Universidad Tecnológica de Durango

Tecnologías de la Información

Programación Orientada a objetos

Actividades

“Evidencias de Activiades y Tareas”

Alumnos:

- Barraza Torres Jesús Daniel

3°A BIS

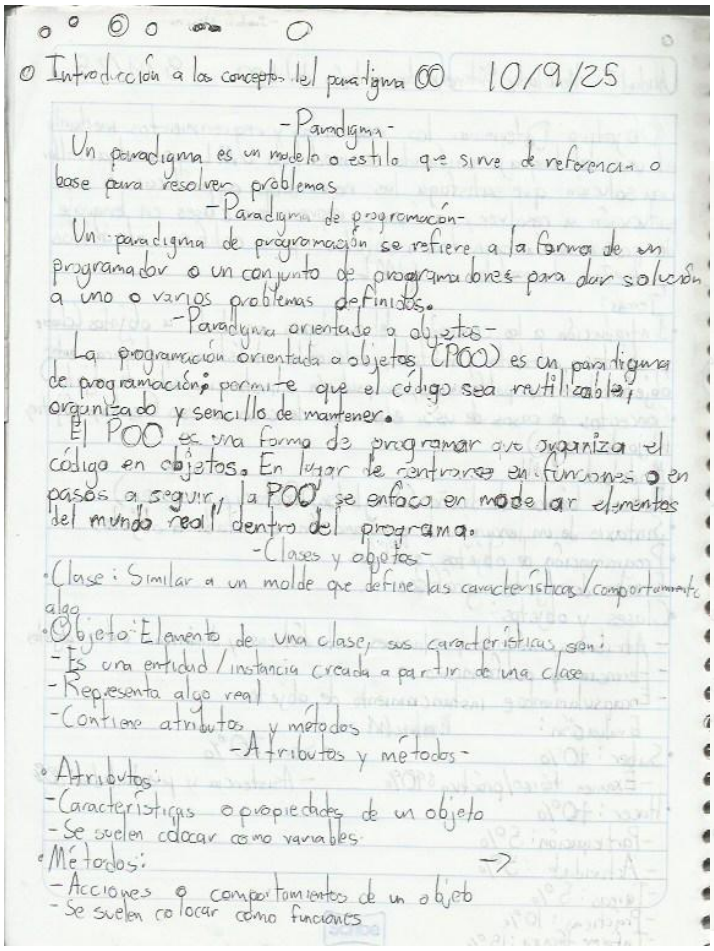
Docente:

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

Septiembre 2025

Tabla de Ilustraciones

Actividad 1



Introducción a los conceptos del paradigma OO 10/9/25

-Paradigma-

Un paradigma es un modelo o estilo que sirve de referencia o base para resolver problemas

-Paradigma de programación-

Un paradigma de programación se refiere a la forma de un programador o un conjunto de programadores para dar solución a uno o varios problemas definidos.

-Paradigma orientado a objetos-

La programación orientada a objetos (POO) es un paradigma de programación, permite que el código sea reutilizable, organizado y sencillo de mantener.

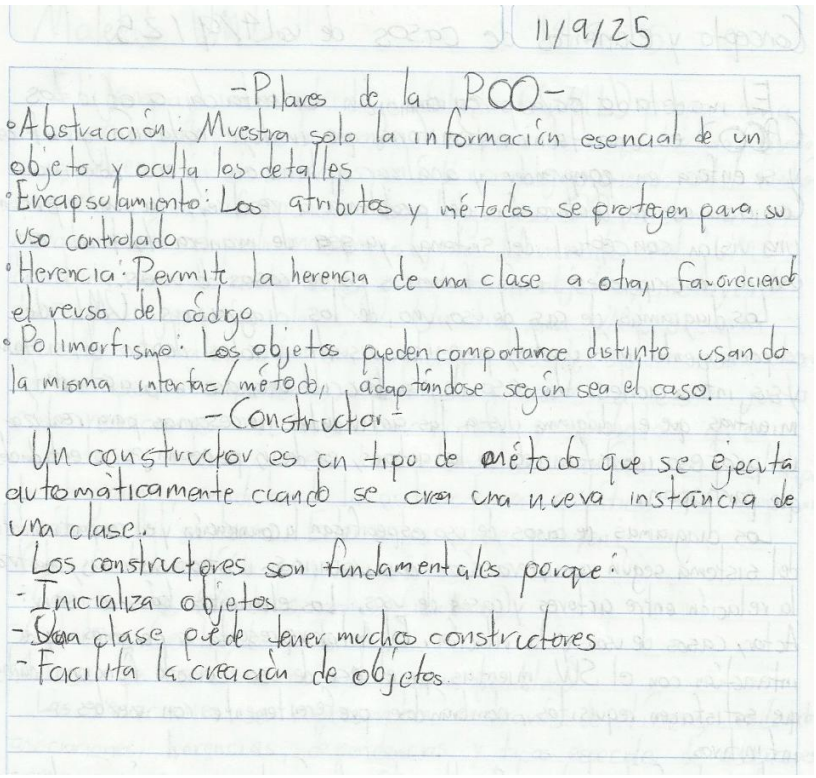
El POO es una forma de programar que organiza el código en objetos. En lugar de centrarse en funciones o en pasos a seguir, la POO se enfoca en modelar elementos del mundo real dentro del programa.

-Clases y objetos-

- Clase: Similar a un molde que define las características/comportamiento algo
- Objeto: Elemento de una clase, sus características son:
  - Es una entidad / instancia creada a partir de una clase
  - Representa algo real
  - Contiene atributos y métodos

-Atributos y métodos-

- Atributos:
  - Características o propiedades de un objeto
  - Se suelen colocar como variables
- Métodos: ->
  - Acciones o comportamientos de un objeto
  - Se suelen colocar como funciones



11/9/25

-Pilares de la POO-

- Abstracción: Muestra solo la información esencial de un objeto y oculta los detalles
- Encapsulamiento: Los atributos y métodos se protegen para su uso controlado
- Herencia: Permite la herencia de una clase a otra, favoreciendo el reuso del código
- Polimorfismo: Los objetos pueden comportarse distinto usando la misma interfaz/método, adaptándose según sea el caso.

-Constructor-

Un constructor es un tipo de método que se ejecuta automáticamente cuando se crea una nueva instancia de una clase.

Los constructores son fundamentales porque:

- Inicializa objetos
- Una clase puede tener muchos constructores
- Facilita la creación de objetos

Ilustración 1. Introducción a los conceptos del paradigma OO

En esta ilustración podemos observar el significado de paradigma dentro del contexto de la programación, junto con el significado del paradigma de la programación orientada a objetos. También se puede observar sus características y otras cosas que utiliza.

Actividad 2

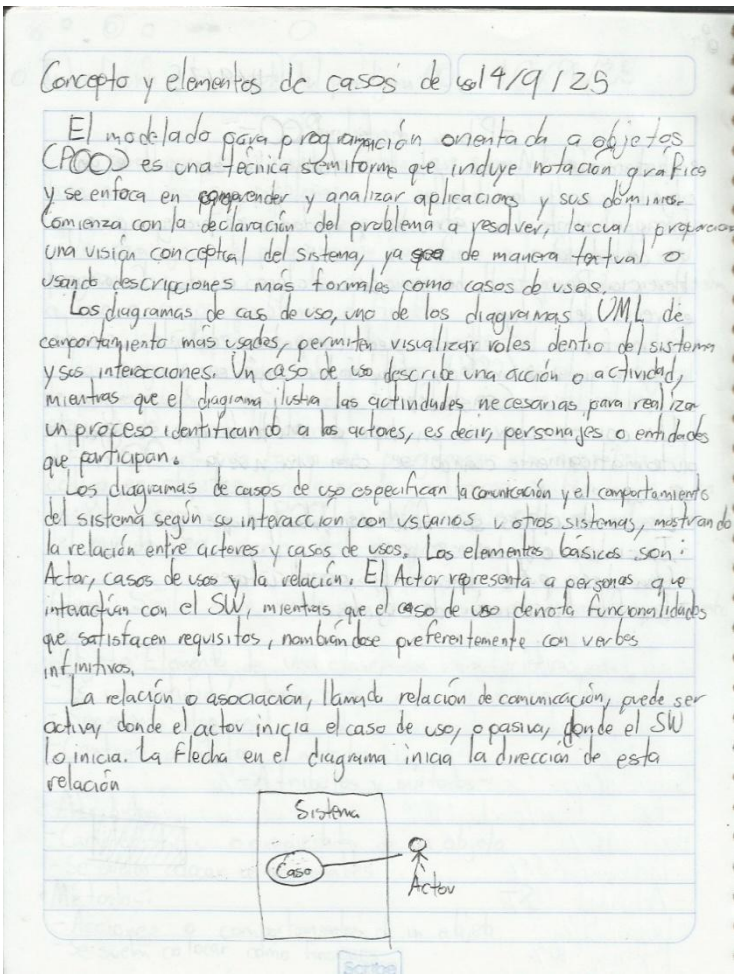


Ilustración 2. Concepto y elementos de casos de uso

En esta ilustración podemos observar los conceptos y los elementos de los casos de uso por los cuales uno puede directamente diagramar todo lo relacionado a la POO. Esto supone una guía directa sobre los diagramas posteriores.

Actividad 3

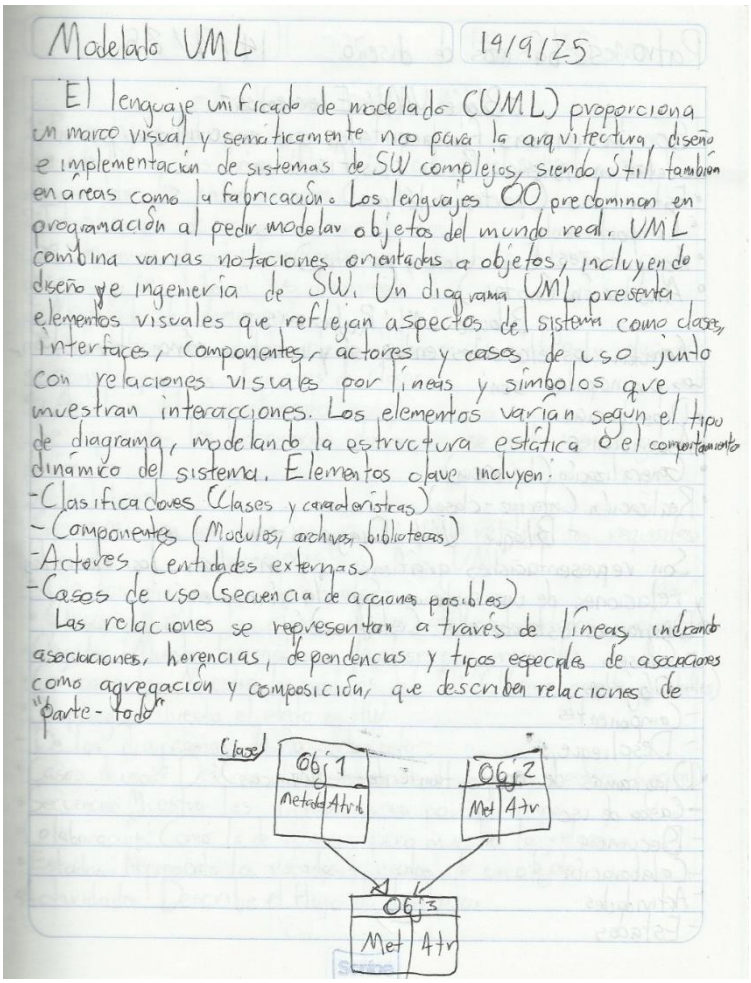
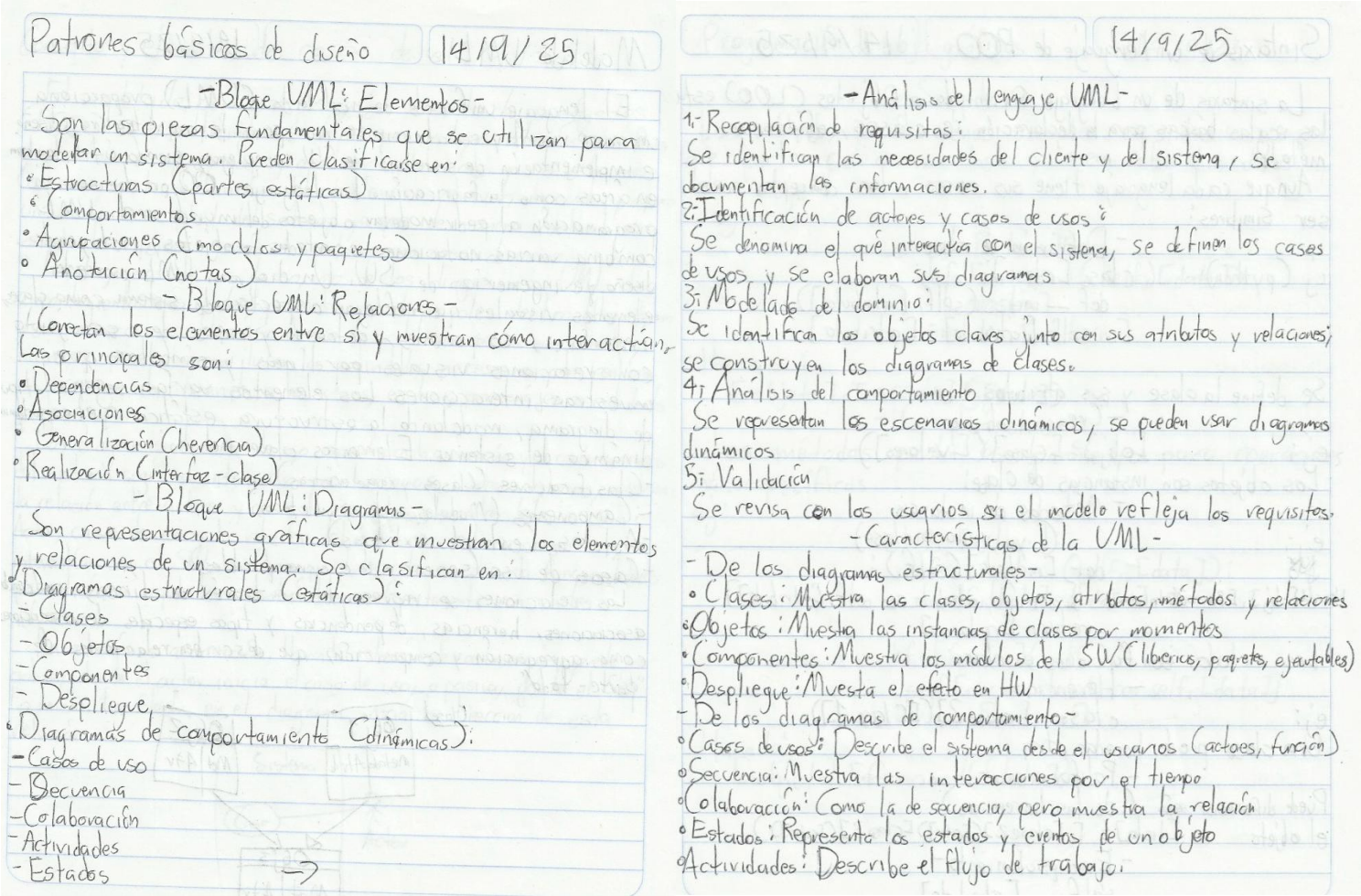


Ilustración 3. Modelado UML

En esta ilustración podemos observar el modelado de lenguaje unificado (UML), junto con sus características básicas que demuestran las reglas básicas de todos los diagramas. A pesar de eso, muchos diagramas son distintos entre si en término de funciones.

Actividad 4



Patrones básicos de diseño 14/9/25

-Bloque UML: Elementos-
Son las piezas fundamentales que se utilizan para modelar un sistema. Pueden clasificarse en:
- Estructuras (partes estáticas)
- Comportamientos
- Agrupaciones (modulos y paquetes)
- Anotación (notas)

-Bloque UML: Relaciones-
Conectan los elementos entre sí y muestran cómo interactúan. Los principales son:
- Dependencias
- Asociaciones
- Generalización (Herencia)
- Realización (Interfaz-clase)

-Bloque UML: Diagramas-
Son representaciones gráficas que muestran los elementos y relaciones de un sistema. Se clasifican en:
- Diagramas estructurales (estáticas):
  - Clases
  - Objetos
  - Componentes
  - Despliegue
- Diagramas de comportamiento (dinámicas):
  - Casos de uso
  - Secuencia
  - Colaboración
  - Actividades
  - Estados

14/9/25

-Análisis del lenguaje UML-
1- Recopilación de requisitos:
Se identifican las necesidades del cliente y del sistema, se documentan las informaciones.
2: Identificación de actores y casos de usos:
Se denomina el qué interactúa con el sistema, se definen los casos de usos y se elaboran sus diagramas
3: Modelado del dominio:
Se identifican los objetos claves junto con sus atributos y relaciones; se construyen los diagramas de clases.
4: Análisis del comportamiento
Se representan los escenarios dinámicos, se pueden usar diagramas dinámicos
5: Validación
Se revisa con los usuarios si el modelo refleja los requisitos.

-Características de la UML-
- De los diagramas estructurales-
  - Clases: Muestra las clases, objetos, atributos, métodos y relaciones
  - Objetos: Muestra las instancias de clases por momentos
  - Componentes: Muestra los módulos del SW (libreria, paquete, ejecutables)
  - Despliegue: Muestra el efecto en HW
- De los diagramas de comportamiento-
  - Casos de usos: Describe el sistema desde el usuarios (actores, función)
  - Secuencia: Muestra las interacciones por el tiempo
  - Colaboración: Como la de secuencia, pero muestra la relación
  - Estados: Representa los estados y eventos de un objeto
  - Actividades: Describe el flujo de trabajo.

Ilustración 4. Patrones básicos de diseño

En esta ilustración podemos observar los patrones básicos de los diversos diagramas UML, incluyendo todos los tipos de diagramas UML y qué muestra exactamente dentro de ella.

Actividad 5

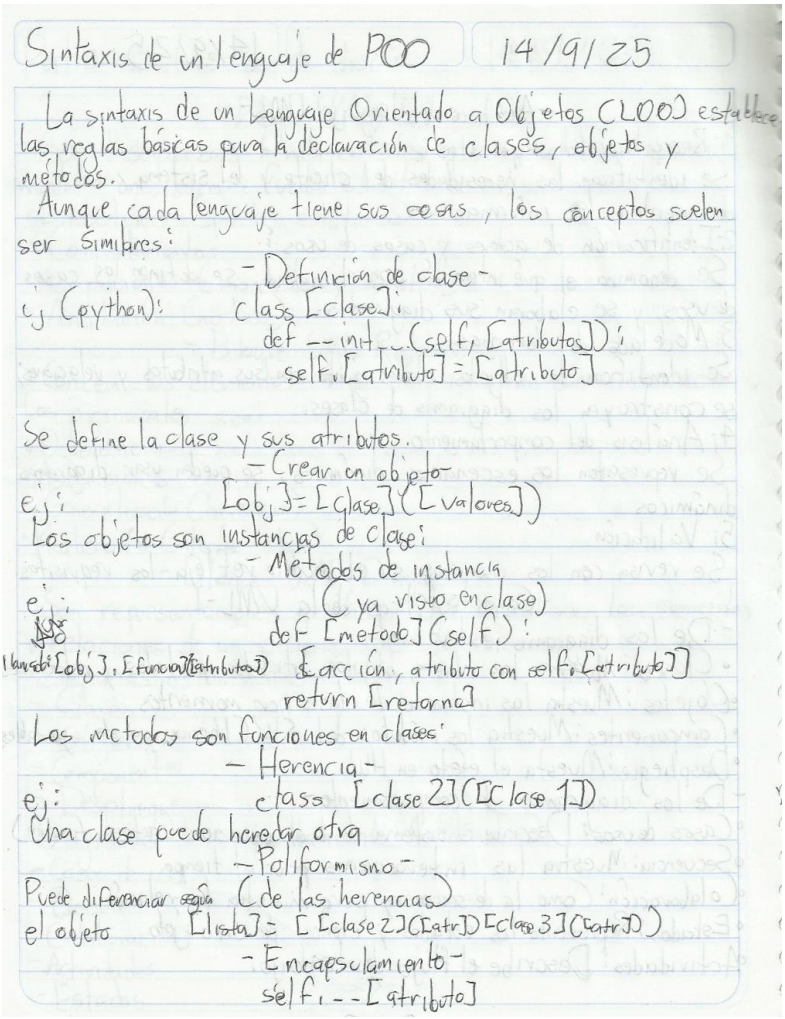
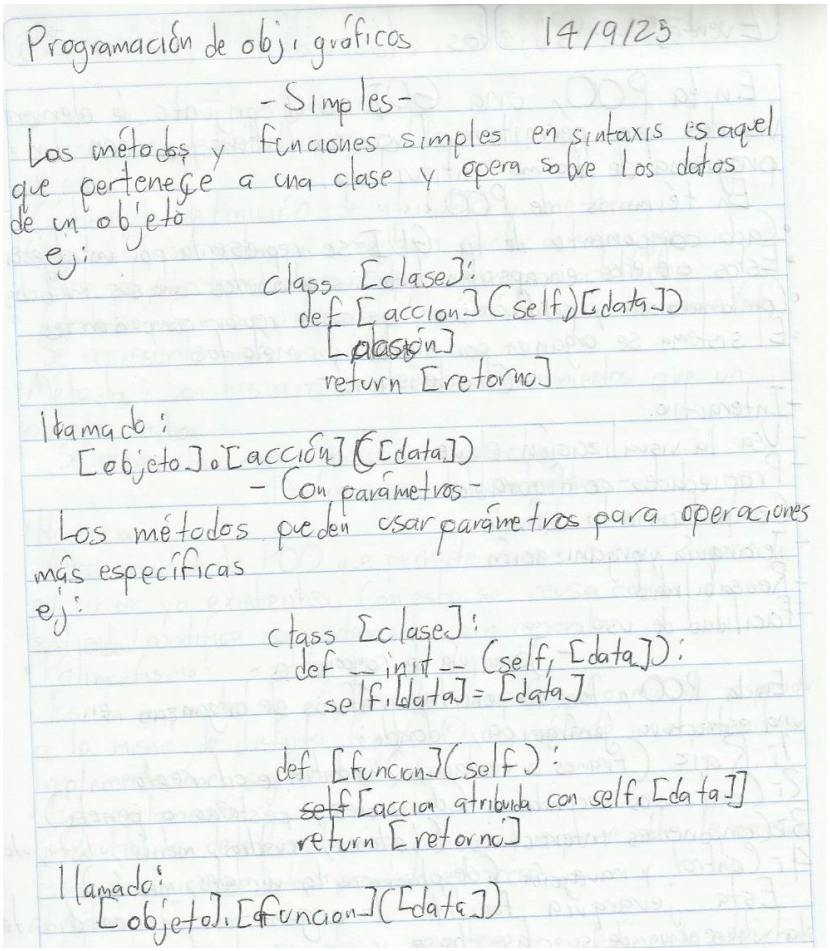


Ilustración 5. Sintaxis de un lenguaje de POO

En esta ilustración podemos observar las reglas de sintaxis del lenguaje orientado a objetos con ejemplos en Python. Con esto podemos armar nuestro código de POO de manera directa y organizada.

Actividad 6


Programación de obj. gráficos 14/9/23

- Simples -

Los métodos y funciones simples en sintaxis es aquel que pertenece a una clase y opera sobre los datos de un objeto

ej:

```
class [clase]:
    def [accion](self,[data])
        [acción]
        return [retorno]
```

llamado:

```
[objeto].[acción]([data])
```

- Con parámetros -

Los métodos pueden usar parámetros para operaciones más específicas

ej:

```
class [clase]:
    def __init__(self, [data]):
        self.[data] = [data]

    def [funcion](self):
        self.[accion atribuida con self.[data]]
        return [retorno]
```

llamado:

```
[objeto].[funcion]([data])
```

Ilustración 6. Programación de objetos gráficos

En esta ilustración podemos observar las propiedades de la programación de objetos gráficos usando a Python como estructura directa para determinar tanto las propiedades de la creación de la clase como el llamado (por un objeto al menos).

Actividad 7

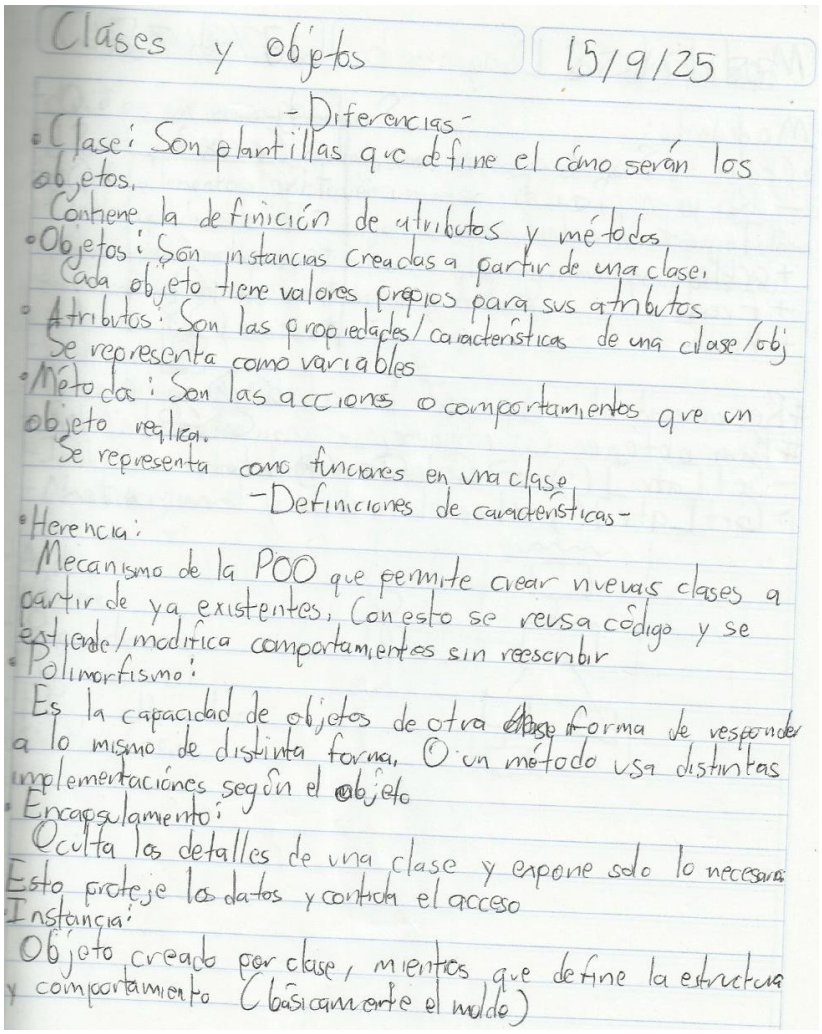
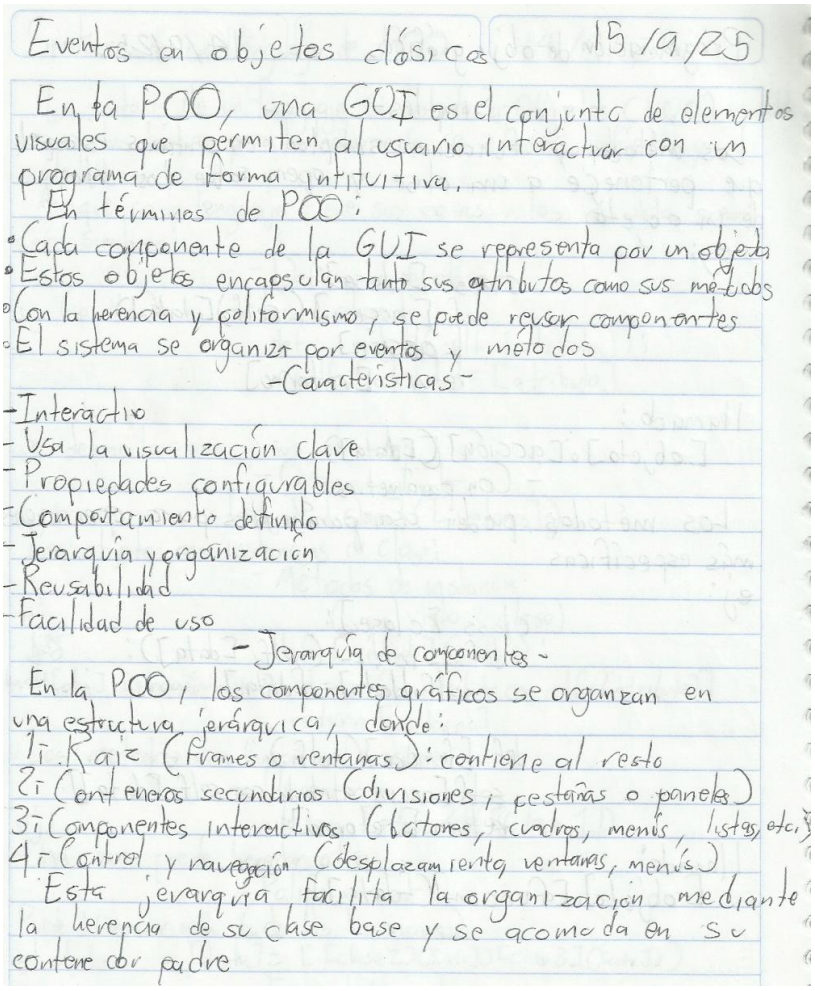


Ilustración 7. Clases y objetos

En esta ilustración podemos ver las definiciones directas entre clases y objetos, estos últimos con sus partes (métodos y atributos). También podemos observar las propiedades que les puedes asignar a las clases.

Actividad 8



Eventos en objetos clásicos 15/9/25

En la POO, una GUI es el conjunto de elementos visuales que permiten al usuario interactuar con un programa de forma intuitiva.

En términos de POO:

- Cada componente de la GUI se representa por un objeto
- Estos objetos encapsulan tanto sus atributos como sus metodos
- Con la herencia y poliformismo, se puede reusar componentes
- El sistema se organiza por eventos y metodos

-Características-

- Interactivo
- Usa la visualización clave
- Propiedades configurables
- Comportamiento definido
- Jerarquía y organización
- Reusabilidad
- Facilidad de uso

- Jerarquía de componentes -

En la POO, los componentes gráficos se organizan en una estructura jerárquica, donde:

1. Raiz (Frames o ventanas): contiene al resto
2. Contenedores secundarios (divisiones, pestañas o paneles)
3. Componentes interactivos (botones, cuadros, menús, listas, etc.)
4. Control y navegación (desplazamiento ventana, menús)

Esta jerarquía facilita la organización mediante la herencia de su clase base y se acomoda en su contenedor padre

Ilustración 8. Eventos en objetos clásicos

En esta ilustración podemos observar las definiciones de las interfaces gráficas por parte de la programación orientada a objetos, recalcando que las características son similares a la programación orientada a objetos.

Retroalimentación

En este momento no quiero opinar mucho con respecto a mis sentimientos con los trabajos debido a que tengo un bias gracias a que he gastado muchos recursos y energía en actividades medianamente irrelevantes lo cual puede llevar a ataques negativos poco éticos y profesionales

Con respecto a las investigaciones puedo decir que este tema es bastante interesante de repasar en momentos pacíficos y de poca presión; aprendiendo de tal manera que uno puede simplificar mucho su metodología de programación mediante el nombramiento de objetos y categorías.

Con respecto al tema en general puedo decir que este tema es bastante poderoso para la construcción rápida de un archivo de programación.