1. (Notas) Escribir un programa que almacene en un fichero de texto (notas.txt) las notas de 20 alumnos. El programa tendrá el siguiente funcionamiento:

    - En el fichero se guardarán como máximo 20 notas, pero se pueden guardar menos. El proceso de introducción de notas (y en consecuencia, el programa) finalizará cuando el usuario introduzca una nota no válida (menor que cero o mayor que 10).
    - Si, al comenzar la ejecución, el fichero ya contiene notas, se indicará al usuario cuántas faltan por añadir y las notas que introduzca el usuario se añadirán a continuación de las que hay.
    - Si, al comenzar la ejecución, el fichero ya contiene 20 notas, se le preguntará al usuario si desea sobrescribirlas. En caso afirmativo las notas que introduzca sustituirán a las que hay y en caso negativo el fichero no se modificará

2. (IndiceMasaCorporal). Disponemos de un fichero de texto que contiene el nombre, el peso y la estatura de una serie de personas, cada dato en una línea. Escribir un programa que lea todo el fichero y muestre por pantalla el nombre de cada persona junto con su índice de masa corporal. El IMC es el peso dividido por el cuadrado de la estatura.

3. (IMCaFichero). Copia la solución del programa anterior y modifícala para que el nombre y el IMC de cada persona no se muestre por pantalla, sino que se grabe en el fichero de texto IMC.txt

4. (CompararFicheros). Implementa un método boolean sonInguales(File f1, File f2), que dados dos objetos de tipo File, devuelva true si los ficheros son iguales, es decir si tienen exactamente el mismo contenido. Si los ficheros tienen el mismo tamaño habrá que abrirlos y leerlos byte a byte para ver si sus contenidos difieren en algo. Para leer, puedes utilizar el método read() de DataInputStream, que lee un byte de información. Utiliza el método main para probar el método sonIguales()

5. (DatosMezclados). Tenemos un fichero de texto que contiene numeros enteros mezclados con otros datos (textos, números reales, etc). Realizar un programa que calcule la media de todos los enteros que hay en el fichero

6. (BuscarProyectosEclipse) Implementar un programa que solicite el nombre de una carpeta y grabe en el fichero proyectos.txt el nombre de todos los proyectos de Eclipse que contiene dicha carpeta. Un proyecto de eclipse es una carpeta que contiene:

    - un fichero llamado .classpath
    - un fichero llamado .project
    - una carpeta llamada .settings

7. (Censura) Escribir un programa que sustituya por otras, ciertas palabras de un fichero de texto. Para ello, se desarrollará el siguiente método y se probará desde el main:

    void aplicaCensura(String fichCensura, String fichEntrada, String fichSalida),

    que lee de un fichero de entrada y mediante un fichero de censura, crea el correspondiente fichero modificado. Por ejemplo:

Fichero de entrada:

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, *no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero*

Fichero de censura

lugar sitio

acordarme recordar

hidalgo noble

Fichero de salida

En un **sitio** de la Mancha, de cuyo nombre no quiero **recordar**, *no ha mucho tiempo que vivía un* ***noble*** *de los de lanza en astillero*

Habrá que seguir los siguientes pasos:

- Leer el fichero de censura y almacenar las parejas de palabras en un Map.
- Abrir el fichero de entrada y el de salida. Recorrer las palabras del fichero de entrada:

  Leer palabra del fichero de entrada.

  Comprobar si está en el map.

  Si está en el map, escribir en el fichero de salida su sustituta.

  Si no está en el Map, escribir en el fichero de salida la palabra original.

8. (Personas) Escribe un programa que, utilizando entre otras la clase DataOutputStream, almacene en un fichero binario llamado personas.dat la información relativa a una serie de personas que va introduciendo el usuario desde teclado:

   - Nombre (String)
   - Edad (entero)
   - Peso (double)
   - Estatura (double)

   La entrada del usuario terminará cuando se introduzca un nombre vacío.

9. (BinarioATexto) Escribe un programa que lea el fichero generado en el ejercicio anterior y guarde su contenido en un fichero de texto, con cada dato en una línea.
10. (Concatenar1) Escribe un programa que dados dos ficheros de texto f1 y f2 confeccione un tercer fichero f3 cuyo contenido sea el de f1 y a continuación el de f2.
11. (Concatenar2) Escribe un programa que dados dos ficheros de texto f1 y f2, añada al final de f1 el

contenido de f2. Es decir, como el ejercicio anterior, pero sin producir un nuevo fichero.

12. (Jugadores) Desarrollar una clase con los siguientes métodos:

    - grabarJugadores(): Pide al usuario, uno a uno, los datos de jugadores de baloncesto: (dorsal, nombre y estatura). Los datos se almacenarán en un fichero binario (jugadores.bin). El método finalizará cuando el usuario introduzca un jugador de dorsal 0.
    - mostrarJugadores(): Recorre el fichero jugadores.bin y muestra su contenido por pantalla.
    - ordenarJugadores(): Recorre el fichero de jugadores y almacena su contenido en un TreeSet<Jugador> con el objetivo de ordenarlo y eliminar sus duplicados. A continuación guarda el contenido del TreeSet, de nuevo, en el fichero. El fichero quedará ordenado y sin duplicados. Se desea ordenar por dorsal y que no haya dos jugadores con el mismo dorsal.

    Para ello, habrá que crear la clase Jugador, con los correspondientes atributos, su método equals y compareTo.

    Usar el método main para probar los métodos desarrollados.

13. (LigaFutbol) Supongamos que disponemos de una carpeta que contiene un fichero de texto por cada jornada de la liga de futbol española.

    - Cada fichero contiene los resultados de una jornada.
    - Los ficheros tienen una línea por cada partido de la jornada, con el siguiente formato, donde local y visitante son las siglas del equipo local y del equipo visitante respectivamente:

    ```
    local  goles visitante goles
    ```

    Desarrolla un programa que muestre por pantalla los puntos obtenidos por cada equipo, teniendo en cuenta que un partido ganado supone 3 puntos, uno perdido supone 0 puntos y uno empatado supone 1 punto. Sugerencia: Apóyate en un Map para contar los puntos.

14. (LigaFutbol) Modifica el programa anterior para que muestre los resultados pero ordenados por puntuación, es decir, en primer lugar el que más puntos ha obtenido.

15. (ServidorSumas) En el código que se muestra a continuación hay dos clases, llamadas Cliente y Servidor.

    La clase Servidor, se pone a la espera de recibir el intento de conexión de un cliente por un puerto determinado.

    La clase Cliente trata de conectarse con un servidor conocida su dirección IP y el puerto por el que escucha.

    Una vez que se produce la conexión, cliente y servidor abren Streams para comunicarse: Lo que el cliente escriba en su OutputStream lo podrá leer el Servidor en su InputStream y viceversa.

    - Modificar el cliente y el servidor de manera que el cliente pase al servidor dos números enteros y el servidor los sume y devuelva el resultado.
    - Modificar el cliente y el servidor de manera que el servidor transmita al cliente un fichero de

texto y el cliente lo muestre por pantalla.

Para poder probar tus programas tendrás que ejecutar en la misma máquina y al mismo tiempo la aplicación cliente y la aplicación servidor, o bien, el cliente en un equipo y el servidor en otro, indicando en la constante que se ha definido en el cliente la dirección IP del servidor

```
import java.net.*;
import java.io.*;

public class Servidor {
    final static int PUERTO = 5000;
    public Servidor(){
        try{
            ServerSocket skServidor = new ServerSocket(PUERTO);
            System.out.println("Servidor: escuchando en el puerto " + PUERTO);
            Socket skCliente = skServidor.accept();

            System.out.println("Servidor: Sirviendo a un cliente");
            DataOutputStream flujoSal = new DataOutputStream(skCliente.getOutputStream());
            DataInputStream flujoEnt = new DataInputStream(skCliente.getInputStream());

           //Completar …




            skCliente.close();
        }catch (IOException ex) {
            System.out.println(ex);
        }
    }
    public static void main(String[] args) {
        new Servidor();
    }
}
```

```
import java.io.*;
import java.net.*;
public class Cliente {
    final static String HOST = "localhost";
    final static int PUERTO = 5000;
    public Cliente(){
        boolean terminar = false;
        while(!terminar){
            try{
                System.out.println("Cliente: Esperando conexión con el servidor");
                Socket skCliente = new Socket(HOST,PUERTO);

                DataInputStream flujoEnt = new DataInputStream(skCliente.getInputStream());
                DataOutputStream flujoSal = new DataOutputStream(skCliente.getOutputStream());
                //Completar
                …
                …

                skCliente.close();
                terminar = true;
            }catch (ConnectException ex){
                try{
                    Thread.currentThread().sleep(300);
                }catch (InterruptedException ie){
                    System.out.println(ie);
                }
            } catch (IOException ex){
                System.out.println(ex);
                terminar = true;
            }
        }
    }
    public static void main(String[] args) {
        new Cliente();
    }
```