AUTOEVALUACIÓN UT11

1. ¿Qué solución producirá el siguiente fragmento de código?

**package** lecturaescritura;

**import** java.io.\*;

**public** **class** LecturaEscritura {

**public** **static** **void** main(String[] args) {

PrintWriter pw;

pw=**new** PrintWriter(System.***out***);

pw.printf("Escribiendo en consola por %d vez....\n",1,2,3,4,5);

}

}

* 1. Error de compilación, pues el número de argumentos pasados a printf es superior al indicado en la cadena de formato
  2. El código compilará y la ejecución imprimirá “Escribiendo en consola por 1 vez..”, terminando con un salto de línea.
  3. El código compilará y la ejecución imprimirá “Escribiendo en consola por 1 vez…\n”
  4. El código compilará y la ejecución no imprimirá nada por consola.

1. Las **clases BufferedXXXX**:
   1. Hacen uso de un buffer intermedio.
   2. Controlan los accesos a los ficheros para mejorar el rendimiento de la operación.
   3. Las opciones a y b son correctas.
   4. Todas las opciones anteriores son incorrectas.
2. ¿En qué paquete se encuentra la **clase Files**?
   1. java.lang
   2. java.io
   3. java.nio
   4. java.util
3. ¿Qué **método** emplearía para **crear un fichero vacío** a partir de **un objeto File existente** sobre el cual se ha establecido el path completo?
   1. newFiles()
   2. newFile()
   3. createFile()
   4. createNewFile()
4. ¿Cuál de las siguientes clases **NO hereda** directamente de **Reader**?
   1. CharaArrayReader
   2. LineNumberReader
   3. BufferedReader
   4. StringReader
5. La clase empleada para manejar **flujos de datos de entrada binarios** es:
   1. InputStream
   2. OutputStream
   3. StreamInput
   4. InputReader
6. Si queremos **guardar una cadena de caracteres** en un flujo binario de tipo **ObjectOutputStream**, usaremos:
   1. writeString()
   2. writeChar()
   3. writeObject()
   4. Nada, no se puede
7. Si queremos **leer una tabla de adenas de caracteres** del flujo binario entrada de tipo **ObjectInputStream,** escribiremos:
   1. String[] tabla=(String[])entrada.readObject();
   2. String tabla=(String)entrada.readObject();
   3. String[] tabla=entrada.readObject();
   4. String[] table=(Object).readObject();
8. Un flujo de tipo **ObjectInputStream** permite acceder a:
   1. Sólo archivos del disco duro.
   2. Cualquier fuente de datos primitivos u objetos de Java
   3. Únicamente a conexiones de red
   4. Solo nos permite leer de la consola.
9. Si guardamos una **cadena de caracteres** usando un **flujo ObjectOutputStream**, podemos **leerla** directamente del archivo:
   1. Usando un procesador de texto
   2. Usando un editor de texto
   3. Usando una hoja de cálculo
   4. Usando un flujo ObjectInputStream
10. Si guardamos una serie de **objetos de la clase Cliente** con un **flujo ObjectOutputStream**, los recuperaremos:
    1. En el mismo orden en que se guardaron
    2. En orden inverso
    3. En un orden aleatorio
    4. Nunca se pueden recuperar.
11. Los **flujos binarios** se cierran:
    1. Con el método close();
    2. Apagando el ordenador
    3. Abortando el programa
    4. Con el método cerrar()
12. Hay que **cerrar los flujos binarios**:
    1. Siempre
    2. Una vez al día
    3. Sólo si no se han abierto con una estructura try-catch con recursos
    4. Nunca.
13. Para encontrar una información almacenada en la mitad de un **fichero secuencial**, podemos **acceder directamente a esa posición** sin pasar por los datos anteriores a esa información.
    1. Verdadero
    2. Falso
14. Para **leer datos** desde un fichero codificados en binario empleamos la clase **FileOutputStream**
    1. Verdadero
    2. Falso
15. Cuando trabajamos con **fichero en Java**, **no es necesario capturar las excepciones**, el sistema se ocupa automáticamente de ellas. ¿Verdadero o Falso?
    1. Verdadero
    2. Falso
16. Para **flujos de caracteres** es mejor usar las clases **Reader y Writer** en vez **de InputStream y OutputStream**
    1. Verdadero
    2. Falso