Test Fin

Prueba entregada

- Fecha de entrega No hay fecha de entrega
- Puntos 10
- Preguntas 22
- Disponible 9 de sep a las 0:00 - 29 de jun de 2025 a las 23:59
- Límite de tiempo Ninguno
- Intentos permitidos Ilimitado

Instrucciones

Este test incluye un total de **22 preguntas** sobre los diferentes contenidos aprendidos en esta unidad.

¡Adelante!

Volver a hacer la prueba

Historial de intentos

	Intento	Hora	Puntuación
MÁS RECIENTE	Intento 1	5 minutos	9,75 de 10

Puntuación para este intento: 9,75 de 10
Entregado 19 de nov a las 12:33
Este intento ha tenido una duración de 5 minutos.

Pregunta 1
0,5 / 0,5 puntos

Dada la siguiente sentencia:

```
File fich = new File("coches\\ford\\modelos.txt");
```

Marca qué operaciones podemos realizar con la variable *fich*. Hay varias respuestas correctas.

¡Correcto!

- [x] Comprobar si existe el fichero modelos.txt.
- [ ] Añadir nuevos modelos de coche en el fichero modelos.txt.

¡Correcto!

- [x] Obtener el número de bytes que ocupa el fichero dentro del sistema de archivos.

Prueba entregada

- [ ] Borrar una lín
- [ ] Modificar la n

¡Correcto!

- [x] Comprobar si el fichero modelos.txt tiene permisos de escritura.

La clase File no nos permite efectuar operaciones de lectura/escritura de ficheros; para eso están los flujos de datos.

Pregunta 2

0,5 / 0,5 puntos

Marca las afirmaciones correctas sobre la clase *File*.

¡Correcto!

- [x] Permite crear un fichero.
- [ ] Permite crear un fichero y escribir una línea en él.

¡Correcto!

- [x] Permite crear una carpeta.
- [ ] Permite crear un fichero y lo abre para escritura.

¡Correcto!

- [x] Permite borrar una carpeta.

La clase File nos permite crear un fichero, pero no escribir en él; para eso están los flujos de datos.

Pregunta 3

0,5 / 0,5 puntos

Indica la afirmación correcta sobre el lenguaje Java.

- ○ Dispone de clases que representan flujos de datos que son simultáneamente de lectura/escritura.
- ○ Dispone de clases que representan flujos de lectura, pero no de escritura.

¡Correcto!

- ◉ Dispone de clases que representan flujos de lectura y de clases que representan flujos de escritura, pero no existen clases que representen flujos de lectura/escritura.

Java no dispone de clases que representen flujos de lectura y escritura. Si necesitamos leer y escribir en un fichero necesitamos dos flujos distintos: un flujo de entrada o lectura, y otro de salida o escritura.

Pregunta 4

0,5 / 0,5 puntos

Para que el usuario pueda introducir una frase por teclado es necesario:

- ○ Un flujo de datos de escritura.

¡Correcto!

- ◉ Un flujo de datos de lectura.

Los *stream* o flujos de datos permiten transmitir secuencias ordenadas de datos desde un origen a un destino. El [Prueba entregada] itivo (lectura de teclado, escritura en pantalla).

- Ninguna de las anteriores, los flujos de datos sólo permiten la entrada/salida con ficheros.

Pregunta 5
0,5 / 0,5 puntos
Selecciona la afirmación correcta:
¡Correcto!

- Todas las clases que representan flujos de datos están ubicadas en el paquete *java.io*.

Las clases que representan flujos de datos están en el paquete java.io.

- Todas las clases que representan flujos de datos están ubicadas en el paquete *java.util*.
- Todas las clases que representan flujos de datos están ubicadas en el paquete *java.lang*.

Pregunta 6
0,5 / 0,5 puntos
El método *exists()* de la clase *File...*

- Permite comprobar si existe un fichero.
- Permite comprobar si existe una carpeta.

¡Correcto!

- Permite comprobar si existen, tanto un fichero, como una carpeta.

Un objeto de la clase *File* puede representar un fichero o una carpeta, por lo tanto, el método *exists()* puede comprobar la existencia, tanto de un fichero, como de una carpeta.

Pregunta 7
0,5 / 0,5 puntos
¿Qué retorna el método *list()* de la clase *File*?

- Un array de objetos *File*.

¡Correcto!

- Un array de objetos *String*.

Aplicado sobre un objeto *File* que representa una carpeta, el método *list()* retorna un array de objetos *String* con nombres de los archivos o carpetas que contiene.

- Un array de objetos genéricos *Object*; se aplica polimorfismo.

Pregunta 8
0,5 / 0,5 puntos
¿Cómo se obtiene el tamaño de un archivo con la clase *File*?
¡Correcto!

- Con el método *length().*

Aplicado sobre un objeto *File* que representa un fichero, el método *length()* retorna el tamaño del fichero expresado en bytes.

- Con el método *size().*

Con el método

Prueba entregada

Pregunta 9

0,5 / 0,5 puntos

¿Cuáles de estas clases representan flujos de datos que actúan como filtros? Hay varias respuestas correctas.

¡Correcto!

- [x] *BufferedReader*

¡Correcto!

- [x] *DataInputStream*
- [ ] *FileInputStream*
- [ ] *FileOutputStream*

¡Correcto!

- [x] *DataOutputStream*
- [ ] *FileReader*

¡Correcto!

- [x] *BufferedWriter*
- [ ] *FileWriter*

Los flujos de datos iniciadores se interactúan directamente con el dispositivo para leer o escribir información. Los flujos de datos que representan filtros se sitúan entre el flujo iniciador y el programa.

Pregunta 10

0,5 / 0,5 puntos

¿Cuál de estas clases representa un flujo de datos que permite escribir en un fichero datos de tipo primitivo (*int*, *double*, *float*, etc.)?

- [ ] *BufferedReader*

¡Correcto!

- [x] *DataOutputStream*

DataInputStream permite escribir datos en un fichero directamente como tipos de datos primitivos (*int*, *float*, *double*, etc.). Actúa como filtro y trabaja en colaboración con otra clase iniciadora, como *FileOutputStream*.

- [ ] *FileReader*
- [ ] *FileInputStream*

Pregunta 11

0,5 / 0,5 punto

Prueba entregada

La clase abstracta que sirve como base para todos los flujos de salida de datos en formato binario es...

¡Correcto!

- *OutputStream*

Las clases base abstractas de las que derivan todos las clases de flujos de datos de salida son: *Writer* (formato Unicode) e *OutputStream* (formato binario).

- *Writer*
- *Reader*
- *InputStream*

Pregunta 12

0,5 / 0,5 puntos

¿Podemos leer un fichero de texto a partir de un objeto de tipo *FileInputStream*?

- No, *FileInputStream* representa un flujo de datos en formato binario y sólo se pueden leer ficheros de texto en formato Unicode de 16 bits.

¡Correcto!

- Sí, aunque hay que realizar la lectura carácter a carácter.

Es posible leer un texto de un archivo con un objeto *FileInputStream,* pero hay que ir leyendo uno a uno todos los bytes que corresponden al texto. No es necesario un flujo que actúe como filtro, aunque mejoraría el rendimiento.

- No, es necesario un iniciador y un filtro, *FileInputStream* es sólo iniciador.

Pregunta 13

0,5 / 0,5 puntos

El método *readLine()* de la clase *BufferedReader...*

¡Correcto!

- Lee una línea del buffer y retorna un objeto *String* con dicha línea.

BufferedReader es un flujo de datos de tipo filtro que proporciona un buffer para realizar la lectura de datos de manera más eficiente. El método *readLine()* lee la siguiente línea del buffer devolviendo un *String* con dicha línea.

- Lee una línea del buffer y retorna un array de objetos *char* con los caracteres de dicha línea.
- La clase BufferedReader no tiene ningún método llamado *readLine()*.

Pregunta 14

0,5 / 0,5 puntos

¿Qué clase permite escribir datos elementales (*int*, *float*, *double*, etc.) en un fichero?

- *DataInputStream*

FileOutputStr

Prueba entregada

¡Correcto!

DataOutputStream

La clase *DataOutputStream* permite escribir datos en un fichero directamente como tipos de datos primitivos (*int*, *float*, *double*, etc.). Actúa como filtro y trabaja en colaboración con otra clase iniciadora como *FileOutputStream*.

Pregunta 15
0,5 / 0,5 puntos

WriteInt(int valor) es un método de la clase...

OutputStreamReader

DataInputStream

¡Correcto!

DataOutputStream

La clase *DataOutputStream* provee métodos para la escritura de datos de tipo elemental como *writeFloat*, *writeInt*, *writeBoolean*, *writeLong*, etc.

BufferedReader

Pregunta 16
0,5 / 0,5 puntos

ReadFloat() es un método de la clase...

¡Correcto!

DataInputStream

La clase *DataInputStream* provee métodos para la lectura desde fichero de datos de tipo elemental (*int*, *float*, *double*, etc.). Estos métodos son: *readBoolean(), readByte(), readChar(), readDouble(), readFloat(),* etc.

DataOutputStream

BufferedReader

Pregunta 17
0,5 / 0,5 puntos

¿Cuál es la clase que sirve como base para todas las excepciones de entrada / salida?

NumberFormatException

Prueba entregada

FileExceptior

InputOutputException

¡Correcto!

IOException

La clase base de la que derivan todas las clases que representan excepciones en operaciones de entrada / salida es *IOException,* que a su vez deriva de *Exception*.

Pregunta 18

0,5 / 0,5 puntos

¿Cuál de estas operaciones no puede realizarse con un objeto de la clase Scanner?

¡Correcto!

Escribir datos en un fichero de texto.

Scanner, como su nombre indica representa un lector y nos permite leer datos de teclado, de un *String* o de un fichero.

Leer datos de un fichero de texto.

Leer datos desde el teclado.

Leer una cadena de texto delimitada por punto y coma.

Pregunta 19

0,25 / 0,25 puntos

Dado el siguiente código:

```
String texto="Luna nueva;Luna creciente;Luna menguante";
Scanner lector = new Scanner(texto);
lector.useDelimiter(";");
lector.next();
System.out.println(lector.next());
lector.close();
```

¿Cuál será el resultado de su ejecución?

Luna.

¡Correcto!

Luna creciente.

El primer *next()* lee el fragmento “Luna nueva”, pero no se muestra en pantalla; el segundo *next()* lee “Luna creciente” y está vez si se muestra en pantalla, porque está encerrado en una sentencia *System.out.println(…).*

Luna decreciente.

Luna nueva.

Prueba entregada

Pregunta 20

0 / 0,25 puntos

Dado el siguiente código:

```
String texto="Rosa López;Miguel De la Parra;Carmen Ruiz";
Scanner lector = new Scanner(texto);
lector.next();
System.out.println(lector.next());
lector.close();
```

¿Cuál será el resultado de su ejecución?

López.

Respondido

Miguel de la Parra.

Respuesta correcta

López;Miguel.

Rosa López.

Pregunta 21

0,25 / 0,25 puntos

La interfaz *Serializable...*

¡Correcto!

Aporta a las clases que implementan la capacidad de persistencia.

Para que un objeto pueda ser guardado en disco debe pertenecer a una clase que implemente la interfaz *Serializable*, es decir, una clase que tenga capacidad de persistencia.

Hace que los objetos de las clases que la implementan se guarden automáticamente en disco.

Serializable no es una interfaz, es una clase abstracta.

Pregunta 22

0,25 / 0,25 puntos

¿Qué función tiene la variable *serialVersionUID*?

Asignar un código numérico a cada objeto que se crea.

¡Correcto!

Contener el número de versión de una clase para evitar problemas en la serialización o deserialización de objetos.

La *serialVersionUID* es el número de versión de la clase, y se utiliza para evitar problemas de incompatibilidad de versión en los procesos de serialización y deserialización entre los que hacen de emisor y receptor del objeto.

Asignar un número de versión secuencialmente a cada uno de los objetos de una clase.

Puntuación de la prueba: 9,75 de 10