11.11. Pide un valor **double** por consola y guárdalo en un archivo binario.

11.12. Abre el fichero de la actividad anterior, y lee el valor **double** contenido en él y muéstralo por pantalla.

11.13. Escribe un programa que lea de un fichero binario una tabla de números **double** y después muestre el contenido de la tabla por consola.

11.14. Introduce por teclado una frase y guárdala en un archivo binario. A continuación, recupérala y muéstrala por pantalla.

11.15. Implementa un programa que lea números enteros desde el fichero números.dat y los vaya guardando en el fichero **pares.dat** e **impares.dat**, según su paridad.

11.16. Implementa una aplicación que gestione una lista de nombres ordenada por orden alfabético. Al arrancar se leerá de un fichero los nombres insertados anteriormente y se pedirán nombres nuevos hasta que se introduzca la cadena "fin". Cada nombre que se introduzca deberá añadirse a los que ya había, de forma que la lista permanezca ordenada. Al terminar, se guardará en el fichero la lista actualizada.

11.17. Escribe un texto, línea a línea, de forma que, cada vez que se pulse Intro, se guarde la línea en un archivo binario. El proceso se termina cuando introduzcamos una línea vacía. Después el programa lee el texto completo del archivo y lo muestra por pantalla.

11.18. Un libro de firmas es útil para recoger los nombres de todas las personas que han pasado por un determinado lugar. Crea una aplicación que permita mostrar el libro de firmas o insertar un nuevo nombre (comprobando que no se encuentre repetido) usando el fichero binario firmas.dat.

11.19. Por motivos puramente estadísticos se desea llevar constancia del número de llamadas recibidas cada día en una oficina. Para ello, al terminar cada jornada laboral se guarda dicho número al final de un archivo binario. Implementa una aplicación con un menú, que nos permita añadir el número correspondiente cada día y ver la lista completa en cualquier momento.

11.20. Implementa una aplicación que permita guardar y recuperar los datos de los clientes de una empresa. Para ello, define la clase **Cliente**, que tendrá los atributos: id (identificador de cliente), **nombre** y **teléfono**. Los objetos **Cliente** se insertarán en una tabla.

Para realizar las distintas operaciones, la aplicación tendrá el siguiente menú:

1. Añadir nuevo cliente.
2. Modificar datos.
3. Dar de baja cliente.
4. Listar los clientes.

La información se guardará en un fichero binario, que se cargará en la memoria al iniciar la aplicación y se grabará en disco, actualizada, al terminar.

11.21. Repite la actividad anterior, pero insertando los objetos **Cliente** en un objeto **Lista** para **Object**, como el definido en la Actividad Resuelta 9.11.

11.22. Implementa una aplicación que gestione los empleados de un banco. Para ello se definirá la clase **Empleado** con los atributos **dni**, **nombre** y **sueldo**. Los empleados se guardarán en un objeto de la clase **Lista** para objetos de la clase **Object**. La aplicación cargará en la memoria, al arrancar, la lista de empleados desde el archivo binario empleados.dat y mostrará un menú con las siguientes opciones:

1. Alta empleado.
2. Baja empleado.
3. Mostrar datos empleado.
4. Listar empleados.
5. Salir.

Al pulsar 5, se grabará en el disco la lista actualizada y terminará el programa.

11.23. Implementa el método, **Integer[ ] fusionar(String fichero1, String fichero2)**, al que se le pasan los nombres de dos ficheros binarios que contienen dos listas ordenadas de objetos **Integer**, y devuelve una tabla ordenada con todos los elementos de los dos ficheros fusionados.

11.24. Implementa el método, **void fusionar(String ficheroBase, String ficheroNuevo)**, que añade a **ficheroBase**, los elementos de **ficheroNuevo**, ambos ordenados. Al final, **ficheroBase** contiene la lista ordenada de todos los elementos de ambos ficheros.

11.25. En una tabla de cadenas se guardan los nombres de 4 ficheros binarios. En cada uno de ellos se guarda una tabla de números enteros ordenados en sentido creciente. Implementa una aplicación donde se introduce por teclado un número entero. El programa debe determinar si ese número se halla en alguno de los 4 ficheros y, en caso afirmativo, en cuál de ellos y en qué lugar de la tabla correspondiente…

11.26. Utilizando un fichero aleatorio, realiza un programa que le muestre al usuario un menú con las siguientes opciones:
Añadir números de tipo double al principio del fichero.

Añadir números de tipo double al final del fichero.
Mostrar el fichero creado.
Sustituir un número indicado por el usuario por otro número que también indique el usuario.
Nota: Un double en JAVA ocupa 8 bytes.

11.27. En base a dos rutas correspondientes a dos directorios, copiar todo el contenido del primero en el segundo (subdirectorios y ficheros).

11.28. En base a un archivo que almacena una serie de referencias y precios de artículos, actualizar en base al IPC actual los precios, de forma que los superiores a 100 euros se incrementen en un 20% y los inferiores se incrementen en un 30%. Se recomienda utilizar la clase java.text.DecimalFormat, para obligar a que el precio tenga un tamaño controlado (rlnPrecio = new DecimalFormat("#.##"); los datos ocuparán lo mismo que un double 8 bytes y se pueden recoger en variables de este tipo)