1.- Relaciona cada herramienta tipo de ejecución con su función correspondiente.
Paso a paso por instrucción → Ejecuta el programa línea a línea
Paso a paso por procedimiento → Ejecuta el programa línea a línea, pero sin entrar en el código de las funciones
Ejecución hasta instrucción → Ejecuta el código hasta una instrucción, y a partir de ella, depuramos
Ejecución hasta el final → Ejecuta el resto del código, sin pararse en ninguna instrucción

2.- Relaciona las herramientas de automatización con su lenguaje.
SimpleTest → PHP
JUnit → Java
CppUnit → C++
FoxUnit → Microsoft Visual FoxPro.

3.- ¿Cómo se denomina el concepto de aplicar el mismo nombre a comportamientos diferentes?
Polimorfismo

4.- ¿Cómo se denomina la propiedad que permite subdividir una aplicación en partes más pequeñas siendo cada una ellas tan independiente como sea posible de la aplicación en sí y de las restantes partes?
Modularidad

5.- ¿Cómo se llama la prueba que comprueba el cumplimiento de los requisitos funcionales?
Validación.

6.- ¿En qué momento se suelen realizar las pruebas de la unidad?
Antes de las pruebas de integración.

7.- ¿Qué componente del IDE es básico en la realización de pruebas?
Depurador.

8.- ¿Qué herramienta de automatización de pruebas no es para Java?
FoxUnit.

9 .-A la relación que se establece entre objetos en los que unos utilizan las propiedades y comportamientos de otros formando una jerarquía se le denomina:
Herencia

10.- Con las clases de equivalencia diseñamos casos de pruebas:
Con valores representativos del rango admitido.

11.- Con las clases de equivalencia no válidas diseñamos casos de pruebas:
Con valores fuera del rango admitido.

12.- Con las clases de equivalencia válidas diseñamos casos de pruebas:
Con valores representativos del rango admitido.

13.- El estándar ISO/IEC 29119 proporciona _____para cubrir todas las fases de la prueba.
vocabulario, **documentación**, **procesos**

14.- El objetivo del cubrimiento en las pruebas de caja blanca es:
Comprobar que todos los caminos se pueden ejecutar.

15.- El objetivo del cubrimiento:
Comprobar que todos los caminos se pueden ejecutar.

16.- En la planificación de pruebas.
Se diseñan los tipos de prueba y los casos de prueba.

17.- En las pruebas estructurales.
Se comprueba la cobertura de decisiones
Se comprueba la cobertura de sentencias
Se comprueba la cobertura de caminos.

18.- La herramienta de prueba unitaria más extendida en Java es:
El Junit.

19.- La inspección de variables.
Permiten ver la evolución de los valores de las variables
Permiten definir que variables se van inspeccionar.

20.- La prueba de software.
Sirve para verificar y validar el sistema.

21.- La prueba de software.
Sirve para verificar y validar el sistema.

22.- La realización de pruebas _____ nos permite detectar errores de cada parte del programa por separado.
unitarias

23.- La regresión es:
un proceso que se realiza cuando se produce un cambio en el código.

24.- Las clases de equivalencia.
Nos permite crear casos de prueba representativos de un conjunto de valores posibles.

25.- Las funciones básicas del depurador son:
Localizar errores en la implementación
Controlar los valores que toman los datos
Verificar el flujo de ejecución.

26.- Las herramientas de automatización de pruebas.
Generan casos de prueba
Muestran los resultados de ejecución de los casos de prueba
Nos permiten controlar la regresión.

27.- Señala las pruebas funcionales.
Particiones equivalentes
Valores límite.

28.- Si tenemos el bucle while ((x>5)&&(x<10)) , podrían ser valores límite _____ para probar los valores límite.
x igual a seis.

29.- Si tenemos el bucle while ((x>5)) , podrían ser valores límite _____ para probar el valor límite de la clase válida.
x igual a 5.

30.- Son ejemplos de herramientas para cubrimiento:
JJPath
Clover.

31.- Son tipos de pruebas.
Funcionales
Estructurales
Regresión.

32.- Son ventajas de la prueba de la unidad:
Simplifican la integración
Documentan el código
Separación de la interfaz y la implementación.

33.- Un caso de prueba.
Se diseña intentando que la probabilidad de detección de errores sea máxima.

34 .-¿Qué documentos deben elaborarse?
Plan de pruebas.

35.- El cubrimiento es un tipo de prueba de caja blanca.
«Verdadeiro»

36.- El metodología de documentación es Métrica v.3. ¿Verdadero o falso?
«Falso»

37.- La fase de prueba no es necesario documentarla.
«Falso»

38.- Los estándares de normalización de prueba BSI cubren todas las fases de la prueba. ¿Verdadero o falso?
«Falso»

Unidad 04

1.- Relaciona cada patrón de refactorización con su función correspondiente.
Cambia el nombre de un paquete, clase, método o campo. → Renombrar
Traslada una clase de un paquete a otro, sin duplicar código. → Mover la clase
Crear métodos getter y setter para acceder a los campos de una clase. → Campos encapsulados., Sustituye un bloque de código por un método. → Insertar método.

2.- Relaciona cada tag Javadoc con su función.
@see → Referencia a otras clases y métodos
@version → Versión y fecha de la clase
@return → Valor devuelto por un método
@author → Autor o autora de la clase.

3.- ¿Cómo se llama el almacén de versiones de CVS?
Repositorio.

4.- ¿Cómo se llama el almacén de versiones de GIT? (la respuesta incorrecta resta 1/3)
Repositorio.

5.- ¿Cuál no es un patrón de refactorización?
Análisis de código.

6.- ¿Qué afirmaciones sobre control de versiones es correcta?
Pueden existir varias versiones de una clase
El almacenamiento de versiones es centralizado.

7.- ¿Qué documento produce Javadoc?
Genera un archivo HTML con la información de las clases y métodos.

8.- ¿Qué herramienta de automatización de documentación usa Java?
A resposta correcta é: Javadoc.

9.- ¿Qué patrón de refactorización se utiliza para crear métodos setter y getter?
Campos encapsulados.

10.- ¿Qué tarea no forma parte de la Gestión de Configuraciones de Software?
Gestión del repositorio.

11.- El concepto de entrega hace referencia a:
una instancia de un sistema que se distribuye a usuarios externos al equipo de desarrollo.

12.- El desarrollo guiado por pruebas (TDDTest Driven Development):
refactoriza el código al mismo tiempo que la pruebas., implica diseñar las pruebas al mismo tiempo que el software., implica que el diseño del código vaya en función de las pruebas.

13.- En CVS la orden que almacena la copia modificada en el repositorio es:
commit.

14.- En GIT la orden que almacena la copia del área de trabajo en el repositorio local es:
commit.

15.- En la Gestión del Cambio se establecen los siguientes tipos de control:
control individual
control de gestión u organizado
control formal.

16.- La documentación nos permite...
Explicar la finalidad de una clase
Explicar el funcionamiento de un método
Facilitar el trabajo de mantenimiento del software

17.- La Gestión de Configuraciones de Software se compone de:
control de cambios
auditorías de configuraciones
generación de informes.

18.- La herramienta de control de versiones en Visual Studio es:
Team Foundation Server.

19.- La refactorización:
utiliza una serie de patrones de aplicación sobre el código fuente.

20.- La refactorización:
modifica el diseño del código pero no su comportamiento.

21.- La refactorización:
modifica el diseño del código pero no su comportamiento.

22.- Los tipos de comentarios admitidos en Java son:
comentarios de una línea
comentarios multilínea
comentarios estilo Javadoc.

23.- Señala la herramienta analizadora de código.
PMD.

24.- Señala los analizadores de código.
PMD
FindBugs.

25.- Son ejemplos de herramientas CASE para gestión de configuraciones:
Bugzilla
Rational.

26.- Son herramientas automatizadas de documentación:
SchemeSpy
Javadoc
DoxyGen

27.- Son herramientas de control de versiones:
CVS
Subversion
Mercurial

28.- Son patrones de refactorización:
Encapsular campos
Renombrado

29.- Un comentario en formato JavaDoc.
Comienza por / y terminan por */**

30.- Una versión:
es la evolución de un único elemento, dentro de un sistema en desarrollo

31.- El repositorio es un almacén centralizado de versiones ¿Verdadero o falso?
«Verdadeiro»

32.- La planificación de la Gestión de Configuración del Software, es regulado por un estándar **IEEE.**
«Verdadeiro»

33.- La refactorización cambia en comportamiento del software. ¿Verdadero o falso?
«Falso»

34.- La refactorización no es necesario documentarla. ¿Verdadero o falso?
«Falso»