



Universidad Tecnológica de Durango Tecnologías de la Información Programación Orientada a Objetos

Actividades

"Evidencias de Actividades y Tareas"

Alumnos:

• Lomas Corral Edson

3°B BIS

Docente:

• Ing. Dagoberto Fiscal Gurrola, M.T.I.

Octubre 2025



Ilustración 1:Introducción a los conceptos del paradigma orientado a objetos	3
Ilustración 2: Concepto de casos de uso, elementos de caso de uso	4
Ilustración 3: Modelado UML.	5
Ilustración 4:Patrones básicos de diseño (singleton, factory, observer)	6



Es un estilo de programaión que organiza el código en claser y objetos cimitando cómo funciona el mundo rent. (ada objeto tiene atributos (características) y métodos (acciones que puedo hicer)
Es un estilo de programición que organiza el código en clases y objetos cimitando cómo funciona el mundo cent. Cida objeto tiene
Es un estilo de programación que organiza el código en clases y objetos cimitando cómo funciona el mundo cente Cida objeto tiene
código en clases y objetos cimitando cómo funciona el mundo cente Cida objeto tiene
codigo en clises y objetos imitado cómo funcione el mundo cente Cida objeto tiene
funcione el mundo cent. Cida objeto tiene
atitute (and training the control of the
atchita (carlot existing) V Matala
MATIONA OS CAMACACIONA (CUS) Y INCADIOS
(Aciones avenuedo haces)
activities after position in the second
lase: Molde o plano
lase: Molde o plano. Objeto: Un ejemplar en conveto.
DI DI
Atributas Propienides.
Metodos Acciones
a second and and and and and and and and and a
111 11111111111111111111111111111111111
Abstración: Ocultar detalles internos
v mostru solo la necesuia.
Enciportamiento. Proteger los datos
Lycholy Mileno. 1 10x ger 103 mass
utilizando private publica etce
Herenous Une clise over heretur
TE GIGAL VIII CIDE
de otras.
Polimorfismo. Un mismo metodo se
compacts different de meido
L'ilizando private public, etc. Herenoja. Una clise puede heredar de otras. Polimorfismo: Un mismo método se comporta diferent de accerdo d
objeto.
tarrant -
1 1 1
Constructor Método especiel que se ejecula al cierre un dejeto.
Método especial que se ejecuta al
a shirth
Jell Villejen.
En pythoninit
- Cyanon -

Ilustración 1:Introducción a los conceptos del paradigma orientado a objetos.

En esta evidencia se presentan los conceptos fundamentales del paradigma de programación orientado a objetos (POO), explicando cómo este modelo organiza el código en clases y objetos dotados de atributos y métodos. Además, se definen sus pilares clave como la abstracción, encapsulamiento, herencia y polimorfismo, principios que en conjunto permiten dividir un programa en componentes lógicos y reutilizables, mejorando así la claridad, organización y mantenimiento del software.



Casos de Uso.	Madalada DMI
Un caso de vso es una forma qué huce un sistema des	de converentur
avé hice un sistema des	de el punto
devista de quien la usa	Modelander
Sive para entender que	se espein que
haga el sistema.	The Manual Control of the Control of
Actor: Es la persons la objetivo. Es le que el	Se von granter
Actor: Es la peisons la	sistem externo
que vsi el sistema.	The state of the s
Objetivo. Ls 10 que es	actor quere
Fly a de au des Paras de	s si que a para
Fly; de eventos: Pasos que alcunzas ese objetivos	esesiges pur
MILNIENS ESE OF EXIVES	Direct amount for
Diagrams de cusa de uso	
Es un dianiama ava muestr	a losactores
Es un diagrama que muestr y los casos de uso ent Actores: iconos con torm	some de dibajo
- Actores: iconos con form	1 de persona.
	THE RESIDENCE OF THE PARTY OF T
· Cisos de uso: Ovidos con	los nombres de
115 Acciones	Mayen de com
There is a form	12 stephendrenk
Fleches: Muertin que sos	for puticipa
en que caso	The street or along
THE STREET STREE	TOTAL PROPERTY OF THE PARTY OF
pour entender como se de programarlo.	del sistema
prus entender como se	comporta untos
de programarlo.	The Agency A C Stan
100	Minister de catal

Ilustración 2: Concepto de casos de uso, elementos de caso de uso

Esta evidencia define el concepto de casos de uso y sus componentes principales: el actor, el objetivo y el flujo de eventos. Además, explica cómo estos elementos se usan en un diagrama para representar la funcionalidad de un sistema desde la perspectiva de quien lo utiliza.



Modelado UMI	7
on this de use of our forms de length	
UML (Unified Modeling Language) engusje de Modelado Unificado) es un l'enquage visus para disense y describir sistemes de	
Modelado Unitiado es un lenguaje visus	
para diserve y describir sistemes de	4
Se usa pour represent u como funcions	W.
Se usa pour represent u como tuncions	1
un sistema wites de programado, as m	10
1. AALIMI - NUA MUESTIAN	1
· Qué hace el sistem.	(1)
Como se olganiza	
Cómo se organiza Cómo interaction sus partes. UML se divide en l'ajandes tipos.	1
Of It sent vite en & ginnes ripos	MAN I
	(7)
Vescilben 11 prite = statica les sistem.	Y.
Describen la prote estation del sistem. cómo: está organizado internamente. Diagrama de clases: muestra clases, atriba	1
Player declases. 1.10-5/12 (125es) Miller	1 PS
· Diagrama de objetos: muestra instancias	1
Viajinina de objetos. Moestra Misa Mars	ne
(1/13es	
Viagram de componentes de provetes de	
despliegue.	7
Diagramis de compostamiento	100
Vescriber como se comporta el siste	oppe
Diagramis de compostamiento Describes como se comporta el siste con el tiempo: flujos, interaciones camb	105
"Vigura de cisos-	26
· Viagrama de secrescia.	120
Vi agrum de actividades.	200
· Viagrama de estados.	
	THE PARTY IS

Ilustración 3: Modelado UML.

Esta evidencia introduce el Modelado UML (Lenguaje Unificado de Modelado), presentándolo como un lenguaje visual para diseñar y describir sistemas de software. Se explica su división en dos tipos principales de diagramas: los estructurales, que definen la organización estática del sistema (como el diagrama de clases), y los de comportamiento, que muestran su dinámica e interacciones(como el diagrama de secuencia).



Actividad 4

Patrones de diseño.
Son soluciones reutilizables a un problema que se presenta en el lesaciollo de
que se presenta en el lesarrollo de
ent + will-
Los pationes de diseño no son código
sino concentos avese pueden aplicar
Exister 3 tiposcommes.
· Ciercionses Secentran en la lierción
Le objetos, ayadundo a instancia clases
eticient-mente.
· Estructurales: Definer la monera de
DIGANIZAN Y estructuras clases Y
all fac
De comportamiento: Se entres en como
los objetos interestran entresi.
*Simple ton: GNINTIZM que una clise tenja
uni soli instruci y piovaci un punto
de acceso globa.
* Fictory: El portion today es un petrón
ciercional que proporcione une intertre proc
ciels of etos very being the
clises deciden que tipo ac objeto eser
+ No to pating of comportance
au n-mit- notition 1 0,105 00, care
(105 observatores) counto su estado
anbia
Carry a de compresente . Reservation de la constante de la con

Ilustración 4:Patrones básicos de diseño (singleton, factory, observer)

Esta evidencia introduce el concepto de patrones de diseño como soluciones reutilizables a problemas comunes en el desarrollo de software. Se presenta su clasificación en tres tipos (creacionales, estructurales y de comportamiento) y se explican tres patrones básicos: Singleton, Factory y Observer, detallando la finalidad de cada uno.



A lo largo del desarrollo de estas actividades, se ha fortalecido significativamente la comprensión de los fundamentos del paradigma orientado a objetos. Cada tema abordado ha contribuido a construir una base sólida, no solo teórica sino también práctica, para modelar y resolver problemas complejos de manera modular y escalable.

El estudio de conceptos como clases, objetos, herencia y polimorfismo permitió entender cómo estructurar el software a imagen de problemas del mundo real. Dominar el encapsulamiento y el instanciamiento de objetos ha sido fundamental para crear componentes de software robustos, reutilizables y fáciles de mantener, promoviendo un código limpio y bien organizado.

La introducción a los casos de uso y al modelado UML aportó una perspectiva crucial sobre la fase de análisis y diseño. Comprender cómo definir actores, objetivos y flujos de eventos, y cómo representar visualmente la arquitectura del sistema, ha facilitado la planificación y la comunicación de las soluciones antes de escribir una sola línea de código.

El análisis de patrones de diseño básicos como Singleton, Factory y Observer brindó una visión estratégica para resolver problemas recurrentes de manera elegante y eficiente. Comprender estas plantillas probadas ha sido clave para fortalecer el pensamiento de diseño y construir software más flexible, adaptable y robusto.

Finalmente, integrar todos estos conocimientos ha demostrado la importancia de seguir un proceso metódico que abarca desde el análisis hasta la implementación. Con esta preparación, se está en condiciones de avanzar hacia el desarrollo de sistemas de software complejos, garantizando soluciones tecnológicas de mayor calidad y valor.