



Anexo 2

Pruebas para la obtención de títulos de Técnico y Técnico Superior

Convocatoria correspondiente al curso académico 2023-2024

(Resolución de 29 de diciembre de 2023 de la Dirección General de Educación Secundaria, Formación Profesional y Régimen Especial)

DATOS DEL ASPIRANTE			FIRMA
Apellidos:			
Nombre:	D.N.I. N.I.E. o Pasaporte:	Fecha:	

Código del ciclo:	Denominación completa del título:
IFCS02	Técnico Superior en Desarrollo de Aplicaciones Multiplataforma
Clave o código del módulo:	Denominación completa del módulo profesional:
485A	Programación

INSTRUCCIONES GENERALES PARA LA REALIZACIÓN DE LA PRUEBA
<ul style="list-style-type: none">El examen tendrá una duración de 2 horas.La prueba consta de un examen tipo test con cuatro opciones de las cuales solamente una es correcta.Cada pregunta se responderá en el espacio dejado al efecto en la hoja de respuestas. Se rellenarán los recuadros para señalar la respuesta seleccionada.Si se quiere rectificar una respuesta contestada, se borrará toda la casilla de la respuesta incorrecta con Tipp-Ex o corrector, tal y como se puede apreciar en el siguiente ejemplo:<ul style="list-style-type: none">Se elimina la selección de b para seleccionar la opción d: <input type="checkbox"/>a <input type="checkbox"/>b <input type="checkbox"/>c <input checked="" type="checkbox"/>dSe dispondrá de una hoja para borrador (o de varias si se requieren), que será proporcionada por el centro. Esa hoja se entregará obligatoriamente al final junto con el examen, si bien nada de lo escrito en la hoja de borrador se valorará en la corrección.Sólo se utilizará bolígrafo negro o azul y Tipp-Ex o corrector, no permitiéndose usar bolígrafo rojo, lapicero, etcétera.No se podrá emplear ningún dispositivo electrónico.Cualquier tachadura o borrón en una respuesta podrá invalidar toda la puntuación de la pregunta asociada.

CRITERIOS DE CALIFICACIÓN Y VALORACIÓN
<ul style="list-style-type: none">El test se calificará sobre 10 puntos. Todas las preguntas se calificarán equitativamente con la misma cantidad de puntos. En cada pregunta se plantearán varias respuestas, y se deberá señalar la única que se considere correcta, según el caso. Cada respuesta correcta que se marque se valorará con 0,25 puntos, y si se marca alguna incorrecta, se valorará con una cantidad negativa equivalente a 0,075 puntos, es decir, se descontarán 0,075 puntos. Si no se está seguro de si una respuesta es correcta o no, y no se marca, no sumará ni restará puntos.Calificación final del módulo profesional:<ul style="list-style-type: none">El alumno obtendrá en el módulo profesional una calificación entera entre 1 y 10.Dicha calificación se calculará:<ul style="list-style-type: none">Si la calificación conseguida en la prueba es inferior a 5: se truncará dicha calificación.Si la calificación conseguida en la prueba es igual o superior a 5 y los decimales:<ul style="list-style-type: none">Son inferiores a 0'5: se redondeará al entero inferior más próximo.Son iguales o superiores a 0'5: se redondeará al entero superior más cercano.La anterior regla tiene una excepción: las notas de examen inferiores a 1 se redondearán a 1.



CONTENIDO DE LA PRUEBA

- 1) ¿Cuál de las siguientes opciones describe correctamente la relación entre estructuras de control y bloques de código en la programación?**
 - a. Las estructuras de control no pueden contener bloques de código, sólo declaraciones simples.
 - b. Un bloque de código es un tipo de estructura de control que define el flujo de ejecución del programa.
 - c. Las estructuras de control utilizan bloques de código para agrupar múltiples instrucciones, permitiendo la ejecución condicional o repetitiva de éstas.
 - d. Los bloques de código y las estructuras de control son independientes y no interactúan entre sí.

- 2) ¿Cuál de las siguientes afirmaciones es verdadera respecto a las conversiones de tipo en lenguajes de programación como Java?**
 - a. Las conversiones implícitas siempre requieren sintaxis especial para ser realizadas.
 - b. Las conversiones explícitas (casting) se realizan automáticamente cuando se asigna un valor de un tipo de menor capacidad a un tipo de mayor capacidad.
 - c. Los comentarios en el código pueden afectar el resultado de las conversiones de tipo al ser compilado.
 - d. Una conversión explícita (casting) es necesaria cuando se asigna un valor de un tipo de dato de mayor capacidad a un tipo de menor capacidad para evitar la pérdida de información.

- 3) En el contexto de la gestión de memoria en Java, ¿qué mecanismo asegura que los objetos no utilizados sean eliminados de la memoria?**
 - a. Recolección de basura, que automáticamente encuentra y libera memoria ocupada por objetos que ya no son accesibles.
 - b. Destructor explícito, que debe ser llamado manualmente para liberar objetos de la memoria.
 - c. Sobrecarga del operador delete, utilizado para eliminar objetos y liberar recursos.
 - d. Referencias nulas, que automáticamente eliminan objetos de la memoria cuando su cuenta de referencias llega a cero.

- 4) Si una clase Calculadora tiene un método estático sumar, ¿cuál de las siguientes afirmaciones es correcta?**
 - a. sumar sólo puede ser llamado después de instanciar un objeto de la clase Calculadora.
 - b. sumar puede ser llamado sin necesidad de crear un objeto de la clase Calculadora.
 - c. Para usar sumar, primero se debe declarar una propiedad estática dentro de Calculadora.
 - d. sumar cambia el estado de los objetos Calculadora cuando se llama.

- 5) Considerando las estructuras de selección en Java, ¿cuál de las siguientes afirmaciones es incorrecta acerca del uso de switch?**
 - a. switch puede usarse con tipos enumerados (enum) para facilitar la legibilidad del código.
 - b. A partir de Java SE 14, switch puede ser utilizado como una expresión que devuelve un valor.
 - c. switch permite el uso de rangos de valores en cada caso (case) para agrupar múltiples condiciones similares. Por ejemplo: case 1..10 para los números del intervalo 1 a 10.
 - d. switch soporta tipos de datos String, permitiendo casos basados en cadenas de texto.



- 6) **Dada la necesidad de ejecutar un bloque de código al menos una vez y repetir su ejecución basada en una condición evaluada al final, ¿cuál estructura de repetición es la más adecuada?**
- Bucle for.
 - Bucle while.
 - Bucle do-while.
 - Bucle for-each.
- 7) **Considerando la sobrecarga de constructores en Java, ¿cuál de las siguientes afirmaciones describe mejor su propósito?**
- Facilitar la creación de objetos con diferentes estados iniciales, proporcionando varios constructores con diferentes parámetros.
 - Permitir la creación de múltiples constructores con el mismo nombre pero diferentes tipos de retorno.
 - Incrementar la eficiencia en tiempo de ejecución al reducir la cantidad de código necesario para instanciar objetos.
 - Obligar a que todos los objetos de una clase se creen con el mismo conjunto de atributos inicializados.
- 8) **¿Qué afirmación describe correctamente el efecto de aplicar el modificador de acceso private a un atributo de una clase en Java?**
- El atributo puede ser accedido directamente por cualquier clase dentro del mismo paquete.
 - El atributo no puede ser leído ni modificado directamente por ninguna otra clase, incluso si estas clases están en el mismo paquete.
 - La visibilidad del atributo se limita a clases que heredan de la clase donde se declara el atributo.
 - El atributo se convierte en accesible para todas las clases, facilitando la integración de sistemas.
- 9) **Si se desea almacenar, como atributo, el Ciclo Formativo en el que está matriculado un alumno y sus posibles valores son SMR, DAM, DAW y ASIR, ¿qué es más recomendable utilizar para almacenar dicha información y realizar posteriores filtrados por dicho atributo?**
- Un String con el nombre del ciclo formativo.
 - Un char con el nombre del ciclo formativo.
 - Un array booleano que tenga a verdadero la posición que corresponda al ciclo formativo en el que se halle matriculado el alumno.
 - Un enumerado.
- 10) **¿Cuál de las siguientes frases encaja mejor con el concepto de sobrecarga de métodos?**
- Implementar dos métodos en la misma clase que tienen el mismo nombre de método pero reciben distintos número de parámetros.
 - Implementar dos métodos en la misma clase que tienen diferente nombre pero sus parámetros son del mismo tipo.
 - Implementar dos métodos en clases distinta y que tienen el mismo nombre.
 - Ampliar la funcionalidad de un método heredando de él.



11) Considere el siguiente método definido en una clase. ¿Qué concepto de Programación Orientada a Objetos se está aplicando?

```
public void actualizarPrecio(double descuento) {  
    this.precio *= (1 - descuento);  
}
```

- a. Polimorfismo, porque el método puede aplicarse a objetos de diferentes tipos.
- b. Herencia, debido a que el método puede ser heredado por otras superclases.
- c. Abstracción, al ofrecer una operación compleja detrás de una simple llamada de método.
- d. Encapsulamiento, ya que modifica el estado interno del objeto de una manera controlada.

12) Si no se desea que un método se pueda sobrescribir, ¿qué palabra reservada hay que utilizar previamente a su nombre?

- a. void.
- b. close.
- c. final.
- d. static.

13) Dado el siguiente código de Java, ¿cuál será la salida por pantalla?

```
public static void main(String[] args) {  
  
    int[] calificaciones = new int[] {1, 3, 5, 8, 10};  
  
    for(int i=0; i<=calificaciones.length; i++) {  
        System.out.print("A");  
    }  
}
```

- a. AAAAA
- b. AAAAAA
- c. A
- d. AAAA

14) Dado el siguiente código de Java, ¿qué ocurrirá al ejecutar el método main?

```
public static void main(String[] args) {  
  
    int dividendo = 10;  
    int divisor = dividendo - 10;  
  
    System.out.println("Cociente -> " + dividendo / divisor);  
}
```

- a. Se mostrará el mensaje Cociente -> 0
- b. Se mostrará el mensaje Cociente -> Math.Infinity
- c. Se mostrará el mensaje Cociente -> 10
- d. Se producirá una excepción de tipo aritmético.



15) Dado el siguiente código de Java, ¿qué podemos afirmar sobre EDAD_MINIMA?

```
final int EDAD_MINIMA = 18;
```

- a. Es una constante.
- b. Se puede cambiar el valor de EDAD_MINIMA en tiempo de ejecución pero solamente para establecer otro valor numérico entero (nunca un valor numérico decimal).
- c. No se puede inicializar en la propia declaración.
- d. No se pueden utilizar guiones bajos en su declaración.

16) Dado el siguiente código de Java, si se desea optimizar la creación y concatenación de la cadena de texto, ¿qué clase es mejor utilizar en vez de String?

```
public static void main(String[] args) {  
  
    String cadena = new String();  
  
    for(int i=0; i<1000; i++) {  
        cadena += i + " ";  
    }  
  
    System.out.println(cadena);  
}
```

- a. StringLong.
- b. Character[].
- c. StringConcats.
- d. StringBuilder.

17) Dado el siguiente código de Java, ¿cuál será la salida por pantalla al ejecutarlo?

```
public static void main(String[] args) {  
  
    String texto = "Lorem ipsum";  
    int numero = texto.length();  
  
    if(numero % 2 == 0)  
        numero = numero / 2;  
    else  
        numero = numero / 4;  
  
    System.out.println(texto.charAt(numero));  
}
```

- a. r
- b. o
- c. m
- d. i



18) Dado el siguiente código de Java, ¿cuál será la salida por pantalla?

```
public static void main(String[] args) {  
    boolean duda = 5 > 5;  
  
    int numero = (duda  
        ? 1  
        : 0;  
  
    System.out.println(++numero + " y " + numero++);  
}
```

- a. 0 y 1
- b. 1 y 2
- c. 0 y 2
- d. 1 y 1

19) Dado el siguiente código de Java, ¿cuál será la salida por pantalla?

```
public static void main(String[] args) {  
  
    Set<String> exámenesPrevistos = new TreeSet<String>();  
    exámenesPrevistos.add("Programación");  
    exámenesPrevistos.add("Lenguaje de Marcas");  
    exámenesPrevistos.add("Bases de datos");  
    exámenesPrevistos.add("Programación");  
    exámenesPrevistos.add("Entornos de desarrollo");  
    exámenesPrevistos.add("Lenguaje de Marcas");  
  
    List<String> exámenes = new ArrayList<String>(exámenesPrevistos);  
  
    System.out.println(String.format("Se han planificado %d exámenes. El primero es %s.",  
        exámenes.size(),  
        exámenes.get(0)));  
}
```

- a. Se han planificado 4 exámenes. El primero es Programación.
- b. Se han planificado 6 exámenes. El primero es Programación.
- c. Se han planificado 4 exámenes. El primero es Bases de datos.
- d. Se han planificado 6 exámenes. El primero es Lenguaje de Marcas.

Teniendo en cuenta la siguiente clase Artículo, responda a las preguntas relacionadas con dicha clase.

```
public class Artículo {  
    private String nombre;  
    private int cantidad;  
    private int codigo;  
  
    public Artículo(String nombre, int cantidad, int codigo) {  
        this.nombre = nombre;  
        this.cantidad = cantidad;  
        this.codigo = codigo;  
    }  
  
    public String getNombre() { return nombre; }  
    public int getCantidad() { return cantidad; }  
    public int getCodigo() { return codigo; }  
    public void setCantidad(int cantidad) { this.cantidad = cantidad; }  
  
    @Override  
    public String toString() {  
        return nombre + ": " + cantidad;  
    }  
}
```



20) Si para gestionar el inventario de artículos, se declara el siguiente diccionario que relaciona el código del artículo (valor entero) con el propio artículo, ¿qué método permitiría obtener una lista con todos los objetos de tipo Artículo?

```
Map<Integer, Artículo> inventario = new HashMap<Integer, Artículo>();
```

- a. keySet().
- b. values().
- c. entrySet().
- d. getKeys().

21) ¿Cuál es la principal razón para elegir `HashMap<Integer, Artículo>` frente al uso de `ArrayList<Artículo>` a la hora de crear el anterior inventario?

- a. HashMap facilita la búsqueda directa de los artículos a través de su código.
- b. HashMap ordena automáticamente los artículos alfabéticamente por nombre (al ser el primer atributo declarado).
- c. HashMap previene cualquier duplicado de artículos basándose en el nombre del artículo.
- d. HashMap utiliza menos memoria que ArrayList para almacenar la misma cantidad de elementos porque optimiza la compartición de información.

22) Si se desea que, de forma natural, los artículos se ordenen por cantidad, ¿qué interfaz debería implementar la clase Artículo?

- a. java.lang.Sort.
- b. java.lang.Comparator.
- c. java.lang.Comparable.
- d. java.lang.Serializable.

Partiendo de las siguientes clases, responda a la siguiente pregunta.

```
public class ClaseA {  
  
    public ClaseA() {}  
  
    public String mensaje() {  
        return "Clase A";  
    }  
}
```

```
public class ClaseB extends ClaseA {  
  
    public ClaseB() {}  
  
    @Override  
    public String mensaje() {  
        return "Clase B";  
    }  
}
```

```
public class ClaseC extends ClaseB {  
  
    public ClaseC() {}  
  
    @Override  
    public String mensaje() {  
        return super.mensaje() + "-" + "Clase C";  
    }  
}
```

```
public class ClaseD extends ClaseC {  
  
    public ClaseD() {}  
  
    @Override  
    public String mensaje() {  
        return "Clase D";  
    }  
}
```



```
public class App {  
  
    public static void main(String[] args)  
    {  
        ClaseC claseC = new ClaseD();  
        System.out.println(claseC.mensaje());  
    }  
}
```

23) ¿Qué mensaje se mostrará por pantalla al ejecutar el método main?

- a. Clase C
- b. Clase B-Clase C
- c. Se produce una excepción y no se muestra ningún mensaje por pantalla.
- d. Clase D

24) Teniendo en cuenta el siguiente código fuente, ¿qué mensaje se mostrará por pantalla al ejecutar el método main?

```
public enum Color {  
    ROJO, VERDE, AZUL;  
}
```

```
public class EnumApp {  
  
    public static void main(String[] args) {  
  
        Color color = Color.AZUL;  
        String configuracion = new String("Color: ");  
  
        switch(color) {  
            case AZUL:  
                configuracion += "(0, 0, 255)";  
            case ROJO:  
                configuracion = "(255, 0, 0)";  
                break;  
  
            default:  
                configuracion += "(0, 255, 0)";  
        }  
  
        System.out.println(configuracion);  
    }  
}
```

- a. (255, 0, 0)
- b. Color: (0, 0, 255)(255, 0, 0)
- c. (0, 0, 255)
- d. Color: (0, 255, 0)



25) Dado el siguiente código en Java, ¿cuál es el propósito del constructor en la clase Auto?

```
public class Auto {  
    private String modelo;  
    private int anyo;  
  
    public Auto(String modelo, int anyo) {  
        this.modelo = modelo;  
        this.anyo = anyo;  
    }  
  
    public String getModelo() { return modelo; }  
  
    public int getAnyo() { return anyo; }  
}
```

- a. Inicializar las propiedades modelo y anyo de cualquier objeto Auto con valores predeterminados.
- b. Permitir la creación de objetos Auto sin especificar ningún detalle sobre el modelo o el anyo.
- c. Asegurar que cada objeto Auto se instancie con valores específicos para modelo y anyo.
- d. Ofrecer un método estático para crear instancias de Auto sin necesidad de utilizar el operador new.

Teniendo en cuentas las siguientes clases de Java, del mismo proyecto, las cuales tienen todos los paquetes necesarios importados, responda a las preguntas relacionadas con dichas clases.

```
public abstract class Empleado {  
    protected String nombre;  
    protected double salario;  
  
    protected Empleado(String nombre, double salario) {  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    public abstract void incrementarSalario(double porcentaje);  
}
```

```
public class Gerente extends Empleado {  
    private double bonificacion;  
  
    public Gerente(String nombre, double salario, double bonificacion) {  
        super(nombre, salario);  
        this.bonificacion = bonificacion;  
    }  
  
    @Override  
    public void incrementarSalario(double porcentaje) {  
        salario += salario * porcentaje / 100 + bonificacion;  
    }  
}
```

```
public interface Evaluable {  
    String evaluarDesempenyo();  
}
```



```
public class Ingeniero extends Empleado implements Evaluable {  
  
    public Ingeniero(String nombre, double salario) {  
        super(nombre, salario);  
    }  
  
    @Override  
    public void incrementarSalario(double porcentaje) {  
        salario += salario * porcentaje / 100;  
    }  
  
    @Override  
    public String evaluarDesempenyo() {  
        return String.format("Evaluando desempeño del ingeniero %s", nombre);  
    }  
}
```

```
public class SistemaGestion {  
  
    public static void main(String[] args) {  
        Empleado empleado1 = new Gerente("Ana", 5000, 500);  
        Empleado empleado2 = new Ingeniero("Luis", 3000);  
  
        empleado1.incrementarSalario(10);  
        if ( /* código_if */ ) {  
            System.out.println(((Ingeniero)empleado2).evaluarDesempenyo());  
        }  
    }  
}
```

26) ¿Cómo se relaciona la clase Gerente con la clase Empleado en el contexto de constructores?

- a. La clase Gerente reemplaza completamente el constructor de Empleado.
- b. Gerente utiliza el constructor de Empleado mediante la llamada a super para su propia inicialización.
- c. Gerente y Empleado deben tener constructores idénticos para ser compatibles.
- d. La clase Gerente ignora el constructor de Empleado y utiliza uno predeterminado.

27) ¿Qué característica define mejor a la clase Empleado y su método incrementarSalario?

- a. Empleado puede instanciarse y su método incrementarSalario puede definir su lógica en la misma clase o en las subclases mediante sobrescritura.
- b. Empleado es una clase que no se puede instanciar, haciendo que su método incrementarSalario deba ser obligatoriamente implementado por las subclases.
- c. El método incrementarSalario devuelve el nuevo salario actualizado con el porcentaje ya aplicado.
- d. Empleado puede ser instanciada directamente si se proporciona una implementación para el método incrementarSalario en el momento de la instancia (mediante la palabra reservada new).

28) ¿Qué código se debe indicar entre los paréntesis de la sentencia if, sustituyendo a /*código_if*/, para que no se produzca una excepción de tipo ClassCastException?

- a. empleado2.getClass() == "Ingeniero".
- b. empleado2 != null.
- c. empleado2 instanceof Ingeniero.
- d. empleado2 is Evaluable.



29) Considerando las definiciones y usos de Empleado como clase abstracta e Evaluable como interfaz, ¿cuál de las siguientes afirmaciones describe adecuadamente una diferencia clave entre clases abstractas e interfaces en Java?

- a. Las clases abstractas pueden contener implementaciones completas para algunos métodos y para otros no, mientras que las interfaces tienen que tener implementación en todos los métodos declarados o en ninguno de ellos.
- b. Una clase en Java puede heredar múltiples clases abstractas para aprovechar la reutilización de código, pero solo puede implementar una interfaz a la vez.
- c. Las interfaces son usadas para definir tipos de datos, mientras que las clases abstractas son utilizadas para proporcionar una implementación parcial.
- d. Las interfaces permiten la herencia múltiple de implementación en Java, proporcionando un medio para eludir la restricción de herencia simple de Java.

Teniendo en cuentas las siguientes clases de Java, del mismo proyecto, las cuales tienen todos los paquetes necesarios importados responda a las preguntas relacionadas con dicho código fuente.

```
public class DivisionPorCeroException extends Exception {  
  
    public DivisionPorCeroException(String mensaje) {  
        super(mensaje);  
    }  
}
```

```
public class Calculadora {  
  
    public double dividir(int numerador, int denominador) throws DivisionPorCeroException {  
        if (denominador == 0) {  
            throw new DivisionPorCeroException("No se puede dividir por cero.");  
        }  
        return numerador / (double)denominador;  
    }  
}
```

```
public class CalculadoraApp {  
  
    public static void main(String[] args) {  
        Calculadora calc = new Calculadora();  
  
        try {  
            double resultado = calc.dividir(10, 0);  
            System.out.println("Resultado: " + resultado);  
        } catch (DivisionPorCeroException e) {  
            System.err.println("Error en la división: " + e.getMessage());  
        } finally {  
            System.out.println("Operación finalizada.");  
        }  
    }  
}
```



30) ¿Qué caracteriza a la clase `DivisionPorCeroException` dentro del contexto del programa?

- a. Es una clase que extiende de `RuntimeException`, por lo que no es obligatorio que se declare en el `throw` aunque sí es recomendable hacerlo.
- b. Se trata de una excepción que extiende de `Exception`, obligando a ser manejada o declarada mediante `throws`.
- c. Es una clase que implementa la interfaz `Error`, indicando un error grave que el compilador deberá manejar.
- d. Es una excepción no comprobada que el compilador no requiere ser manejada o declarada.

31) En el método `dividir`, ¿qué técnica de manejo de excepciones se utiliza para la `DivisionPorCeroException`?

- a. Captura inmediata dentro del mismo método.
- b. Supresión, ignorando cualquier excepción de este tipo que ocurra.
- c. Propagación, declarándola en la firma del método.
- d. Conversión a una excepción no comprobada y lanzamiento.

32) ¿Cuál es el propósito de incluir el bloque `finally` en el método `main`?

- a. Para asegurar que la excepción `DivisionPorCeroException` sea siempre capturada.
- b. Para ejecutar código de limpieza que debe ejecutarse independientemente de si ocurrió una excepción o no.
- c. Para reintentar la operación de división en caso de fallo inicial.
- d. Para lanzar cualquier excepción que no haya sido capturada en los bloques `try` o `catch`.

33) Si `denominador` es cero, ¿qué efecto tiene utilizar `DivisionPorCeroException` en el flujo del programa?

- a. El programa termina inmediatamente, mostrando la pila de llamadas hasta el punto de la excepción.
- b. La ejecución del método `dividir` se detiene, y el control se transfiere al bloque `catch` correspondiente que puede manejar esta excepción.
- c. El compilador automáticamente convierte `DivisionPorCeroException` en una advertencia, permitiendo que el programa continúe.
- d. La excepción es ignorada a menos que se especifique explícitamente en un bloque `finally`.



Teniendo en cuentas las siguientes clases de Java, del mismo proyecto, las cuales tienen todos los paquetes necesarios importados responda a las preguntas relacionadas con dicho código fuente.

```
public class Usuario implements java.io.Serializable {  
    private String nombre;  
    private int edad;  
  
    public Usuario(String nombre, int edad) {  
        this.nombre = nombre;  
        this.edad = edad;  
    }  
  
    @Override  
    public String toString() {  
        return String.format("El usuario %s tiene %d año(s)", nombre, edad);  
    }  
}
```

```
public class RegistroUsuarios {  
  
    public RegistroUsuarios() {}  
  
    public static void guardarUsuario(Usuario usuario, String archivo) throws IOException {  
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(archivo))) {  
            oos.writeObject(usuario);  
        }  
    }  
  
    public static void leerUsuarios(String archivo) throws IOException, ClassNotFoundException {  
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(archivo))) {  
            while (true) {  
                try {  
                    Usuario usuario = (Usuario) ois.readObject();  
                    System.out.println(usuario);  
                } catch (EOFException e) {  
                    break;  
                }  
            }  
        }  
    }  
}
```

```
public class UsuariosApp {  
  
    public static void main(String[] args) {  
        String archivo = "usuarios.dat";  
        Usuario usuario1 = new Usuario("Ana", 30);  
        Usuario usuario2 = new Usuario("Luis", 25);  
  
        try {  
            RegistroUsuarios.guardarUsuario(usuario1, archivo);  
            RegistroUsuarios.guardarUsuario(usuario2, archivo);  
            RegistroUsuarios.leerUsuarios(archivo);  
        } catch (IOException | ClassNotFoundException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



34) En el método leerUsuarios, ¿cómo se maneja el final del archivo (EOF) durante la lectura?

- a. Usando un bucle while que verifica `ois.available() > 0`.
- b. Capturando la excepción `EOFException` y terminando el bucle.
- c. Comprobando si `readObject()` retorna null.
- d. Utilizando `FileInputStream` para verificar el tamaño del archivo.

35) ¿Cuál es el propósito de implementar la interfaz Serializable en la clase Usuario?

- a. Permitir que los objetos Usuario sean enviados a través de flujos de red.
- b. Habilitar el almacenamiento y recuperación de objetos Usuario de un archivo.
- c. Asegurar que sólo los objetos Usuario puedan ser escritos en un archivo especificado.
- d. Incrementar la eficiencia del almacenamiento en archivos de objetos Usuario.

36) Considerando la funcionalidad del programa, ¿qué mejora podría implementarse para optimizar la escritura de múltiples objetos Usuario en el mismo archivo?

- a. Usar `FileWriter` en lugar de `ObjectOutputStream` para mejorar la eficiencia.
- b. Inicializar `ObjectOutputStream` fuera del método `guardarUsuario` para reutilizarlo.
- c. Aplicar compresión de datos al objeto Usuario antes de escribirlo en el archivo.
- d. Crear un `ArrayList<Usuario>` y escribir la lista completa en el archivo en lugar de objetos individuales.



Teniendo en cuenta la siguiente clase de Java, la cual tiene todos los paquetes necesarios importados, responda a las preguntas relacionadas con dicho código fuente.

```
import java.sql.*;

public class GestionLibros {

    public static Connection conectarConBD() {
        String url = "jdbc:mysql://localhost:3306/biblioteca";
        String usuario = "usuarioBD";
        String contrasena = "passwordBD";

        try {
            return DriverManager.getConnection(url, usuario, contrasena);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void insertarLibro(Connection conn, String titulo, String autor) {
        String sql = "INSERT INTO libros (titulo, autor) VALUES (?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, titulo);
            pstmt.setString(2, autor);
            pstmt.executeUpdate();

            System.out.println("Libro insertado correctamente.");
        } catch (SQLException e) {
            System.err.println("Error al insertar el libro: " + e.getMessage());
        }
    }

    public static void consultarLibros(Connection conn) {
        String sql = "SELECT * FROM libros";

        try (Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.println(rs.getInt("id") + ": " +
                    rs.getString("titulo") + " - " +
                    rs.getString("autor"));
            }
        } catch (SQLException e) {
            System.err.println("Error al consultar los libros: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        Connection conn = conectarConBD();

        if (conn != null) {
            insertarLibro(conn, "Don Quijote", "Miguel de Cervantes");
            consultarLibros(conn);

            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```



37) En el método insertarLibro, ¿cuál es el propósito de usar PreparedStatement en lugar de Statement?

- a. Permite ejecutar consultas sin parámetros.
- b. Mejora el rendimiento al reutilizar el objeto para múltiples consultas.
- c. Previene inyecciones SQL al permitir el uso de parámetros.
- d. Sólo PreparedStatement permite la ejecución de consultas de inserción.

38) ¿Cuál es la clase que permite iterar sobre la información obtenida de la base de datos?

- a. ResultSet.
- b. Statement.
- c. PreparedStatement.
- d. DriverManager.

39) Tomando como referencia el código fuente, ¿cuántas columnas, como mínimo, existen en la tabla libros?

- a. Como mínimo 2 columnas.
- b. Como mínimo 3 columnas.
- c. Como mínimo 4 columnas.
- d. Como mínimo 5 columnas.

40) Si se desea implementar un programa que muestre una pantalla gráfica al hacer clic sobre un botón, ¿dónde se debe poner el código para mostrar dicha pantalla gráfica?

- a. En el escuchador o listener.
- b. En el manejador o handler.
- c. En el método equals() del evento.
- d. En el método show() del modificador de acceso del objeto de tipo botón.