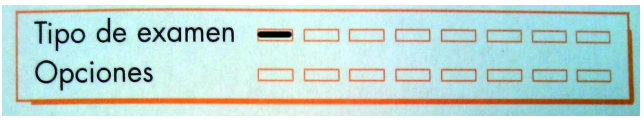

Sólo existe una respuesta acertada. Los errores restan un tercio de lo que suman los aciertos.

Orientación a objetos

1. Unified Modeling Language (UML)
    a. es un lenguaje de programación
    b. es un lenguaje de programación orientado a objetos
    c. es una metodología
    d. es un lenguaje de modelado de sistemas informáticos
2. En una relación de tipo asociación, si en uno de los extremos no se indica multiplicidad, esto significa que
    a. no interviene ningún objeto de esa clase
    b. interviene únicamente un objeto de esa clase
    c. puede intervenir un objeto de la clase (también es posible que no intervenga)
    d. intervienen muchos objetos de la clase
3. Sobre el role
    a. un objeto de una clase puede tener diferentes roles, tantos como relaciones en las que intervenga
    b. un objeto de una clase sólo tiene un posible role independientemente de las relaciones en las que intervenga
    c. cada objeto de una clase puede tener un role diferente en una misma relación
    d. ninguna de las anteriores
4. Sobre la multiplicidad
    a. en cualquier tipo de relación tiene sentido indicar la multiplicidad
    b. sólo en asociación se puede indicar la multiplicidad, en los otros tipos de relaciones no tiene sentido
    c. tanto en asociación como en agregación es posible indicar la multiplicidad, pero en generalización no tiene sentido
    d. ninguna de las anteriores
5. Cual de las siguientes afirmaciones es FALSA con respecto a la composición:
    a. es un tipo de agregación
    b. la vida de las partes no va más allá de la vida del todo
    c. las partes sólo pueden pertenecer a un todo
    d. la multiplicidad del todo puede ser más de 1
6. Las subclases heredan
    a. forma, comportamiento y relaciones
    b. forma y comportamiento, pero no relaciones
    c. forma, pero no comportamiento ni relaciones
    d. comportamiento, pero ni forma ni relacione

7. Una clase abstracta es aquella que
    a. implementa algún método abstracto
    b. todos sus métodos son abstractos
    c. no se puede instanciar (al ser un nivel de abstracción demasiado elevado no tiene sentido semánticamente)
    d. ninguna de las anteriores
8. En un interface UML (la pregunta no se refiere a un interface Java)
    a. todas las operaciones son abstractas y no puede tener atributos
    b. todas las operaciones son abstractas y puede tener atributos
    c. al menos una operación es abstracta y puede tener atributos
    d. al menos una operación es abstracta y no puede tener atributos
9. Sobreescritura es el mecanismo que se produce cuando se implementa en una subclase...
    a. un método abstracto de alguna superclase (por lo tanto no implementado en ninguna superclase)
    b. un método ya implementado en alguna superclase (el nivel de abstracción en la subclase permite implementarlo de forma más eficiente)
    c. un método ya implementado en alguna superclase para cambiar la semántica
    d. un método definido en un interface que implementa la subclase
10. Tenemos los siguientes elementos en nuestro modelo: el interface Redimensionable que define el servicio "redimensionar", la clase Figura que implementa el interface Redimensionable y que presenta un método abstracto "área" y, finalmente, la clase Elipse que hereda de Figura. Cual de las siguientes afirmaciones es FALSA:
    a. Figura debe ser abstracta ya que tiene un método abstracto
    b. Elipse, para ser concreta ha de implementar "área" y "redimensionar"
    c. Elipse puede ser concreta si implementa "área" ya que hereda directamente de Figura
    d. Si Elipse implementa sólo "área" debería ser una clase abstracta ya que no implementa "redimensionar"

Java

11. Dado el array “datos”, la sentencia

    System.arraycopy(datos, i + 1, datos, i, datos(length – i – 1));

    a. produce un error de compilación
    b. desplaza varios elementos del array a la posición anterior a la que ocupan
    c. desplaza el elemento i del array a la posición i+1
    d. desplaza el elemento i+1 del array a la posición i

12. La sentencia

ArrayList<CuentaBanco> usuarios = new ArrayList<CuentaBanco>();

    a. crea un ArrayList de *usuarios*
    b. declara pero no inicializa un ArrayList de *CuentaBanco*
    c. inicializa pero no declara un ArrayList de *CuentaBanco*
    d. ninguna de las anteriores

13. Una operación aritmética entre un entero, a, y un número en coma flotante, b, produce como resultado
    a. el truncado de la parte fraccional de b y la ejecución de la operación
    b. un casting expreso
    c. un número en coma flotante
    d. la reducción de a al módulo del rango y la ejecución de la operación

14. La asignación: Double d = 10;
    a. asigna el valor 10 a la variable del tipo primitivo d
    b. asigna el valor 10 a la variable del tipo primitivo Double
    c. es equivalente a: Double d = new double(10);
    d. ninguna de las anteriores

15. En relación a los constructores, es FALSO que
    a. una clase puede tener varios constructores
    b. se invocan al instanciar un objeto
    c. devuelven un valor
    d. el programador no siempre tiene por qué implementarlos

16. En relación a los constructores por defecto en clases heredadas, es FALSA la afirmación
    a. una clase puede tener varios constructores por defecto
    b. pueden llamar automáticamente a otros constructores
    c. pueden llamar a otros constructores
    d. ninguna de las anteriores

17. La sintaxis del método: *void nada () { return; }*
    a. Es correcta
    b. Es incorrecta porque en el cuerpo no se especifica el valor de retorno
    c. Es incorrecta porque no se especifican parámetros
    d. Es incorrecta porque no se especifica el calificador de acceso

18. Un método recibe como parámetro un objeto con dos atributos, uno atributo del tipo *int* y otro del tipo *Int*. Si el método altera el contenido del primero es cierto que el cambio
    a. es permanente porque el parámetro se pasa por valor
    b. no es permanente porque el parámetro se pasa por valor
    c. es permanente porque el parámetro se pasa por referencia
    d. no permanente porque el parámetro se pasa por referencia

19. Los cambios en un atributo declarado como *static* a través de una instancia
    a. da un error. No se pueden realizar cambios en un atributo *static*
    b. da un error. No se pueden realizar cambios en un atributo static a través de una instancia
    c. afecta sólo a la instancia
    d. afectan a todas las instancias
20. Un método de clase sin parámetros
    a. no puede ser llamado desde otras clases
    b. es de instancia
    c. puede devolver un objeto
    d. tiene acceso a los atributos de una instancia
21. Por defecto, los métodos especificados en una interfaz son
    a. public
    b. private
    c. protected
    d. Package-private
22. Por defecto, los atributos de las clases son
    a. public
    b. private
    c. protected
    d. package-private
23. La sobrecarga se aplicaría si contásemos con dos métodos con
    a. distinto valor de retorno y diferentes parámetros
    b. mismo valor de retorno y mismos parámetros
    c. mismo identificador y diferentes parámetros
    d. distinto identificador mismos parámetros
24. En relación a las relaciones de composición es cierto que
    a. los datos primitivos se insertan mediante referencias
    b. la inicialización de las referencias se hace en el momento declaración
    c. Java permite composición con múltiples clases
    d. requieren la una llamada al constructor por defecto
25. En relación a las relaciones de herencia
    a. es posible seleccionar los métodos a heredar
    b. son compatibles con la sobrecarga de métodos heredados
    c. el acceso a los atributos heredados se hace mediante *super*
    d. la clase derivada debe reimplementar los métodos de la clase base
26. Determine cuál de las siguientes afirmaciones sobre las interfaces es FALSA
    a. pueden definir constantes
    b. es un tipo de clase con métodos sin implementación
    c. exigen la implementación obligatoria en derivadas instanciables
    d. sus métodos pueden ser implementados de forma distinta en varias clases

27. Sobre las clases abstractas, es cierto que
    a. tienen métodos implementados
    b. los métodos abstractos admiten distintas implementaciones
    c. no se puede derivar una clase abstracta de otra case abstracta
    d. no admiten atributos que no sean constantes

28. En relación a la herencia, las interfaces y las clases abstractas (CA) es cierto que:
    a. Java implementa la herencia múltiple a través de la herencia de varias interfaces
    b. Una clase puede heredar un máximo de una interfaz, pero puede implementar varias CA
    c. Desde el punto de vista de la eficiencia, las CA son más rápidas
    d. ninguna de las anteriores

29. La clase MiClase hereda el método *void esPesado() {System.out.println (" Pesado ");}* e implementa una interfaz que contiene la definición *void esPesado(){System.out.println (" Pesado ");}*. En esas circunstancias
    a. error de compilación por colisionar en *MiClase* los métodos *esPesado()*
    b. prevalece el primer método en *MiClase* por ser heredado
    c. prevalece el segundo método en *MiClase* por ser implementado
    d. ninguna de las anteriores

30. Sobre el polimorfismo
    a. realiza una conversión hacia abajo
    b. aporta reductibilidad ligada dinámicamente
    c. permite seleccionar el método adecuado en tiempo de ejecución
    d. todas las anteriores