# Ejercicios prácticos

|  |
| --- |
| **Asignatura**: Acceso a datos |
| **Nombre**: Actividad práctica 1 |
| **Unidades**: 2 |

## Guía

* Se plantean diferentes problemas que tratan aspectos de las unidad 2.
* Para aquellos ejercicios que se pida código fuente se debe poner todo el código de cada clase en formato texto dentro del recuadro.
* Los ejercicios pueden tener más de una solución, buscar la óptima, teniendo en cuenta el número de instrucciones, sentencias y estructuras, etc.
* Es recomendable utilizar de apoyo la documentación oficial de java de Oracle. <https://docs.oracle.com/en/java/>

## Ejemplo

Realiza un programa que sea capaz de leer de un fichero denominado “fichero.txt” y escriba su contenido por consola.

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.FileReader;  **import** java.io.IOException;  **public** **class** Ejemplo {  **public** **static** **void** main(String[] args) **throws** IOException {    FileReader fr = **new** FileReader("fichero.txt");  BufferedReader br = **new** BufferedReader(fr);  String linea;  **while** ((linea = br.readLine()) != **null**) {  System.***out***.println(linea);  }  fr.close();  }  } |

## Enunciado de los ejercicios

## Ejercicio 1.

Realiza un programa que permita a un usuario escribir, leer y borrar el contenido de un fichero, para ello realiza el siguiente menú de usuario:

“1.- Escribir en el fichero

2.- Leer el contenido del fichero

3.- Borrar el contenido del fichero

4.- Salir”

La opción 1 permite escribir al final del fichero el texto que introduzca por teclado el usuario. La opción 2 permite leer todo el fichero y mostrarlo por consola. La opción 3 debe borrar todo lo que hay en el fichero y la opción 4 debe detener el programa.

|  |
| --- |
| package ejercicio1;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Scanner;  public class Ejercicio1 {  public static void main(String[] args) {  String rutaFichero = "Ejercicios/src/ejercicio1/mi\_fichero.txt";  Scanner input = new Scanner(System.in);  int option = 0;  System.out.println("Bienvenido al programa de escritura y modificación de ficheros.\n");  while (option != 4) {  System.out.println("Menú:");  System.out.println("1.- Escribir en el fichero");  System.out.println("2.- Leer el contenido del fichero");  System.out.println("3.- Borrar el contenido del fichero");  System.out.println("4.- Salir");  System.out.print("Ingrese una opción: ");  try {  option = Integer.parseInt(input.nextLine());  } catch (NumberFormatException e) {  System.out.println("Por favor, ingrese un número válido.");  continue;  }  switch (option) {  case 1:  escribirEnFichero(input, rutaFichero);  break;  case 2:  leerFichero(input, rutaFichero);  break;  case 3:  borrarFichero(rutaFichero);  break;  case 4:  System.out.println("Saliendo del programa...");  break;  }  }  input.close();  }  private static void escribirEnFichero(Scanner input, String rutaFichero) {  FileWriter fw = null;  try {  fw = new FileWriter(rutaFichero, true); // Usar true para agregar al final del fichero  System.out.println("Escriba su texto: ");  String textoUsuario = input.nextLine();  fw.write(textoUsuario + "\n"); // Escribir en el fichero  System.out.println("Texto escrito con éxito.");  } catch (IOException e) {  System.out.println("Error al abrir el archivo: " + e.getMessage());  } finally {  try {  if (fw != null) {  fw.close(); // Cerrar el FileWriter  }  } catch (IOException e) {  System.out.println("Error al cerrar el archivo: " + e.getMessage());  }  }  }  private static void leerFichero(Scanner input, String rutaFichero) {  FileReader fr = null;  BufferedReader br = null;  try {  fr = new FileReader(rutaFichero);  br = new BufferedReader(fr);  String linea;  System.out.println("Contenido del fichero:");  while ((linea = br.readLine()) != null) {  System.out.println(linea);  }  } catch (IOException e) {  System.out.println("Error al abrir el documento: " + e.getMessage());  } finally {  try {  if (br != null) {  br.close();  }  } catch (IOException e) {  System.out.println("Error al cerrar el archivo: " + e.getMessage());  }  }  }  private static void borrarFichero(String rutaFichero) {  FileWriter fw = null;  try {  fw = new FileWriter(rutaFichero);  fw.write(""); // Escribir una cadena vacía para borrar el contenido del fichero  System.out.println("Contenido del fichero borrado con éxito.");  } catch (IOException e) {  System.out.println("Error al abrir el archivo: " + e.getMessage());  } finally {  try {  if (fw != null) {  fw.close(); // Cerrar el FileWriter  }  } catch (IOException e) {  System.out.println("Error al cerrar el archivo: " + e.getMessage());  }  }  }  } |

## Ejercicio 2.

Escribe un programa que lea el contenido de un fichero y escriba su contenido al revés en el mismo fichero. Por ejemplo, si el contenido leído es “Esta es una frase de ejemplo” debe escribir “ejemplo de frase una es Esta”

|  |
| --- |
| package ejercicio2;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Arrays;  import java.util.Collections;  public class Ejercicio2 {  public static void main(String[] args) {  String filePath = "/home/edgar/Desktop/Actividad Práctica 1/Ejercicios/src/ejercicio2/fichero.txt";  try {  // Inicializamos StringBuilder con la frase deseada  StringBuilder content = new StringBuilder("Esta es una frase de ejemplo\n");  // Leemos el archivo  try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  String line;  while ((line = reader.readLine()) != null) {  content.append(line).append("\n");  }  }  // Dividimos el contenido en líneas, invertir cada línea y reconstruir  String[] lines = content.toString().split("\n");  StringBuilder reversedContent = new StringBuilder();  for (String ln : lines) {  String[] words = ln.split("\\s+");  Collections.reverse(Arrays.asList(words));  reversedContent.append(String.join(" ", words)).append("\n");  }  // Escritura en el archivo  try (FileWriter writer = new FileWriter(filePath)) {  writer.write(reversedContent.toString().trim());  }  System.out.println("Contenido invertido y escrito en el archivo.");  } catch (IOException e) {  System.out.println("Error al leer o escribir el archivo.");  e.printStackTrace();  }  }  } |

## Ejercicio 3.

Uno de los mayores problemas que existen con la lectura y escritura de datos en ficheros, es la codificación. Existen diferentes codificaciones (ANSI, Latin, ISO 8859, UTF, etc..) por eso nos piden hacer un programa para recodificar un fichero. El programa debe abrir un fichero en una codificación concreta y generar un nuevo fichero con otra codificación que queramos.  
NOTA. Para ver y cambiar la codificación que tiene un fichero se puede utilizar la aplicación Notepad++ desde el menú/Codificación.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

|  |
| --- |
| package ejercicio3;  /\*Uno de los mayores problemas que existen con la lectura y escritura de datos en ficheros, es la codificación. Existen diferentes codificaciones (ANSI, Latin, ISO 8859, UTF, etc..) por eso nos piden hacer un programa para recodificar un fichero. El programa debe abrir un fichero en una codificación concreta y generar un nuevo fichero con otra codificación que queramos.  NOTA. Para ver y cambiar la codificación que tiene un fichero se puede utilizar la aplicación Notepad++ desde el menú/Codificación. \*/  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  public class Ejercicio3 {  public static void main(String[] args) {  String inputFile = "Ejercicios/src/ejercicio3/input.txt"; // Fichero original  String outputFile = "Ejercicios/src/ejercicio3/output.txt"; // Fichero de salida  String inputEncoding = "ISO-8859-1"; // codificación actual del fichero original  String outputEncoding = "UTF-8"; // codificación deseada para el fichero de salida  try (FileInputStream inputStream = new FileInputStream(inputFile);  InputStreamReader reader = new InputStreamReader(inputStream, inputEncoding);  FileOutputStream outputStream = new FileOutputStream(outputFile);  OutputStreamWriter writer = new OutputStreamWriter(outputStream, outputEncoding)) {  // Para leer el contenido del fichero original y escribirlo en el fichero de  // salida  int character;  while ((character = reader.read()) != -1) {  writer.write(character);  }  System.out.println("El fichero ha sido recodificado con éxito.");  } catch (IOException e) {  System.out.println("Ha ocurrido un error al recodificar el fichero.");  e.printStackTrace();  }  }  } |

## Ejercicio 4.

Realiza una aplicación para listar el contenido de una ruta cualquiera. Se debe mostrar una lista de directorios y ficheros que hay en su interior, donde cada línea es un directorio o fichero con el siguiente formato: “drwx Tamaño Nombre”

* d : Indica si es un directorio, si no es un directorio aparecerá un -
* r : Indica si permite la lectura
* w: Indica si permite la escritura
* x: Indica si permite la ejecución
* Tamaño: Es el tamaño en bytes
* Nombre: Es el nombre del fichero o directorio

Un ejemplo de cómo se vería el contenido que hay dentro de la ruta de la carpeta de eclipse

Texto

Descripción generada automáticamente con confianza media

NOTA. Para formatear la información que se imprime por consola se puede utilizar la instrucción *System.out.printf()*

|  |
| --- |
| package ejercicio4;  import java.io.File;  import java.util.Scanner;  public class Ejercicio4 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Introduce la ruta del directorio: ");  String directoryPath = scanner.nextLine();  scanner.close();  File directory = new File(directoryPath);  File[] files = directory.listFiles();  if (files != null) {  for (File file : files) {  String type = file.isDirectory() ? "d" : "-";  String readPermission = file.canRead() ? "r" : "-";  String writePermission = file.canWrite() ? "w" : "-";  String executePermission = file.canExecute() ? "x" : "-";  long size = file.length();  String name = file.getName();  System.out.println(type + "rwx " + size + " " + name);  }  } else {  System.out.println("El directorio no existe o no es accesible.");  }  }  } |

## Ejercicio 5.

La gran mayoría de aplicaciones disponen de ficheros de configuración para especificar los parámetros de arranque, modificar ciertos comportamientos de la aplicación o destinados a otra finalidad. Un ejemplo, es el fichero de configuración *config.ini* del IDE Eclipse.

Texto, Aplicación

Descripción generada automáticamente

Este tipo de fichero se le denominan “properties” o de propiedades porque su contenido sigue un formato concreto. Previamente aparece una propiedad seguido de un igual ‘=’ y después su valor. Los comentarios van precedidos de #.

En este ejercicio vas a tener que buscar como funciona la clase *java.util.Properties* para trabajar con los ficheros properties. Puedes conocer los métodos que tienen desde la documentación oficial. <https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html>

Realiza una aplicación que permita cargar un fichero properties y luego muestre un menú al usuario para realizar las siguientes acciones:

1.- Leer propiedad. Permite leer una propiedad cualquiera según lo que indique el usuario por teclado y mostrar su valor.

2.- Escribir una propiedad y su valor. Permite almacenar una nueva propiedad y su valor.

3.- Guardar y salir. Actualizar el fichero properties con todos los cambios realizados y finalizar la aplicación.

El programa terminará cuando el usuario decida guardar y salir, mientras tanto debe seguir funcionando.

|  |
| --- |
| package ejercicio5;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.util.Properties;  import java.util.Scanner;  public class Ejercicio5 {  public static void main(String[] args) {  Scanner input = new Scanner(System.in);  int option = 0;  System.out.println("Bienvenido al programa de carga y escritura de propiedades.\n");  // Solicitar la ruta del fichero al usuario  System.out.print("Ingrese la ruta del fichero properties: ");  String rutaFichero = input.nextLine();  Properties properties = new Properties();  try (FileInputStream fileInputStream = new FileInputStream(rutaFichero)) {  properties.load(fileInputStream);  } catch (IOException e) {  System.out.println("Error al cargar el fichero properties.");  e.printStackTrace();  return;  } finally {  input.close();  }  while (option != 3) {  System.out.println("Menú:");  System.out.println("1.- Leer propiedad.");  System.out.println("2.- Escribir propiedad y su valor.");  System.out.println("3.- Guardar y salir.");  System.out.print("Ingrese una opción: ");  try {  option = Integer.parseInt(input.nextLine());  } catch (NumberFormatException e) {  System.out.println("Por favor, ingrese un número válido.");  continue;  }  switch (option) {  case 1:  System.out.print("Ingrese el nombre de la propiedad: ");  String propertyName = input.nextLine();  String propertyValue = properties.getProperty(propertyName);  if (propertyValue != null) {  System.out.println("El valor de la propiedad " + propertyName + " es: " + propertyValue);  } else {  System.out.println("La propiedad " + propertyName + " no existe.");  }  break;  case 2:  System.out.print("Ingrese el nombre de la propiedad: ");  String newPropertyName = input.nextLine();  System.out.print("Ingrese el valor de la propiedad: ");  String newPropertyValue = input.nextLine();  properties.setProperty(newPropertyName, newPropertyValue);  System.out.println("Propiedad agregada con éxito.");  break;  case 3:  try (FileOutputStream fileOutputStream = new FileOutputStream(rutaFichero)) {  properties.store(fileOutputStream, "Propiedades actualizadas");  System.out.println("Fichero properties actualizado y guardado con éxito.");  } catch (IOException e) {  System.out.println("Error al guardar el fichero properties.");  e.printStackTrace();  }  break;  default:  System.out.println("Opción inválida. Por favor, ingrese una opción válida.");  break;  }  }  input.close();  }  } |

## Ejercicio 6.

Elabora un programa que sea capaz de escribir y leer sobre un fichero de forma secuencial indexado. El acceso secuencial indexado es aquel que en un fichero se guarda la información y en otro fichero se guarda un índice con las posiciones donde se encuentra cada elemento.

Según nos indican el programa tiene que guardar la información de los alumnos y sus notas en un fichero denominado alumnos.dat, y a partir de un identificador de alumno tiene que devolver el alumnos que corresponde, de esta forma nos ahorramos buscar por nombre de alumno y que las búsquedas sean más eficientes. A continuación tenemos una representación de como debe estar estructurada la información en los dos ficheros:

|  |  |  |
| --- | --- | --- |
|  | Nombre | Nota |
| 0x0001 | Juan Perez | 5 |
| 0x0002 | Maria González | 7 |
| 0x0003 | Carlos Rodríguez | 3 |
| Fichero con la información. *alumnos.dat* | | |

|  |  |
| --- | --- |
| Identificador | Puntero |
| 1 | 0x0001 |
| 2 | 0x0002 |
| 3 | 0x0003 |
| Fichero con los índices. *indice.dat* | |

Nuestro analista nos facilita el trabajo y nos indica como tenemos que realizar el programa. Según los requisitos es necesario lo siguiente:

1. Elaborar un proceso de escritura inicial para introducir los datos en el fichero alumnos.dat, a la vez actualizar y escribir en el fichero índice.dat la posición y puntero. Con el método *getFilePointer()* de la clase *RandomAccessFile* es posible obtener el puntero de la ubicación que se encuentra en el fichero.
2. Elaborar un proceso de lectura que a partir de un identificador de un alumno muestre sus datos, nombre y nota. Se debe consultar previamente el fichero índice y luego acceder al fichero de los datos en la posición indicada en el puntero. Con el método *seek(Puntero)* de la clase *RandomAccessFile* es posible ubicarnos en la parte del fichero en el puntero indicado.

Para complicar la situación, nos han pedido que el fichero índice sea consultado o escrito en una sola operación, esto quiere decir que no podemos estar accediendo y consultando varias veces sobre el fichero índice. Para ello, busca en Java que estructura permite tener en memoria una colección, clave:valor. Con el método *writeObject()* y *readObject()* de la clase *ObjectInputStream* es posible obtener escribir y leer objetos de un fichero.

|  |
| --- |
| package ejercicio6;  import java.io.\*;  import java.util.HashMap;  import java.util.Map;  import java.util.Scanner;  public class Ejercicio6 {  private static final String ALUMNOS\_FILE = "alumnos.dat";  private static final String INDICE\_FILE = "indice.dat";  public static void main(String[] args) {  Scanner input = new Scanner(System.in);  try {  escribirDatosIniciales();  System.out.print("Ingrese el identificador del alumno: ");  int identificador = input.nextInt();  leerDatosPorIdentificador(identificador);  } catch (IOException | ClassNotFoundException e) {  e.printStackTrace();  }  input.close();  }  private static void escribirDatosIniciales() throws IOException {  try (RandomAccessFile alumnosFile = new RandomAccessFile(ALUMNOS\_FILE, "rw");  ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream(INDICE\_FILE))) {  Map<Integer, Long> indiceMap = new HashMap<>();  escribirAlumno(alumnosFile, "Juan Perez", 5, indiceMap);  escribirAlumno(alumnosFile, "Maria González", 7, indiceMap);  escribirAlumno(alumnosFile, "Carlos Rodríguez", 3, indiceMap);  objectOutputStream.writeObject(indiceMap);  }  }  private static void escribirAlumno(RandomAccessFile alumnosFile, String nombre, int nota,  Map<Integer, Long> indiceMap) throws IOException {  long posicionActual = alumnosFile.getFilePointer();  indiceMap.put(indiceMap.size() + 1, posicionActual);  alumnosFile.writeUTF(nombre);  alumnosFile.writeInt(nota);  indiceMap.put(indiceMap.size(), posicionActual);  }  private static void leerDatosPorIdentificador(int identificador) throws IOException, ClassNotFoundException {  try (RandomAccessFile alumnosFile = new RandomAccessFile(ALUMNOS\_FILE, "r");  ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream(INDICE\_FILE))) {  Map<Integer, Long> indiceMap = (Map<Integer, Long>) objectInputStream.readObject();  if (indiceMap.containsKey(identificador)) {  long posicion = indiceMap.get(identificador);  alumnosFile.seek(posicion);  if (alumnosFile.getFilePointer() < alumnosFile.length()) {  System.out.println("Identificador: " + identificador);  System.out.println("Nombre: " + alumnosFile.readUTF());  System.out.println("Nota: " + alumnosFile.readInt());  } else {  System.out.println("No hay datos disponibles");  }  } else {  System.out.println("Identificador no encontrado");  }  }  }  } |