CVDS 012023 Parcial corte 1

Parcial que contiene preguntas basadas en los siguientes temas:

1. SVC
2. Principios SOLID.
3. Patrones de diseño
4. Pruebas Unitarias.
5. TDD

Puntos: 57/71 Hora: 20:59

1. SVC

✕ **Incorrecto** 0/2 Puntos

1. Que traduce las siglas SVC? *

- (●) Software Version Control
- ( ) System Version Control
- ( ) Source Version Control

✓ **Correcto** 2/2 Puntos

2. Para crear un repositorio local desde un repositorio remoto, cual comando debería usar *

- [ ] git pull
- [x] git clone
- [ ] git push
- [ ] git checkout
- [ ] git commit

✓ **Correcto** 2/2 Puntos

3. ¿Cuál es el comando para guardas cambios en el repositorio local? *

- [ ] git push
- [ ] git save
- [x] git commit
- [ ] git push

✓ **Correcto** 2/2 Puntos

4. Git es un sistema: *

- [ ] Centralizado
- [ ] Hibrido
- [x] Distribuido

2. Principios SOLID.

✓ **Correcto** 2/2 Puntos

5. El acrónimo de SOLID es:

- Singular Creation principle, Outside and Near principle , Liskov sustitution principle ,Inversion sentence principle, Deep Inner principle.
- Single Responsability principle, Open and Close principle, Liskov sustitution principle,Interface segregation principle, Dependency Inversion principle.
- Simple Object principle, Object and Clear principle, Liskov change principle ,Interface intermediate principle, Dependency injection principle.

✗ **Incorrecto** 0/3 Puntos

6. Cuando se tiene un condicional complejo (muchos if-else) es posible que NO se esté cumpliendo:

- Single Responsability
- Open/Closed
- Dependency Inversion
- Interface Segregation

✓ **Correcto** 3/3 Puntos

7. ¿Cuál es el principio que busca desacoplar o eliminar dependencias?

- Interface Segragation
- Dependency Inversion
- Inversion of Control

✓ **Correcto** 3/3 Puntos

8. El Principio de responsabilidad única establece que:

- Debemos tener un objetivo, la clase o función solo debe hacer una o más de una cosa a la vez.
- Debemos tener un objetivo, la clase o función establece muchas funcionalidades a la vez.
- Debemos tener un objetivo, la clase o función solo debe hacer exactamente una cosa.

✓ **Correcto** 3/3 Puntos

9. Principio de abierto cerrado (Open and Close principle) establece que:

- Los objetos o entidades deben estar cerrados para la extensión, pero abiertos para la modificación.
- Los objetos o entidades deben estar abiertos para la modificación, pero cerrados para la extensión.
- Los objetos o entidades deben estar abiertos para la extensión, pero cerrados para la modificación.

✓ **Correcto** 3/3 Puntos

10. El principio DRY dice que:

- No se deben usar métodos de otras clases
- No se deben tener atributos que se llamen igual en clases diferentes
- No se debe tener lógica repetida en el código

3. Patrones de diseño

✓ **Correcto** 3/3 Puntos

11. ¿Como se clasifican los patrones de diseño? *

- ( ) Estado, Comportamiento, Multi Capa
- ( ) Rápido, Independiente, Repetible
- (•) Comportamiento, Estructura, Creación

✗ **Incorrecto** 0/3 Puntos

12. ¿Si tengo una implementación que está verificando el estado y mensajes que entran a una clase, que tipo de patrón se está utilizando? *

- ( ) Comportamiento
- (•) Observer
- ( ) Mediator

✓ **Correcto** 3/3 Puntos

13. Un patrón de diseño GoF es: *

- ( ) Un algoritmo usado en programación orientada a objetos.
- (•) Una solución a un problema común en programación orientada a objetos.
- ( ) Una estructura de datos usada en programación orientada a objetos
- ( ) Un modelo para un tipo particular de clases.

✗ **Incorrecto** 0/3 Puntos

14. ¿Cuáles de las siguientes afirmaciones son un beneficio de los patrones de diseño GoF? (Seleccione las que aplique) *

- [ ] Analizar el código es más rápido
- [x] Modificar el código es más fácil
- [ ] Codificar el programa es fácil
- [ ] Mas rapidez en las pruebas

✗ **Incorrecto** 0/3 Puntos

15. Los patrones estructurales son útiles para: *

- ( ) Abstraer el proceso de creación de objetos
- (•) Asignar responsabilidades a las clases
- ( ) Combinar clases para crear estructuras complejas.

✓ **Correcto** 3/3 Puntos

16. Cuando se desea crear productos concretos, sin tener muchos condicionales en el código, puede usar: *

- ( ) Sigleton
- (•) Factory Method
- ( ) Observer
- ( ) Builder

4. Pruebas Unitarias.

✓ **Correcto** 3/3 Puntos

17. ¿Cuales son los principios de las pruebas unitarias?

- Prueba primero
- Red/Green/Refactor
- Arrange - Action - Assert
- Rápida, Independiente, Repetible, Auto-Validada, A tiempo

✓ **Correcto** 3/3 Puntos

18. ¿Cuál es el objetivo de la prueba unitaria?

- Busca asegurar que el código funciona de acuerdo con las especificaciones
- Encontrar el error de lo que se está ejecutando
- Evitar errores y generar calidad

✓ **Correcto** 3/3 Puntos

19. **¿Cuál tipo de prueba de hace después de las pruebas unitarias?**

- Pruebas de sistema
- Pruebas de integración
- Ninguna de las anteriores

✓ **Correcto** 3/3 Puntos

20. En las Pruebas Unitarias: ¿Qué es lo que se debe hacer por cada Unidad?

- Verificar que los módulos de nivel inferior llaman a los de nivel superior
- Definir los casos de prueba (pruebas de caja blanca)
- Decidir qué acciones tomar cuando se descubran problemas.

✓ **Correcto** 3/3 Puntos

21. ¿Cuál frase describe mejor las pruebas unitarias?

- Probar que todos los elementos funcionan juntos
- Probar un sistema integrado en su totalidad
- Probar métodos y funciones de manera individual
- Probar si se satisfacen los criterios de aceptación

✓ **Correcto** 3/3 Puntos

22. ¿Cuál es el patrón de ordenamiento de las pruebas unitarias?

- Declaración, Validación, Documentación
- Diseño, Codificación, Prueba
- Declaración, Validación, Ejecución
- Declaración, Ejecución, Validación

5. TDD

✓ **Correcto** 2/2 Puntos

23. ¿En qué consiste la parte Green de TDD?

- (●) Escribir el mínimo código para que el test pase.
- ( ) Escribir más tests
- ( ) Escribir el feature de los Gherkin

✓ **Correcto** 4/4 Puntos

24. ¿Cuál de las siguientes afirmaciones es correcta, de acuerdo a los beneficios obtenidos con TDD? (Seleccione las que aplique)

- [x] Minimiza el tiempo de desarrollo
- [x] Evita el desperdicio
- [ ] Genera una guía de pruebas para el usuario final
- [x] Asegura la implementación de los escenarios de negocio.
- [ ] Dificulta el refactoring

✓ **Correcto** 4/4 Puntos

25. ¿En qué consiste la parte Red de TDD?

- ( ) Escribir un test que no funciona, por compilación
- ( ) Escribir Gherkin
- (●) Escribir un test que no funciona, por validación
- ( ) Escrbir código de lógica de negocio

Volver a la página de agradecimiento