Prueba Diagnostica POO

Nombre: Armando Leonel Yañez Arevalo

Responde las siguientes preguntas, recuerda esto es una prueba diagnóstica, no hay respuestas incorrectas, pero algunas respuestas serán más precisas que otras y esto no afectará tu calificación pero si nos ayudará a entender tu nivel de conocimiento para poder ayudarte mejor.

Parte I: Teoría y Conceptos de la POO

1. Define brevemente la Programación Orientada a Objetos (POO) y explica por qué es un paradigma importante en el desarrollo de software.

R= Es una forma de programar que se inspira en los objetos del mundo real, se basa en objetos reales para crear objetos virtuales.

2. Diferencia entre clases y objetos en la POO. Proporciona un ejemplo práctico de cada uno.

R= Las clases son como una especie de plantilla y un objeto es como tal la creación de una variable con las características dadas en la clase, un ejemplo puede ser la creación de una clase persona donde posteriormente en el código se crea a juanito que es tipo persona y juanito tendrá las capacidades de la clase.

3. Describe los pasos para la declaración de una clase en un lenguaje de programación orientado a objetos de tu elección.

R= Primero, se utiliza la palabra class para comenzar la declaración de la clase, después, se le da un nombre a la clase y se usa la palabra public para indicar que los miembros declarados serán accesibles desde fuera de la clase. Entre paréntesis se incluyen los atributos y métodos de la clase, después se le puede agregar a la clase sus respectivos métodos y al final se colocan en privado las variables correspondientes en private.

4. ¿Cuál es la función de los constructores y destructores en la programación orientada a objetos? Proporciona un ejemplo de uso.

R= Los constructores son se usan para inicializar los atributos de un objeto, por ejemplo al crear un objeto persona que tiene nombre, si no le pones nombre a este objeto pero imprimes el nombre la consola te puede llegar a imprimir N/A ya que está inicializado con ese valor, esto

indicando que está vacío y sobre el destructor no me acuerdo…

5. Explica el proceso de instanciación de objetos en la POO y cómo se relaciona con la creación de instancias de una clase.

R= Solo se que para instanciar un objeto es necesario llamar a la clase, como si se estuviera creando una variable de tipo clase creada, es importante nombrarla.

6. ¿En qué consiste la sobrecarga de funciones miembro y por qué es útil en la programación orientada a objetos?

R= La sobrecarga es la forma de utilizar una misma función pero recibiendo distintas cosas, por ejemplo una función que te sume números, pero si le das 2, entre el método sumar (int num1, int num2) o si le das 3 números entre la función sumar sumar (int num1, int num2, int num 3), esto es importante para el funcionamiento de algunos métodos ya que puede darle flexibilidad a la clase y evitar errores.

7. ¿Qué significa la auto referencia del objeto (this) en la POO y cómo se utiliza comúnmente en la implementación de métodos?

R= Recuerdo que se utilizaba el -> y creo que era para asignar valores.

8. Explica el concepto de clases compuestas y proporciona un ejemplo de su aplicación.

R= No recuerdo.

Parte II: Encapsulamiento

9. Define los niveles de encapsulamiento: público, privado y protegido. Proporciona un ejemplo de cada uno.

R= El publico es aquel que todos pueden acceder, aqui normalmente se ponen los métodos que son accesibles a todos, el privado es aquel que no se puede acceder, estos se utiliza par proteger datos importantes y el protegido son aquellos datos que son accesibles para algunos que nosotros configuremos como amiga.

10. Explica el concepto de funciones amigas (friend) en la POO y cómo afectan al encapsulamiento.

R= Las clases amigo son aquellas que pueden acceder a nuestra sección protegida, estos pueden jugar con los datos o acceder a métodos que se encuentren ahí.

Parte III: Datos Estáticos

11. Describe los diferentes aspectos de datos estáticos: variables estáticas, miembros estáticos de una clase y métodos estáticos. Proporciona ejemplos para cada uno.

R= Se que las variable estáticas son aquellas que no cambian nunca de valor, pero no estoy seguro.

Parte IV: Tipos Abstractos de Datos

12. Explora la programación modular y su relación con los Tipos Abstractos de Datos (TAD). ¿Cómo se implementa la interfaz, la implementación y la utilización de módulos en este contexto?

R= No se ;c

13. Define y explica la importancia de los espacios de nombres en la programación orientada a objetos.

R= La facilidad de su uso.

Parte V: Herencia

14. Explica el concepto de herencia y sus ventajas en la POO. Proporciona ejemplos de clases derivadas.

R= La herencia como tal es la forma en la que una clase puede adquirir atributos y métodos de una clase padre, para usos posteriores, un ejemplo podría ser una clase persona que derive una clase trabajador y una clase estudiante, ambos tienen atributos y métodos de humanos pero también tienen diferentes métodos y atributos según su labor.

15. ¿Cómo se gestionan los constructores y destructores en clases derivadas? Proporciona un ejemplo.

R= Se utiliza el constructor sólo en los atributos nuevos que se le agreguen a la clase derivada, por ejemplo en una clase padre persona que contenga nombre y edad, después una clase derivada estudiante que incluya extra el atributo escuela, en el constructor del estudiante solo se agregara el de el atributo escuela.

16. Describe los diferentes tipos de herencia y explora la herencia múltiple. ¿Cuáles son los desafíos asociados con la herencia múltiple?

R= Se que existe la herencia explícita e implícita, una para cuando el objeto ya tiene asignado valores y otra se usa cuando se agregan valores dentro de la clase, de la herencia múltiple no recuerdo.

Parte VI: Polimorfismo

17. Define el concepto de polimorfismo y diferencia entre polimorfismo estático y dinámico.

R= Se que el polimorfismo es crear métodos con el mismo nombre dentro de diferentes clases con diferentes funciones, o la misma función pero con ligeros cambios.

18. Explica el uso de funciones virtuales y cómo contribuyen al polimorfismo en la POO.

R= No recuerdo.

19. ¿Qué son las clases abstractas? Proporciona un ejemplo de su implementación.

R= No recuerdo,

20. Describe el uso de las palabras clave override y final en el contexto del polimorfismo.

R= No recuerdo.

Parte VII: Controlador de Versiones (Git y Github)

21. Define el controlador de versiones y explora la importancia de Git y Github en el desarrollo de software.

R= Es una forma de almacenamiento y control de nuestros proyectos, el cual facilita los trabajos colaborativos y mantener un orden en el desarrollo de proyectos.

22. Describe el flujo de trabajo típico de Git (Git workflow) y explique cada etapa.

R= Hay 2 instrucciones esenciales el pull y el push, al pull es para jalar los cambios del proyecto a nuestra máquina y el push para enviar, el push va acompañado de un commit el cual ayuda a saber qué cambios se realizaron, estos cambios se verán como ramificaciones.

23. Explica los pasos para iniciar un nuevo repositorio en Git.

R= Primero desde la página de github es necesario crearlo y configurarlo para posteriormente colocarlo en la computadora y subir los archivos de nuestros codigos, hay algunas IDE que directamente lo enlazan.

24. Detalla cómo realizar y modificar commits en Git.

R= Para hacer commits se debe haber hecho un cambio y rellenar un recuadro donde vendrán los cambios realizados.

25. ¿Cómo se remueven cambios del stage y se revierten cambios en Git?

R= Se que para revertir se usa el revert.

26. Explica el concepto de repositorios remotos y cómo se utilizan en colaboración.

R= Repositorios que están en la nube y se pueden usar en colaboración jalando y pusheando información desde diferentes máquinas.

27. Describe la creación y manejo de ramas en Git.

R= Se pueden crear ramas que son como versiones del codigo main o versiones de otra rama la cual funciona para hacer cambios o pruebas para posteriormente fusionar a la main.

28. ¿Cómo se unifican ramas al branch principal (main) en Git?

R= No se.

29. Explica la importancia del manejo del historial de versiones en Git.

R= Se puede ver desde la interfaz gráfica de la IDE o usando el comando git log.

30. ¿Cuál es la diferencia entre Git y Github? ¿Cómo se relacionan entre sí?

R= Git es local y github tiene servicios en línea, se relacionan porque funcionan muy parecidos solo que github se utiliza con otros propósitos como trabajo colaborativo.