Luis Julian Peña Rodriguez

1. Conceptos básicos de programación:

A. **¿Qué es la Programación Orientada a Objetos (POO)?**

La Programación Orientada a Objetos (POO) es un enfoque en la programación que se basa en el concepto de objetos, los cuales pueden contener datos en forma de campos (conocidos como atributos o propiedades) y código en forma de procedimientos (conocidos como métodos). Los principios fundamentales de la POO son el encapsulamiento, la abstracción, la herencia y el polimorfismo. Estos principios facilitan la creación de software modular, reutilizable y más fácil de mantener. A continuación, se presenta una breve explicación de los principios fundamentales de la POO:

**Encapsulamiento**: Agrupa datos y métodos que operan sobre estos datos dentro de una clase, con la finalidad de proteger los datos de accesos no autorizados y modificaciones inesperadas. Esto permite un control de acceso a través de modificadores de visibilidad (public, private, protected).

**Abstracción**: Permite modelar entidades del mundo real seleccionando solo los atributos y comportamientos necesarios para el propósito del software, ignorando los detalles irrelevantes. Esto ayuda a simplificar la complejidad del sistema.

**Herencia**: Permite crear nuevas clases a partir de clases existentes, heredando atributos y métodos de estas. Esto facilita la reutilización de código y la creación de jerarquías de clases.

**Polimorfismo**: Permite que una interfaz pueda ser utilizada para diferentes tipos de objetos, permitiendo que los mismos métodos se comporten de manera diferente según el objeto que los invoque.

B. ¿Cuál es la diferencia entre una Clase y una Interfaz?

Las diferencias más relevantes entre estos dos términos serían las siguientes:

Las clases pueden contener tanto la declaración como la implementación de métodos, mientras que las interfaces solo declaran métodos que deben ser implementados por las clases que las implementan.

Las clases pueden heredar de otras clases, mientras que las interfaces no pueden heredar de una clase, pero una clase puede implementar múltiples interfaces.

Las clases pueden tener constructores y atributos, mientras que las interfaces no pueden tener ninguna de estas dos cosas.

En conclusión, las clases se usan cuando se necesita definir una implementación detallada y el estado de los objetos, y las interfaces se usan cuando se desean definir contratos y asegurar que diferentes clases proporcionen ciertas funcionalidades de manera consistente.

C. ¿Cuál es la diferencia entre un método estático y uno no estático?

La diferencia más relevante entre métodos estáticos y no estáticos es la siguiente:

Métodos no estáticos: Se utilizan cuando se requiere manipular datos específicos de una instancia de la clase. Estos métodos están asociados con instancias individuales y pueden acceder y modificar directamente los atributos de esas instancias.

Métodos estáticos: Se utilizan para operaciones que no dependen de instancias individuales y pueden ser llamados directamente a través de la clase. Son útiles para funciones utilitarias o métodos que realizan cálculos generales y no requieren acceder al estado de un objeto específico.

El uso de estos métodos depende de si la operación requiere acceso y manipulación de datos específicos de instancias de la clase o si puede ser generalizada y compartida entre todas las instancias.

D. ¿Qué es un ciclo de Vida de Software?

El ciclo de vida de software se refiere a la secuencia de etapas o fases que un software atraviesa desde su inicio hasta su obsolescencia. Este ciclo proporciona una estructura y metodología para el desarrollo, mantenimiento y eventual desuso del software. Las etapas típicas incluyen:

Recolección de Requisitos: Especificación de los requerimientos que el software debe cumplir.

Análisis: Elaboración de un diseño detallado que puede incluir casos de uso y diagramas.

Diseño: Definición de la arquitectura del software, estructura de datos, etc.

Implementación: Escritura del código del software y realización de pruebas unitarias.

Pruebas: Verificación para asegurar que el software cumpla con los requisitos establecidos.

Despliegue: Puesta en funcionamiento del software en el entorno de producción.

Mantenimiento: Actualizaciones y correcciones necesarias mientras el software está en producción.

Retiro: Planificación y ejecución de la retirada del software cuando ya no es necesario.

Los ciclos de vida de software varían según las metodologías utilizadas y las necesidades específicas del proyecto y la organización.

E. ¿Cuál es la diferencia entre throw; y throw ex; dentro de un catch

La diferencia entre estos dos es que throw relanza la excepción original con toda su traza de pila. Esto significa que se mostrarán todos los errores que se generaron a lo largo de la ejecución del programa, conservando el historial completo desde donde se originó hasta donde se maneja o relanza. En cambio, throw ex relanza la excepción como una nueva traza de pila que únicamente muestra el contexto del bloque catch actual. Esto significa que solo se verá el último error arrojado.