1. Recursividad
    a. Concepto
        i. La función de la recursividad es la de brindar nuevos métodos para la realización de estructuras cíclicas. Estas pueden ser implementadas en toda situación cuyos resultados puedan ser expresados como una sucesión de movimientos, pasos o transformaciones que se rigen por un grupo de normas no ambiguas.
    b. Estructura:
        i. Caso base: una solución básica para un caso peculiar
        ii. Caso recursivo: la resolución de este problema implica a que se vuelva a usar la función original, con parámetros que se asimilen al del caso base. Pasos:
            1. El procedimiento se llama a si mismo
            2. El problema es resuelto, tratando un problema similar a menor escala
            3. A medida que el tamaño del proyecto vaya disminuyendo este asegurara que el problema principal se haya resuelto
    c. Beneficios:
        i. Es lo que mas se asimila a una descripción matemática
        ii. Son mas simples y sencillos de analizar
        iii. Logran adaptarse fácilmente a las estructuras de datos recursivas
        iv. Sus algoritmos son estructurados, modulares, simples pero elegantes

2. ¿Qué es la programación orientada a objetos?
    a. De manera literal se puede definir a la Programación Orientada a Objetos (POO) como un modelo sistemático de programación de tal manera que el código sea ordenado en varias unidades (clases), cuya finalidad es la de la creación de objetos que logren relacionarse entre si para cumplir un objetivo general
    b. Partes:
        i. Clases: conceptualmente las clases es donde se almacenará todas las características y acciones de los objetos que creemos
        ii. Propiedades: Son las características de una clase.
        iii. Métodos: Lo métodos son las acciones que una clase puede llegar a realizar.
        iv. Objetos: Son todos aquellos que poseen propiedades y métodos que cumplen con un objetivo especificado por el usuario.

3. Pilares de POO
    a. Abstracción: Es la acción de fragmentar los datos de un objeto para que se logre generar una nueva clase.
    b. Encapsulamiento: Este pilar es el que nos permite modificar la privacidad de nuestras clases. Es decir, limita y define el acceso a las clases

c. Herencia: La herencia se pude definir como una relación especial entre dos clases, las cuales son clase base y clase derivada, donde la derivada adquiere la posibilidad de utilizar propiedades y métodos de la clase base, llegando incluso a modificar estas. Resumiendo, la herencia permite la ceración de nuevas clases a partir de otras
d. Polimorfismo: El polimorfismo en líneas generales es la situación en la que un método recibe un parámetro para que este mismo abarque varios casos. Es decir que es utilizado para que los métodos de un mismo nombre tengan comportamientos distintos.

4. Diagrama de clases
    a. El diagrama de clases nos presentara la oportunidad de representar gráficamente la estructura del sistema, en este mismo se podrán observar cada una de las clases creadas y sus interacciones, tales como la herencia. Estas estarán representadas en bloques que están unidos por medio de líneas y arcos. En líneas mas generales y simplificadas, los Diagramas de son interacciones entre nodos y arcos, representando interacciones, relaciones, interfaces, y colaboraciones entre las mismas clases.

5. Relaciones entre clases
    a. Asociaciones: La asociación se puede definir como la relación estructural que genera una conexión entre dos objetos. Estas usualmente son bidireccionales a excepción de cuando se les restringe la navegación en un solo sentido
        i. Multiplicidad de asociaciones

Reflexiva	1	Uno y solo 1
Binaria	0..1	Cero o Uno
N-aria	N..M	De N hasta M
||	*	Cero o Varios
||	0.. *	Cero o varios
||	1.. *	Uno o varios (mínimo 1)

            1. Cuando la multiplicidad mínima es 0, su relación es opcional. Pero cuando su relación mínima es mayor o igual a 1, su relación es obligatoria.

        ii. Agregación
            1. En estos casos particulares permite la representación entre una clase base y sus elementos agregados, en otras palabras, las clases agregadas no afectan al funcionamiento principal de la clase base
        iii. Composición

1. La composición representa una clase base que está compuesta principalmente por otras clases que son indispensables para su funcionamiento correcto.

iv. Podríamos concluir que la agregación es una relación pasiva y la composición una relación dominante

b. Dependencia: es aquella relación sistemática entre dos elementos en el cual un mínimo cambio al elemento independiente influye radicalmente en la clase dependiente, esta es representada por una línea discontinua con una punta de flecha señalando a la clase dependiente.

c. Herencia/ Generalización:

i. La herencia es la relación entre clases bases y clases derivadas, es decir que objetos de distintas clases pueden llegara tener atributos y comportamiento por los métodos similares

6. Clases padre clases hijo
    a. Clase padre:
        i. También llamada clase base, es aquella de la cual se pueden crear varias clases derivadas (clases hijas)
    b. Clase Hijo
        i. También llama clase derivada, es la que un inicio comparte similitudes con la clase padre llegando a presentar atributos y métodos similares, pero esta puede ser modificada para necesidades del usuario.