alumno codifica las clases requeridas con el acoplamiento implementado, respetando las reglas y principios correspondientes y las capacidades del lenguaje de programación orientado a objetos empleado.	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador..	2 horas

		8. El alumno compila y ejecuta el programa que contiene las clases con bajo acoplamiento implementado. 9. El alumno presenta el reporte de la práctica.		
14	Diseñar objetos persistentes para satisfacer las necesidades de calidad de datos logrando la consistencia e integridad de los mismos a través del tiempo y del espacio, mediante el uso de estructuras de datos apropiadas y mecanismos de encriptación; con una actitud crítica, propositiva y con creatividad.	1. El docente explica el concepto persistencia. 2. El docente explica los atributos de calidad de los datos, tales como consistencia, integridad, etc., los cuales deben preservarse a través del tiempo y del espacio. 3. El docente explica los principios y reglas para lograr la persistencia de objetos y las directivas básicas para su implementación. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos para la persistencia a lograr. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos para la persistencia a lograr. 6. El alumno analiza el problema planteado, identificando las características de la persistencia a lograr. 7. El alumno diseña la persistencia requerida, respetando las reglas y principios	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		correspondientes. 8. El alumno codifica las clases requerida para implementar la resistencia deseada, respetando las reglas y principios correspondientes y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene las clases para lograr la resistencia deseada. 10. El alumno presenta el reporte de la práctica.		
15	Diseñar objetos concurrentes, para modelar situaciones del mundo real de acceso paralelo a datos y bloques de código, mediante el uso de directivas básicas y técnicas de comunicación síncrona y asíncrona, y haciendo uso de las capacidades del lenguaje modelado y de programación; con una actitud crítica, propositiva y creatividad.	1. El docente explica el concepto concurrencia. 2. El docente explica lo que es el acceso concurrente a datos y a bloques de código, y la comunicación síncrona y asíncrona. 3. El docente explica los principios y reglas para control la concurrencia y las directivas básicas para su implementación. 4. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos del control de concurrencia a lograr. 5. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos específicos del control de concurrencia a lograr. 6. El alumno analiza el problema	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		planteado, identificando las características del control de concurrencia a lograr. 7. El alumno diseña el control de concurrencia requerido, respetando las reglas y principios correspondientes. 8. El alumno codifica las clases requeridas con el control de concurrencia implementado, respetando las reglas y principios correspondientes y las capacidades del lenguaje de programación orientado a objetos empleado. 9. El alumno compila y ejecuta el programa que contiene las clases con el control de concurrencia implementado. 10. El alumno presenta el reporte de la práctica.		
16	Diseñar objetos concurrentes, para modelar situaciones del mundo real de acceso paralelo a datos y bloques de código, mediante el uso de patrones específicos y técnicas de comunicación síncrona y asíncrona, y haciendo uso de las capacidades del lenguaje modelado y de programación; con una actitud crítica, propositiva y creatividad.	1. El docente explica los patrones de diseño para el control de concurrencia. 2. El docente explica los principios y reglas de los patrones de diseño específicos para el control de concurrencia. 3. El docente proporciona la descripción de la práctica a realizar, precisando el problema a tratar y los requerimientos específicos del control de concurrencia a lograr. 4. El alumno lee la descripción de la práctica para entender el problema a tratar y los requerimientos para aplicar los	Apuntes del curso, literatura a consultar, computadora, conexión a Internet, herramientas software, lenguaje de programación, editor de texto, compilador.	2 horas

		patrones de diseño específicos para el control de concurrencia. 5. El alumno analiza el problema planteado, identificando las características de los patrones de diseño para implementar el control de concurrencia. 6. El alumno diseña el del control de concurrencia requerido mediante patrones específicos seleccionados acorde al contexto planteado, respetando las reglas y principios correspondientes. 7. El alumno codifica las clases requeridas empleando los patrones específicos para el control de concurrencia implementado, respetando las reglas y principios correspondientes y las capacidades del lenguaje de programación orientado a objetos empleado. 8. El alumno compila y ejecuta el programa que contiene las clases implementadas usando patrones de diseño específico para lograr el control de concurrencia requerido. 10. El alumno presenta el reporte de la práctica.		

VII. MÉTODO DE TRABAJO

Encuadre: El primer día de clase el docente debe establecer la forma de trabajo, criterios de evaluación, calidad de los trabajos académicos, derechos y obligaciones docente-alumno.

Estrategia de enseñanza (docente)

- Exposición de temas y conceptos mediante explicaciones y presentaciones por medios electrónicos
- Demostraciones de prácticas de códigos de programas
- Demostraciones de planeación y desarrollo de un proyecto de tamaño mediano
- Coordinar y supervisar las prácticas tanto de taller como de laboratorio
- Elaborar y aplicar evaluaciones parciales.
- Facilitar material bibliográfico introductorio para la comprensión de conceptos y el cuerpo de conocimiento actual de la programación orientada a objetos
- Promover el trabajo colaborativo, pensamiento crítico y reflexivo en el estudiante

Estrategia de aprendizaje (alumno)

- Participación para mostrar que ha estudiado los temas contenidos en cada unidad, realización de prácticas en el laboratorio de cómputo mediante las cuales se pueda
- Participar, fortalecer y afianzar el conocimiento, trabajando en equipo y usando computadoras personales y herramientas que permitan el modelado, creación, compilación y ejecución de programas orientados a objetos, pr
- Presentación de entregables relacionados con prácticas realizadas y proyecto, los cuales permitan visualizar claramente las soluciones dadas a los problemas planteados
- Resolver evaluaciones parciales propuestas por el docente

VIII. CRITERIOS DE EVALUACIÓN

La evaluación será llevada a cabo de forma permanente durante el desarrollo de la unidad de aprendizaje de la siguiente manera:

Criterios de acreditación

- Para tener derecho a examen ordinario y extraordinario, el estudiante debe cumplir con los porcentajes de asistencia que establece el Estatuto Escolar vigente.
- Calificación en escala del 0 al 100, con un mínimo aprobatorio de 60.

Criterios de evaluación

- Participación……..……10%
- Evaluaciones parciales……40%
- Realización de prácticas formuladas y entregables sobre tareas específicas ….……20%
- Evidencia de desempeño 1 ……15%
  (Entrega de un proyecto: aplicación de tamaño mediano)
- Evidencia de desempeño 2……..15%
  (Reporte técnico)

Total……100%

IX. REFERENCIAS

Básicas

Alagic, S. (2015). *Object-Oriented Technology*. Switzerland: Springer.

Booch, G., Maksimchuk, R.A., Engle, M.W., Young, B.J, Conallen, J., y Houston, K. (2007). *Object-Oriented Analysis and Design with Applications* (3ª ed.). *(1)*11.Estados Unidos: Addison Wesley. [clásica]

Dathan, B., Ramnath, S. (2015). *Object-Oriented Analysis, Design and Implementation: An Integrated Approach*. (2ª ed.). Springer.

Jeya-Mala, D. Geetha, S. (2013). *Object Oriented Analysis and Design Using UML*. Estados Unidos: McGraw-Hill Education. [clásica]

McLaughlin, B.D., Pollice, G., y West, D. (2008). *Head First Object-Oriented Analysis & Design*. Recuperado de https://hal.archives-ouvertes.fr/hal-01484481/document

Metz, S. (2018). *Practical Object-Oriented Design: An Agile Primer Using Ruby.* Boston, USA: Addison-Wesley Professional.

Minutillo, J., y Damonte, B. (2016). *Head first Design Patterns Archit*. *(204)*7. 114-118.

Weisfeld, M. (2019). *The Object-Oriented Thought Process (5ª ed.).* (Developer's Library). Addison-Wesley Professional.

Complementarias

Baesens, B. (2015). *Beginning Java Programming: The Object-Oriented Approach.* (1ª ed.). Wrox.

Brun, A. (2019). *Python Programming: A Step By Step Guide From Beginner To Expert (Beginner, Intermediate & Advanced).* Kindle Edition. Brun, A. (Ed.). Amazon Digital Services LLC.

Clark, D. (Marzo, 2013). *Beginning C# Object-Oriented Programming (Expert's Voice in .NET)* (2ª ed.). Apress. [clásica]

Deitel, P., y Deitel, H. (2012). *Java For Programmers* (2ª ed.). [clásica]

Dusty, P. (2015*). Python 3 Object Oriented Programming*. (2ª ed.). Packt Publishing Ltd.

Phillips, D. (2015). *Python 3 Object-oriented Programming*. (2ª ed.) United Kingdom: Packt Publishing. ISBN: 9781784398781.

Schildt, H. (2017). *Java: A Beginner's Guide.* (7ª ed.). McGraw-Hill Education.

Schildt, H. (2018) *Java: The Complete Reference*. (10ª ed). McGraw-Hill Education.

Stefanov, S., Kuman-Chetan, S. (2013). Object-Oriented JavaScript (2ª ed.). Packt Publishing Limited. [clásica]

X. PERFIL DEL DOCENTE

El docente que imparta la asignatura debe contar con título de Licenciatura con una Ingeniería en un área a fin a la ciencia de la computación. Preferentemente con posgrado (maestría y doctorado) en un área a fin. Poseer experiencia docente y/o experiencia práctica en el campo disciplinar de la asignatura. Contar liderazgo en el campo disciplinar y capacidad de abstracción, capacidad de análisis y diseño, comunicación y ser innovador.