Cuestionario: Diseño y realización de pruebas de software

1. ¿Cuál de las siguientes afirmaciones sobre las pruebas de software es correcta?

   a. Las pruebas de software garantizan que no hay errores en el código.
   b. Las pruebas de software solo son necesarias en proyectos grandes.
   c. Las pruebas ayudan a detectar defectos, pero no pueden garantizar un software 100% libre de errores.
   d. Solo los testers deben realizar pruebas, los desarrolladores no.

2. ¿Qué tipo de prueba se basa en la ejecución del código para detectar defectos?

   a. Pruebas estáticas
   b. Pruebas dinámicas
   c. Análisis de código
   d. Inspecciones

3. ¿Cuál de las siguientes opciones describe una prueba de caja negra?

   a. Se basa en la estructura interna del código fuente.
   b. Solo prueba unidades individuales del sistema.
   c. Evalúa la funcionalidad sin conocer la implementación interna.
   d. Se centra en la cobertura del código.

4. ¿Qué técnica de prueba se enfoca en evaluar caminos de ejecución del código?

   a. Prueba de equivalencia
   b. Prueba basada en experiencia
   c. Prueba de caja blanca
   d. Prueba de regresión

5. ¿Cuál de las siguientes no es una ventaja de las pruebas unitarias?

   a. Detectan errores en la integración de múltiples módulos.
   b. Facilitan la detección temprana de fallos en funciones individuales.
   c. Mejoran la mantenibilidad del código.
   d. Ayudan a evitar la propagación de errores a niveles superiores.

6. ¿Qué tipo de prueba se realiza antes de la entrega final para validar que cumple con los requisitos del cliente?

a. Pruebas de integración
b. Pruebas de aceptación
c. Pruebas unitarias
d. Pruebas de caja blanca

7. ¿Qué técnica de prueba de caja negra divide los datos de entrada en grupos que deben comportarse de manera similar?

   a. Análisis de valores límite
   b. Clases de equivalencia
   c. Cobertura de decisiones
   d. Pruebas de integración

8. ¿Cuál de los siguientes tipos de prueba se enfoca en verificar la interacción entre diferentes módulos del sistema?

   a. Pruebas unitarias
   b. Pruebas de integración
   c. Pruebas de regresión
   d. Pruebas de caja blanca

9. ¿Qué herramienta se usa comúnmente en Java para realizar pruebas unitarias?

   a. Selenium
   b. JUnit
   c. Postman
   d. Git

10. ¿Qué técnica de caja blanca mide el número de caminos de ejecución en un código?

    a. Pruebas de partición de equivalencia
    b. Análisis de valores límite
    c. Complejidad ciclomática
    d. Pruebas de regresión

11. ¿Qué documento describe el conjunto de pruebas a realizar, los criterios de aceptación y los recursos necesarios?

    a. Plan de pruebas
    b. Informe de defectos
    c. Especificación de requisitos
    d. Caso de uso

12. ¿Cuál de los siguientes es un beneficio de la automatización de pruebas?

    a. Reduce el número total de pruebas necesarias.
    b. Permite ejecutar pruebas repetitivas de forma rápida y eficiente.
    c. Elimina la necesidad de pruebas manuales.
    d. Solo se usa en entornos ágiles.

13. ¿Qué tipo de prueba se enfoca en asegurar que un cambio en el código no ha afectado funcionalidades previas?

    a. Pruebas de integración
    b. Pruebas de aceptación
    c. Pruebas unitarias
    d. Pruebas de regresión

14. ¿Cuál de las siguientes opciones describe mejor las pruebas exploratorias?

    a. Son pruebas completamente automatizadas.
    b. No siguen un plan predefinido y dependen de la experiencia del tester.
    c. Solo se usan en sistemas críticos.
    d. No requieren interacción humana.

15. ¿Qué tipo de prueba se realiza sin ejecutar el código?

    a. Pruebas estáticas
    b. Pruebas de integración
    c. Pruebas funcionales
    d. Pruebas de caja blanca

16. ¿Qué técnica de prueba evalúa los valores máximos y mínimos de un sistema?

    a. Partición de equivalencia
    b. Análisis de valores límite
    c. Pruebas de integración
    d. Pruebas de regresión

17. En una prueba de caja blanca, ¿qué criterio asegura que cada instrucción del código se ejecuta al menos una vez?

    a. Cobertura de sentencias
    b. Cobertura de condiciones
    c. Cobertura de partición de datos
    d. Pruebas funcionales

18. ¿Cuál de los siguientes es un criterio de finalización de pruebas?

    a. Se han ejecutado al menos 10 casos de prueba.
    b. No se han encontrado más errores en la última prueba.
    c. Se ha alcanzado un nivel de cobertura y los defectos encontrados han sido corregidos.
    d. El cliente ha solicitado detener las pruebas.

19. ¿Cuál de los siguientes niveles de prueba se realiza primero en un ciclo de vida de desarrollo tradicional?

    a. Pruebas unitarias
    b. Pruebas de integración
    c. Pruebas de sistema
    d. Pruebas de aceptación

20. ¿Qué técnica de prueba permite evaluar el comportamiento del software sin conocimiento previo del código?

    a. Pruebas de caja blanca
    b. Pruebas de caja negra
    c. Análisis estático
    d. Pruebas de cobertura

21. ¿Qué es JUnit en el contexto del desarrollo en Java?

    a. Un compilador de Java.
    b. Un framework para la ejecución de código en producción.
    c. Un framework para escribir y ejecutar pruebas unitarias en Java.
    d. Un depurador de código para encontrar errores.

22. ¿Qué anotación en JUnit indica que un método es una prueba?

    a. `@Test`
    b. `@JUnit`
    c. `@UnitTest`
    d. `@Check`

23. ¿Cuál de las siguientes afirmaciones sobre JUnit es correcta?

    a. JUnit solo se puede usar con Maven.
    b. JUnit requiere que todas las pruebas sean ejecutadas manualmente.

c. JUnit permite la automatización de pruebas en Java.
d. JUnit no admite la ejecución de múltiples pruebas en paralelo.

24. ¿Qué método de JUnit se usa para verificar que dos valores son iguales?

    a. `assertTrue()`
    b. `assertEquals()`
    c. `assertSame()`
    d. `assertIdentical()`

25. ¿Qué ocurre si una prueba en JUnit falla?

    a. El programa se detiene automáticamente.
    b. Se genera un error en la ejecución de las pruebas, pero el programa sigue corriendo.
    c. Se reinicia la máquina virtual de Java.
    d. Se ignora la prueba fallida.

26. ¿Cuál es el propósito de la anotación `@BeforeEach` en JUnit 5?

    a. Ejecutar una prueba después de cada caso de prueba.
    b. Ejecutar código antes de cada prueba individual.
    c. Ignorar una prueba específica.
    d. Indicar que una prueba es opcional.

27. ¿Cómo se puede agrupar varias pruebas en JUnit 5?

    a. Con la anotación `@TestGroup`.
    b. Usando `@Nested` o `@Tag`.
    c. Creando un archivo `. junit`.
    d. Definiendo una clase estática con pruebas.

28. ¿Qué anotación en JUnit permite ignorar una prueba temporalmente?

    a. `@SkipTest`
    b. `@Disabled`
    c. `@Ignore`
    d. `@Inactive`

29. ¿Qué significa cuando una prueba de JUnit lanza una excepción?

    a. La prueba siempre se considera exitosa.
    b. La prueba falla a menos que la excepción esperada haya sido definida en la

anotación `@Test(expected = . . . )`.

c. Se detiene la ejecución de todas las demás pruebas.

d. Se ejecuta nuevamente la prueba.

30. ¿Qué método de aserción en JUnit verifica si una condición booleana es verdadera?

a. `assertFalse()`

b. `assertTrue()`

c. `assertNull()`

d. `assertSame()`