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

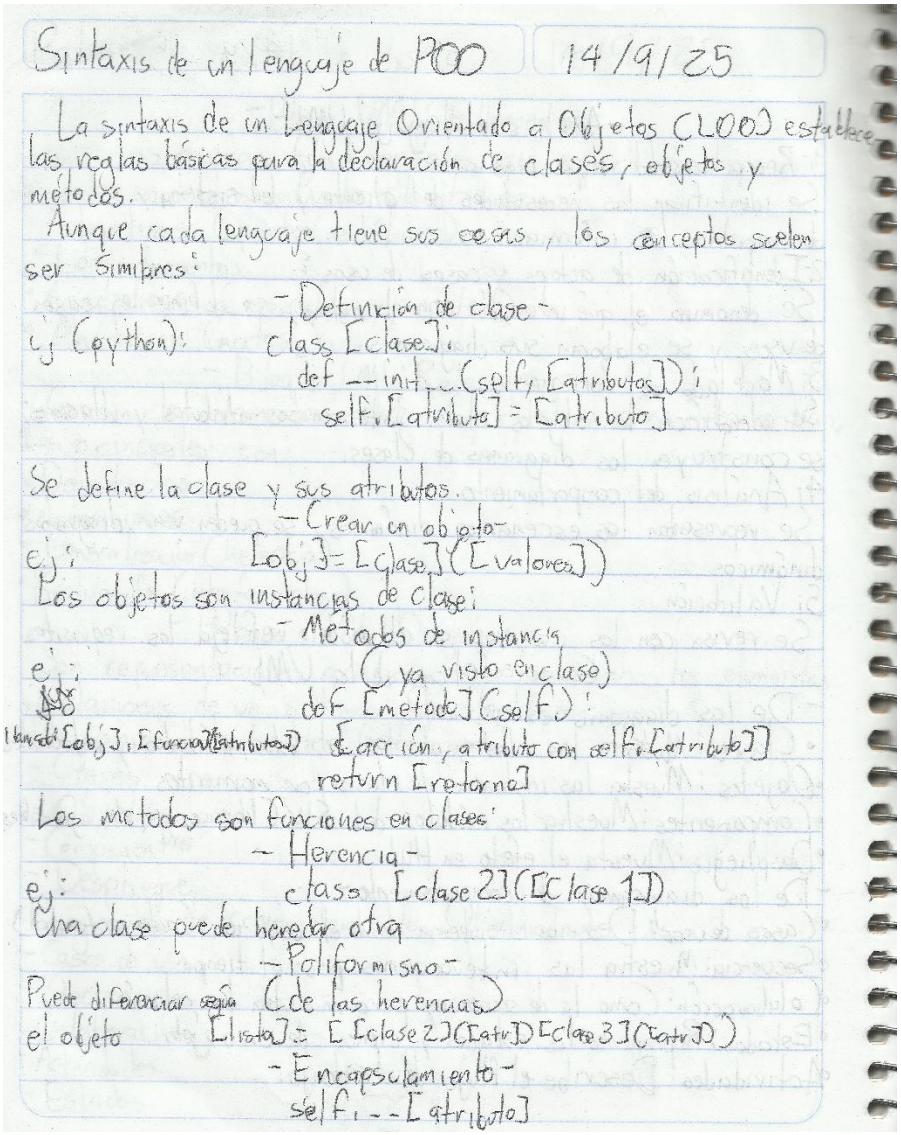
Docente:

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

Noviembre 2025

Tabla de Ilustraciones

Actividad 1


Sintaxis de un lenguaje de POO 14/9/25

La sintaxis de un Lenguaje Orientado a Objetos (LOO) establece las reglas básicas para la declaración de clases, objetos y métodos.
Aunque cada lenguaje tiene sus cosas, los conceptos suelen ser similares:

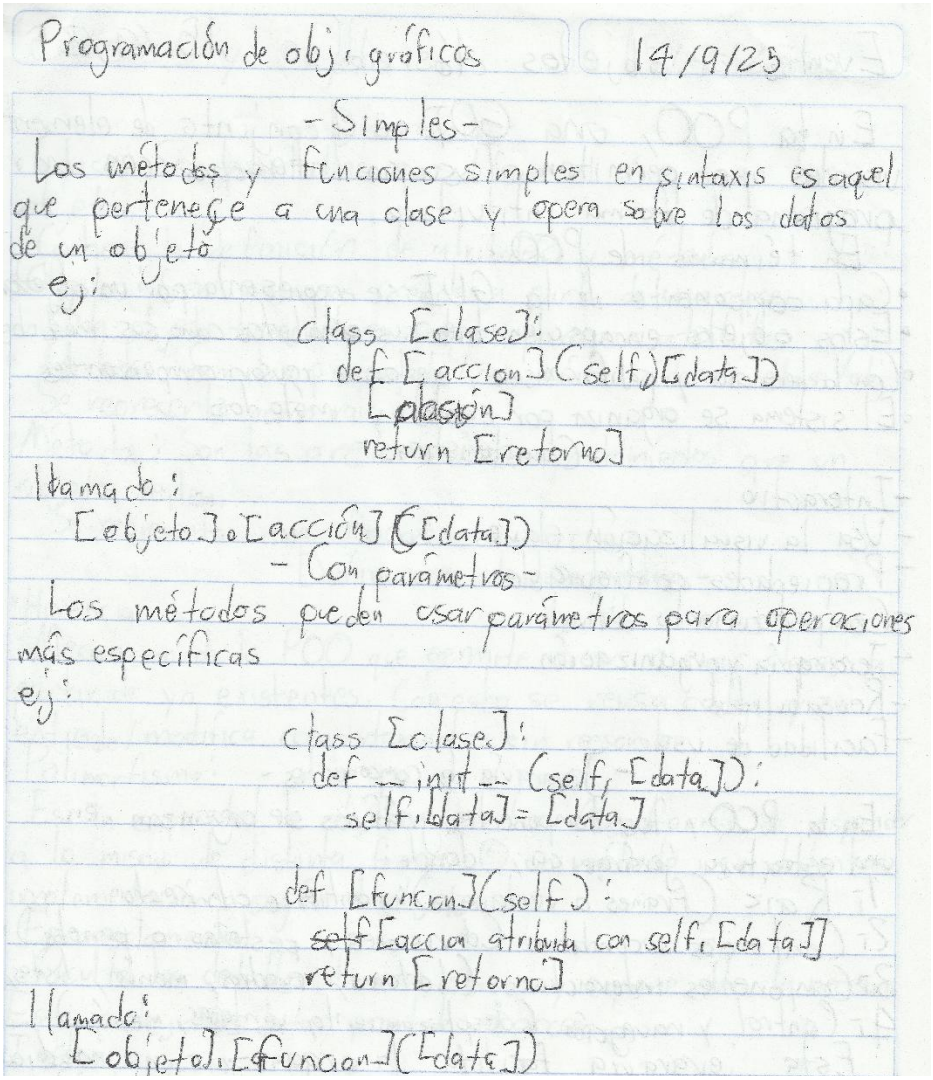
- Definición de clase -
ej (python): class [clase]:
def __init__(self, [atributos]):
self.[atributo] = [atributo]

Se define la clase y sus atributos.
- Crear un objeto -
ej: [obj] = [clase]([valores])
Los objetos son instancias de clase:
- Metodos de instancia (ya visto en clase)
ej: def [metodo](self):
[obj].[funcion]([atributos]) &acción, atributo con self.[atributo]
return [retorna]
Los metodos son funciones en clases:
- Herencia -
ej: class [clase 2]([clase 1])
Una clase puede heredar otra
- Poliformismo -
Puede diferenciar según (de las herencias)
el objeto [lista] = [[clase 2]([atr]) [clase 3]([atr])]
- Encapsulamiento -
self.__[atributo]

Ilustración 1. 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 2


Programación de obj. gráficos 14/9/25

-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])
        [clasión]
        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]
```

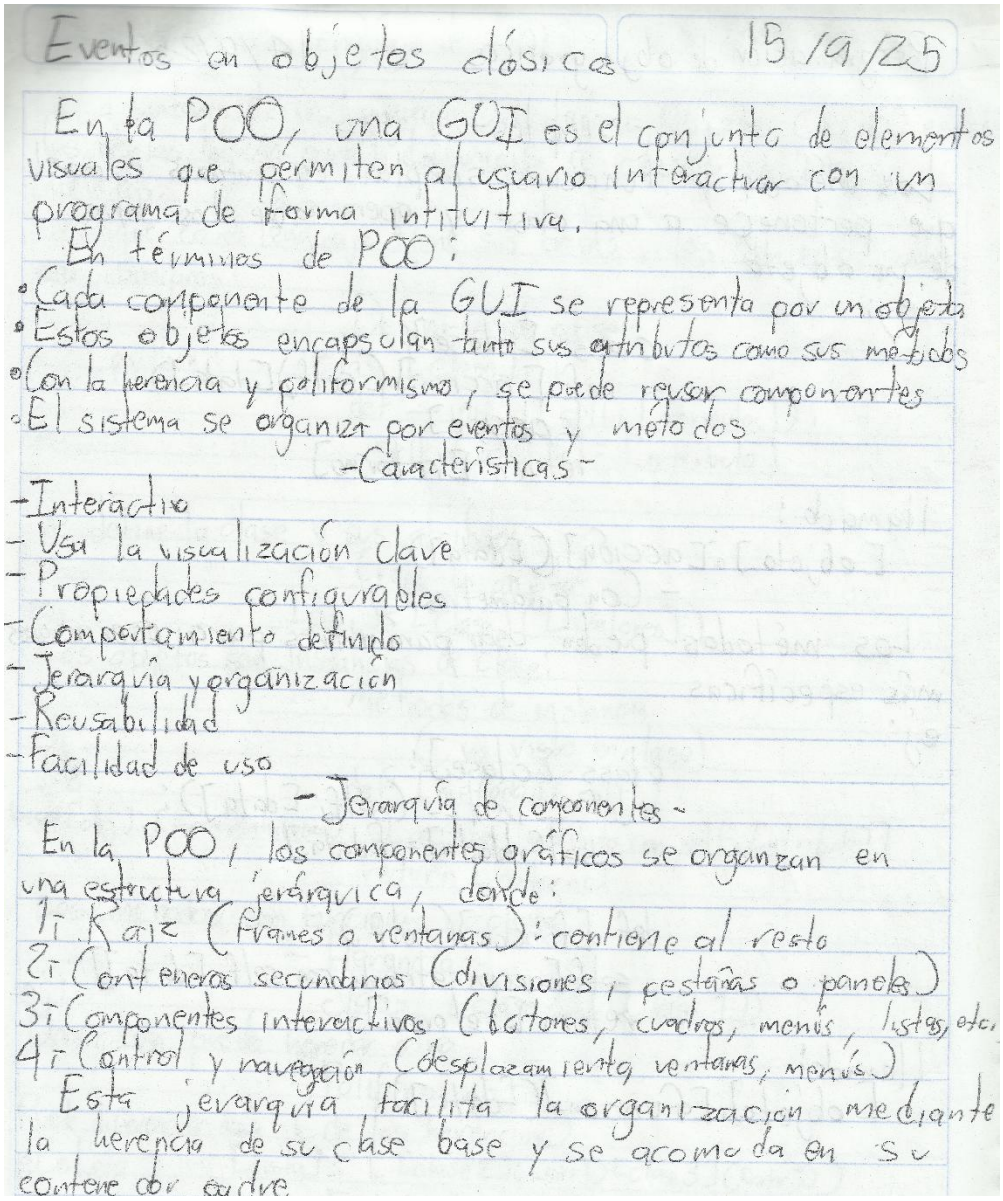
llamado:

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

Ilustración 2. 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 3



Eventos en objetos básicos 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 métodos
- Con la herencia y polimorfismo, se puede reusar componentes
- El sistema se organiza por eventos y métodos

-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:

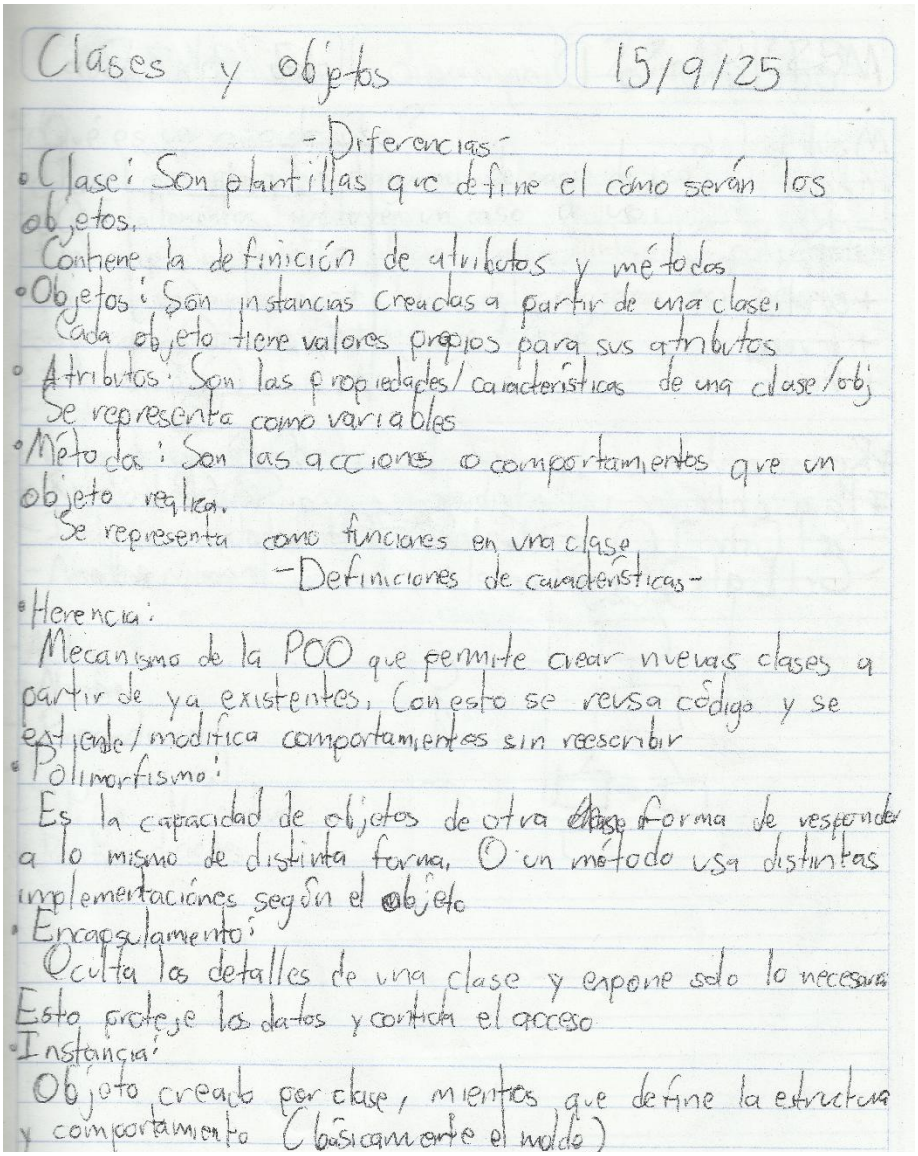
1. Raíz (Frames o ventanas): contiene al resto
2. Contenedores secundarios (divisiones, pestañas o paneles)
3. Componentes interactivos (botones, cuadros, menús, listas, etc.)
4. Control y navegación (desplazamiento ventanas, menús)

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

Ilustración 3. Eventos en objetos gráficos

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.

Actividad 4


Clases y objetos 15/9/25

-Diferencias-

- Clase: Son plantillas que define el cómo serán los objetos.
  Contiene la definición de atributos y métodos
- Objetos: Son instancias creadas a partir de una clase.
  Cada objeto tiene valores propios para sus atributos
- Atributos: Son las propiedades/características de una clase/obj
  Se representa como variables
- Métodos: Son las acciones o comportamientos que un objeto realiza.
  Se representa como funciones en una clase

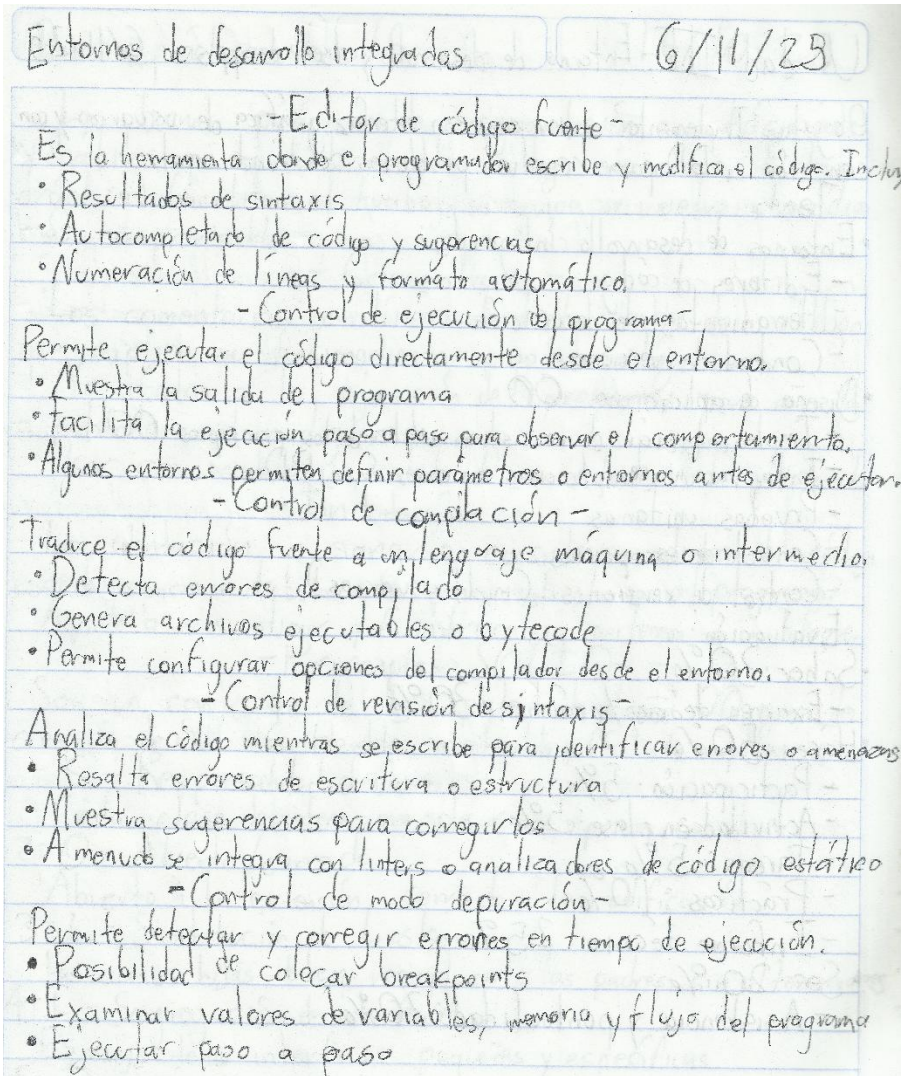
-Definiciones de características-

- Herencia:
  Mecanismo de la POO que permite crear nuevas clases a partir de ya existentes, Con esto se reusa código y se extiende/modifica comportamientos sin reescribir
- Polimorfismo:
  Es la capacidad de objetos de otra clase forma de responder a lo mismo de distinta forma, O un método usa distintas implementaciones según el objeto
- Encapsulamento:
  Oculta los detalles de una clase y expone solo lo necesario
  Esto protege los datos y controla el acceso
- Instancia:
  Objeto creado por clase, mientras que define la estructura y comportamiento (básicamente el molde)

Ilustración 4. 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 5

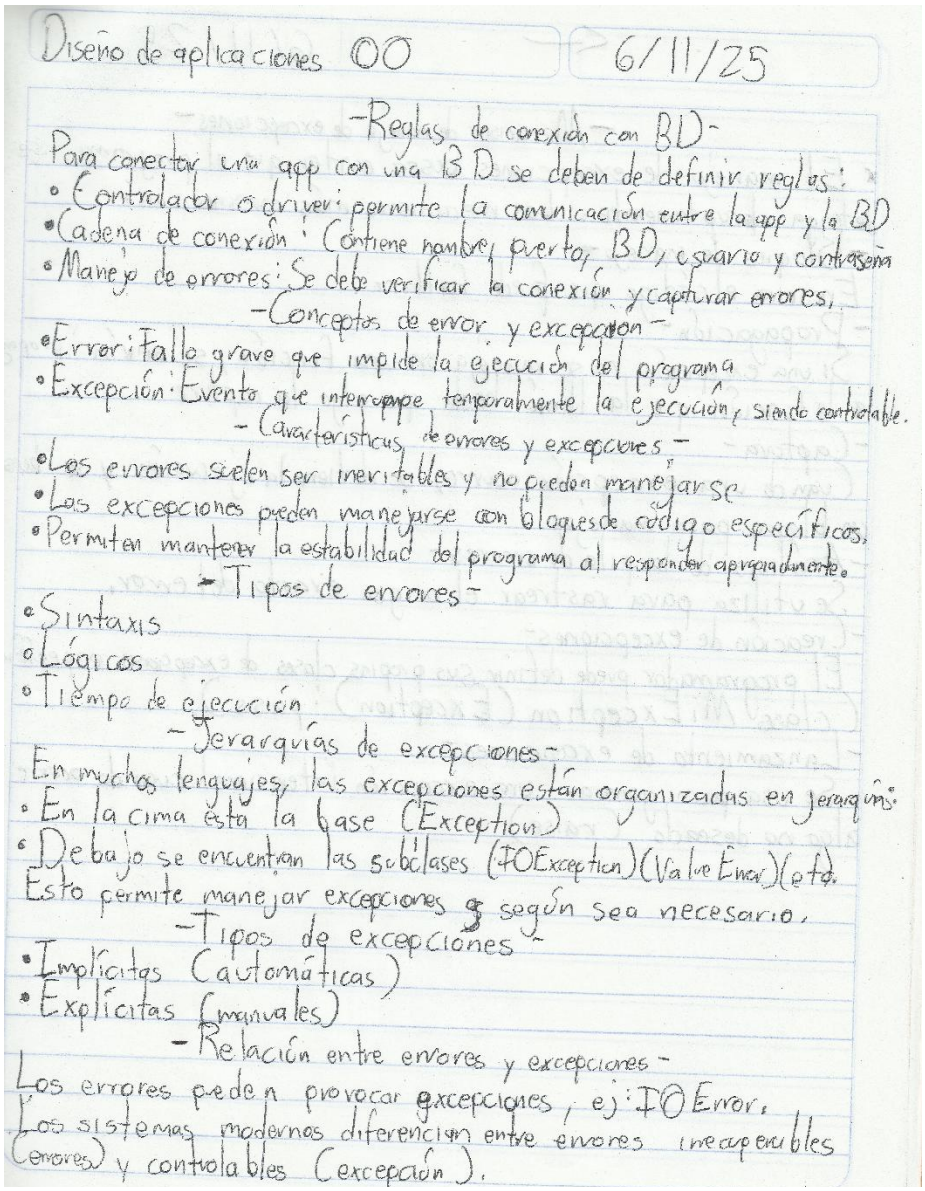

Entornos de desarrollo integrados 6/11/23

-Editor de código fuente-
Es la herramienta donde el programador escribe y modifica el código. Inclu
• Resaltados de sintaxis
• Autocompletado de código y sugerencias
• Numeración de líneas y formato automático.
-Control de ejecución del programa-
Permite ejecutar el código directamente desde el entorno.
• Muestra la salida del programa
• Facilita la ejecución paso a paso para observar el comportamiento.
• Algunos entornos permiten definir parametros o entornos antes de ejecutar.
-Control de compilación-
Traduce el código fuente a un lenguaje máquina o intermedio.
• Detecta errores de compilado
• Genera archivos ejecutables o bytecode
• Permite configurar opciones del compilador desde el entorno.
-Control de revisión de sintaxis-
Analiza el código mientras se escribe para identificar errores o amenazas
• Resalta errores de escritura o estructura
• Muestra sugerencias para corregirlos
• A menudo se integra con linters o analizadores de código estático
-Control de modo depuración-
Permite detectar y corregir errores en tiempo de ejecución.
• Posibilidad de colocar breakpoints
• Examinar valores de variables, memoria y flujo del programa
• Ejecutar paso a paso

Ilustración 5. Entornos de desarrollo integrados

En esta ilustración podemos observar los entornos de desarrollo integrados. En este datos se ve con respecto a las características de estos entornos: editor de código, control de ejecución, control de compilación, control de sintaxis y modo depuración.

Actividad 6



Diseño de aplicaciones OO 6/11/25

-Reglas de conexión con BD-

Para conectar una app con una BD se deben de definir reglas:
- Controlador o driver: permite la comunicación entre la app y la BD
- Cadena de conexión: Contiene nombre, puerto, BD, usuario y contraseña
- Manejo de errores: Se debe verificar la conexión y capturar errores.

-Conceptos de error y excepción-

- Error: Fallo grave que impide la ejecución del programa
- Excepción: Evento que interrumpe temporalmente la ejecución, siendo controlable.

-Características de errores y excepciones-

- Los errores suelen ser inevitables y no pueden manejarse
- Los excepciones pueden manejarse con bloques de código específicos.
- Permiten mantener la estabilidad del programa al responder apropiadamente.

-Tipos de errores-

- Sintaxis
- Lógicos
- Tiempo de ejecución

-Jerarquías de excepciones-

En muchos lenguajes, las excepciones están organizadas en jerarquías:
- En la cima está la base (Exception)
- Debajo se encuentran las subclases (IOException)(ValueError)(etc.
Esto permite manejar excepciones según sea necesario.

-Tipos de excepciones-

- Implícitas (automáticas)
- Explícitas (manuales)

-Relación entre errores y excepciones-

Los errores pueden provocar excepciones, ej: IO Error.
Los sistemas modernos diferencian entre errores irrecuperables (errores) y controlables (excepción).

Ilustración 6. Diseño de app OO

En esta ilustración podemos observar el tema de diseño de aplicaciones orientado a objetos. En este tema vemos las reglas de conexión a la base de datos y varias cosas del diseño de excepciones.

Retroalimentación

Este tema es definitivamente un tema que nos puede ayudar a manejar varias situaciones ya conocidas previamente, pero de una manera más radical: los términos audiovisuales. Con las interfaces gráficas uno puede convertir funciones complejas y con subfunciones en algo fácil de comprender para el usuario (si se aplica de manera correcta, por supuesto).

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.

En cuanto a la investigación no puedo decir mucho lamentablemente ya que fue corta y fue de temas vistos en otras clases así que… xd.