Ejercicios:

**Ejercicio 1:**

Crea la clase producto con los siguientes atributos: id (int), nombre (String), precio (double), descuento (boolean) y tipo (char)

Crea 5 o 6 productos e introdúcelos en una lista.

Escribe con RandomAccessFile un fichero con los siguientes datos de los productos:

* id (int)
* nombre (String, 10 caracteres)
* precio (double)
* descuento (boolean)
* tipo (char)

b) lee el archivo de acceso aleatorio y muestra su contenido por pantalla.

**Ejercicio 2:**

Buscar y reemplazar texto en un archivo:

Objetivo: Crear una herramienta que busque una cadena de texto específica en un archivo y la reemplace por otra.

Consideraciones:

Utilizar java.io.BufferedReader y java.io.BufferedWriter para leer y escribir líneas de texto.

Implementar un algoritmo de búsqueda y reemplazo eficiente.

Crear un nuevo archivo con los cambios para preservar el original.

**Ejercicio 3:**

Crea una lista de objetos Persona (con atributos nombre y edad) y:

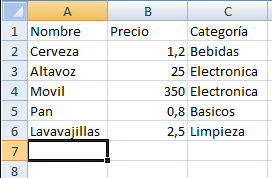
* Calcula la edad promedio de las personas.
* Encuentra la persona más joven.
* Imprime una lista con los nombres de las personas mayores de 30 años

**Ejercicio 4:**

Lee un archivo CSV con datos de productos (nombre, precio, categoría) utiliza el contenido leído para inicializar un arraylist:

* Calcula el producto más caro de cada categoría.
* Imprime una lista con los productos cuyo precio está entre 10 y 20 euros.

Ejemplo del CSV descrito:



**Ejercicio 5:**

Crea un String con un contenido bastante largo. Ahora trabaja con dicho String para obtener por medio de stream el número de ocurrencias de cada palabra.

Consejo: Si quieres simplificarlo mira los métodos de la clase Collectors

Ejemplo:

String text = "este es un ejemplo de texto para contar palabras este texto puede ser todo lo largo que quieras";

{que=1, ser=1, puede=1, de=1, lo=1, es=1, todo=1, texto=2, este=2, para=1, palabras=1, un=1, quieras=1, largo=1, contar=1, ejemplo=1}

**Ejercicio 6:** (realiza antes el ejercicio 8) **Examen 2022.**

Tenemos una clase coche con al menos los siguientes atributos: marca, color, matrícula. Y una clase Persona con al menos los siguientes atributos: nombre, dni y una lista de coches de su propiedad.

Realiza las siguientes búsquedas:

* Muestra la información de las personas que tiene un coche rojo
* Personas con un coche Opel.
* Encontrar a la persona con más coches.

**Ejercicio 7: Examen 2022.**

Tenemos una clase persona con los siguientes atributos:

int id**;**

String nombre**;**

String apellidos**;**

String curso**;**

int nota**;**

int edad**;**

Para inicializar los datos, tienes un archivo “inicialización.java” que añade 1000 personas a una lista. Una vez inicializada tu lista realiza las siguientes búsquedas haciendo uso de programación funcional:

* Personas mayores de edad, que están cursando un curso que empieza por A y su nombre contiene una N
* Personas con edades entre 20 y 25 años y que estén cursando Acceso a Datos
* 3. Muestra las personas que han aprobado el curso de POO
* 4. Muestra el número de personas que hay matriculadas en cada curso.

**Ejercicio 8: Madrid 2019**

Crea un modelo (POJO) con al menos los siguientes atributos:

public class Cancion **{**

String titulo**;**

String cantante**;**

**}**

Crea una lista de canciones y utiliza el siguiente código para inicializarla:

canciones **=** **new** ArrayList**<>(**

Arrays**.**asList**(**

**new** Cancion**(**"Livin' on Prayer"**,** "Bon Jovi"**),**

**new** Cancion**(**"Long Hot Summer"**,** "Keith Urban"**),**

**new** Cancion**(**"It's my Life"**,** "Bon Jovi"**),**

**new** Cancion**(**"Dolor Fantasma"**,** "Amadeus"**),**

**new** Cancion**(**"Run To You"**,** "Bryan Adams"**),**

**new** Cancion**(**"Summer of 69"**,** "Bryna Adams"**),**

**new** Cancion**(**"Paranoid"**,** "Black Sabbath"**),**

**new** Cancion**(**"Cherokee"**,** "Europe"**),**

**new** Cancion**(**"River Bank"**,** "Brad Paisley"**)**

**));**

Realiza las siguientes consultas:

* Busca las canciones de “Bon Jovi” usando programación tradicional (bucles)
* Busca las canciones de “Bon Jovi” usando programación funcional.
* Busca las canciones de “Bon Jovi” usando programación funcional y las canciones encontradas deben acabar en nueva lista.
* Cuenta el número de canciones que tiene “Bon Jovi” en la lista.
* Realiza una agrupación por cantante y muestra el número de canciones que tiene cada cantante.
* Por último introduce una canción existente de nuevo en la lista, por ejemplo: canciones**.**add**(new** Cancion**(**"Summer of 69"**,** "Bryan Adams"**));** Como podrás observar ahora existe un duplicado, Queremos imprimir la información de todas las canciones, pero sin dicho duplicado.

**Ejercicio 9: Madrid 2020**

Crea un modelo (POJO) que represente un estudiante con los siguientes atributos:

private int id**;**

private String dni**;**

private String nombre**;**

private String apellidos**;**

private String nombreCurso**;**

private double nota**;**

private int edad**;**

Carga los siguientes alumnos en una lista:

// Cargamos la lista de Alumnos

listaAlumnos**.**add**(new** Alumno**(**1**,** "1717213183"**,** "Javier"**,** "Molina Cano"**,** "Java 8"**,** 7**,** 28**));**

listaAlumnos**.**add**(new** Alumno**(**2**,** "1717456218"**,** "Ana"**,** "Gómez Álvarez"**,** "Java 8"**,** 10**,** 33**));**

listaAlumnos**.**add**(new** Alumno**(**3**,** "1717328901"**,** "Pedro"**,** "Marín López"**,** "Java 8"**,** 8.6**,** 15**));**

listaAlumnos**.**add**(new** Alumno**(**4**,** "1717567128"**,** "Emilio"**,** "Duque Gutiérrez"**,** "Java 8"**,** 10**,** 13**));**

listaAlumnos**.**add**(new** Alumno**(**5**,** "1717902145"**,** "Alberto"**,** "Sáenz Hurtado"**,** "Java 8"**,** 9.5**,** 15**));**

listaAlumnos**.**add**(new** Alumno**(**6**,** "1717678456"**,** "Germán"**,** "López Fernández"**,** "Java 8"**,** 8**,** 34**));**

listaAlumnos**.**add**(new** Alumno**(**7**,** "1102156732"**,** "Oscar"**,** "Murillo González"**,** "Java 8"**,** 10**,** 32**));**

listaAlumnos**.**add**(new** Alumno**(**8**,** "1103421907"**,** "Antonio Jesús"**,** "Palacio Martínez"**,** "PHP"**,** 9.5**,** 17**));**

listaAlumnos**.**add**(new** Alumno**(**9**,** "1717297015"**,** "César"**,** "González Martínez"**,** "Java 8"**,** 8**,** 26**));**

listaAlumnos**.**add**(new** Alumno**(**10**,** "1717912056"**,** "Gloria"**,** "González Castaño"**,** "PHP"**,** 10**,** 28**));**

listaAlumnos**.**add**(new** Alumno**(**11**,** "1717912058"**,** "Jorge"**,** "Ruiz Ruiz"**,** "Python"**,** 8**,** 22**));**

listaAlumnos**.**add**(new** Alumno**(**12**,** "1717912985"**,** "Ignacio"**,** "Duque García"**,** "Java Script"**,** 9.4**,** 32**));**

listaAlumnos**.**add**(new** Alumno**(**13**,** "1717913851"**,** "Julio"**,** "González Castaño"**,** "C Sharp"**,** 10**,** 22**));**

listaAlumnos**.**add**(new** Alumno**(**14**,** "1717986531"**,** "Gloria"**,** "Rodas Carretero"**,** "Ruby"**,** 7**,** 18**));**

listaAlumnos**.**add**(new** Alumno**(**15**,** "1717975232"**,** "Jaime"**,** "Jiménez Gómez"**,** "Java Script"**,** 10**,** 18**));**

Realiza las siguientes consultas con programación funcional:

* Muestra todos los alumnos: Lista de Alumnos🡪 debes usar una referencia a método.
* Alumnos cuyo apellido empiezan con el caracter L u G
* Número de Alumnos
* Alumnos con nota mayor a 9 y que sean del curso PHP
* Imprimir los 2 primeros Alumnos de la lista
* Imprimir el alumno con menor edad
* Imprimir el alumno con mayor edad
* Encontrar el primer Alumno
* Alumnos que tienen un curso en el que el nombre contienen la A
* Alumnos en que la longitud de su nombre es mayor a 10 caracteres
* Obtiene los alumnos en los cuales el nombre del curso empieza con el caracter 'P' y la longitud sea <= a 6
* Crea una nueva lista llamada “listaNueva” con el contenido de la consulta anterior.

**Ejercicio 10: Madrid 2021**

Podrás observar en el repositorio un archivo csv llamado “product.csv” debes crear un POJO que represente un producto. Y cargar todos los productos del archivo en una lista y realiza las siguientes consultas usando programación funcional:

* Imprime la lista de productos.
* Realiza el equivalente a un select name from productos.
* Imprime el nombre de los productos cuyo stock sea menos a 10
* Imprime el nombre de los productos cuyo stock sea menor a 10, pero ordenado por número de el número de stock de menor a mayor (ayuda: para esta consulta es probable que tengas que usar el método sorted, este método recibe un Comparator. Ésta misma interfaz Comparator tiene algunos métodos que nos serán de gran ayuda)
* Realiza la misma consulta anterior pero ahora ordenando de mayor a menor.
* Muestra el nombre de los productos con unidades en stock mayor de 10 ordenados ordenar por unidad de stock de forma descendente y por nombre de producto de forma ascendente.
* Muestra el nombre de los productos con unidades en stock mayor de 10 ordenados ordenar por unidad de stock de forma ascendente y por nombre de producto de forma descendente.
* Obtener el número de productos agrupados por proveedor.
* Obtener la suma del precio unitario de todos los productos agrupados por el número de existencias en el almacén, pero solo obtener aquellos registros cuya suma sea mayor a 100
* Calcula el promedio de existencias en almacén.
* Producto con el precio unitario más alto.
* Imprime la lista de productos, pero limitando el número de productos devueltos a 50 (muestra los 50 primeros, operador limit en sql)