Práctico 10 – Genéricos, Lamdas y Stream

Ejercicio 1

Implementar una clase genérica **Terna**, la cual representa una agrupación de tres elementos.
Recordar que se utilizará el argumento genérico T para el tipo de dato genérico.
Además deberá crear dos constructores, uno que inicializa todo en null, y el otro que inicializa a los 3 elementos que se le pasarán como argumentos. Los dos métodos, el get y el set, deberán tirar una excepción *(Exception)* en caso que se quiera obtener o modificar una posición no existente *(que no esté entre 1 y 3, ya que son 3 elementos)*
Y por último deberá implementarle un método toString, que devuelva un string de la forma "Terna: " + elemento1 + ", " + elemento2 + ", " + elemento3

Hacer pruebas luego en un main de otra clase, creando ternas de Integer por un lado y ternas de Strings por otro lado, por ejemplo.

Ejercicio 2

Implementar un método genérico llamado intercambio, que intercambie dos elementos de cualquier tipo.

Ejercicio 3

Implementar un método genérico llamado contarOcurrencias, que tome como argumentos un array del tipo genérico T y un elemento del tipo T, y devuelva un int que represente la cantidad de veces que el elemento está en el array.
Considerar cuando el elemento sea null. Y utilizar equals para hacer la comparación, ya que lo tendrá que considerar como objetos.

Probarlo para un array con números y otro con strings.

Ejercicio 4

Dado un array que haya declarado y llenado con algunos elementos, sea String o Integer *(no int)* y teniendo algunos elementos duplicados, deberá escribir el código que cree un conjunto a partir de ese array. Esto puede servir para quitar duplicados de un array por ejemplo.
Deberá utilizar, de la colección Set, en particular **HashSet**. Mostrar en pantalla el resultado.

Ejercicio 5

A la inversa que en el punto anterior, dado un conjunto de números Integer o de cadenas String, escribir un código que genere un array basado en ese conjunto. Recuerde no utilizar tipos de datos primitivos como int. Luego muestre en pantalla utilizando Arrays.toString(miarreglo)

Ejercicio 6

Implementar un clase llamada **Password** que cumpla las siguientes condiciones:
Debe tener los atributos **longitud** y **pwd**. Por defecto la longitud será de 8.
Tendrá dos constructores, el por defecto y otro que tomará una longitud para crear una contraseña aleatoria de esa longitud.
Tendrá los siguientes métodos:
esFuerte() que devolverá true o false según si la contraseña es fuerte o no *(para que sea fuerte debe tener más de 2 mayúsculas, al menos una minúscula, al menos 5 números)*
generatePwd() que generará una contraseña aleatoria, de la longitud que tenga.
Los Get, Un Set para longitud ***(NO para para pwd)***.

Ahora, en otro programa con un main, crea un arreglo de contraseñas, con un tamaño que se le indique por teclado.
Al crear cada contraseña que se irá guardando en ese arreglo, pide al usuario la longitud de la misma.
Crea además otro array, del mismo tamaño que el arreglo de contraseñas.

Este segundo array, guardará en cada posición si la contraseña del otro array en esa misma posición es fuerte o no. (Es decir, será un arreglo de booleanos)
Muestra en pantalla, al final, todas las contraseñas y si son fuertes o no.
Guarda en un archivo de texto, solo aquellas contraseñas fuertes.

Ejercicio 7
Crear a mano un archivo de texto y guardar contraseñas inventadas, una por línea.
Utilizando la clase del punto anterior *(modificarla si lo cree necesario)* revisar cada contraseña del archivo de texto y determinar cuáles son fuertes, y cuantas son en total las contraseñas fuertes.

Uso de Expresiones Lambda y Stream

Ejercicio 8
Considerando el siguiente código que muestra en pantalla todos los números de una lista, escriba el código equivalente utilizando expresiones lambda.

```
List<Integer> lista = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);

for (Integer n : lista) {
    System.out.print(n + " ");
}
```

Ejercicio 9
Considerando la misma lista de números del punto anterior, mostrar en pantalla los números mayores a 5, utilizando expresiones lambda y stream.

Ejercicio 10
Considerando la misma lista de números del punto 8, mostrar en pantalla solo los números pares, utilizando expresiones lambda y stream.

Ejercicio 11
Dada una lista de números, con números aleatorios enteros, entre 1 y 50, mostrarla ordenada, utilizando expresiones lambda y stream.

Ejercicio 12
¿Recuerda el ejercicio 4 donde se utilizaba la conversión de un arraylist hacia un conjunto para quitar duplicados? ¿Cómo haría ahora para generar una nueva lista, quitando todo duplicado de una lista de números por ejemplo?

Ejercicio 12
Crear la clase Alumno, con los siguientes campos

```
private int legajo;
private String dni;
private String nombre;
private String apellido;
private String curso;
private double nota;
private int edad;
```

Con todos sus métodos, incluido un método toString *(que sea un Override del String toString, como siempre)*
Utilizar esta clase Alumno en otra clase, para crear una lista de 20 alumnos *(utilizando ArrayList)*

Realizar los siguientes puntos, utilizando expresiones lambda y Streams para resolverlos:

a. Imprimir en pantalla todos los alumnos.
b. Imprimir en pantalla todos los alumnos cuyo apellido empiece con la letra P.
c. Mostrar todos los alumnos con nota mayor a 9
d. Obtener el alumno de menor edad del curso.
e. Obtener el alumno de mayor edad.
f. Mostrar los alumnos cuyos nombres terminen con la letra o.
g. Mostrar los alumnos cuyo apellido sea de más de 10 caracteres.
h. Generar una nueva lista con todos los alumnos que tienen nota menor a 6.
i. Generar una nueva lista con todos los alumnos con nota 10 en el curso de Java.
j. Generar una nueva lista con todos los alumnos cuyos nombres empiecen con la letra A y cuya longitud sea menor a 8.
k. Mostrar la cantidad total de alumnos.
l. Mostrar los alumnos ordenados por apellido.