Para realizar este ejercicio se ha de utilizar el código proporcionado.

Se trata de un solo ejercicio compuesto por varias partes.

Los datos ya están introducidos en el programa y las clases creadas, listo para empezar a realizar los ejercicios sin perder tiempo.

IMPORTANTE: Puedes utilizar cualquier forma para resolver estos ejercicios. Lo importante, aparte de aprender/practicar, es poder realizarlos. Se han realizado los ejercicios principalmente con lambdas, pero no hay ningún motivo por el cual no puedas utilizar cualquier otra forma para resolverlo. Se libre.

Explicación:

Clases.

(Abstracta) Persona. (DNI y Nombre)

- Estudiante (**id** y ***Map*** de asignaturas (Nombre y **Nota**))

- Profesor (**id** y fijo ***(BOOLEAN)***)
  Id -> **Integer**, **Nota** -> **Float**. **(AVISO: No es lo mismo que int y float.)**

Instituto. **Atributos ->**

- Director (Clase **Profesor**)
- **Map** de Estudiantes (id y **Estudiante**)
- **Map** de Profesores (id y **Profesor**)

Otros.

GeneradordeDatos.java -> Este archivo contiene las funciones lambda con todos los datos para realizar los diferentes ejercicios. **No es necesario manipular este fichero.** En la primera línea del archivo main, se llama a la función principal que nos proporcionará todos los datos.

```
Map<Integer, Instituto> mapdeInstitutos = new
HashMap<>(GeneradordeDatos.listaInstitutos.get());
```

Ejercicios.

Ejercicio 1.

Al entrar al Main, hay una línea que imprime el mapadeInstitutos.

```
//  Inicio ----->
mapadeInstitutos.entrySet().forEach(System.out::println);
```

Pero el resultado final es el siguiente:

```
1=El Instituto con director: La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 true, lista de profesores: {20=La persona con dni: 12839475B, nombre: Pedro es profesor con id: 20 true, 21=La persona
2=El Instituto con director: La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 true, lista de profesores: {32=La persona con dni: 29384756T, nombre: Victor es profesor con id: 32 true, 33=La persona
3=El Instituto con director: La persona con dni: 00000003L, nombre: Pedro es profesor con id: 3 true, lista de profesores: {48=La persona con dni: 76543215E, nombre: Joaquin es profesor con id: 48 true, 49=La perso
```

El objetivo es mostrar los datos en un buen formato, pero no podemos modificar las clases. Para ello deberemos iterar la lista del main y darle el formato que buscamos. Además, para finalizar, deberemos **exportarlo a un archivo TXT** para poder visualizarlo mejor. Por ejemplo Informe.txt

```
Instituto con id: 1
La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 es fijo
Lista de profesores:
La persona con dni: 12839475B, nombre: Pedro es profesor con id: 20 es fijo
La persona con dni: 36472819H, nombre: Sofía es profesor con id: 21 es fijo
La persona con dni: 94857362V, nombre: Javier es profesor con id: 22 no es fijo
La persona con dni: 38475961L, nombre: Marta es profesor con id: 23 es fijo
La persona con dni: 29837564G, nombre: Raul es profesor con id: 24 es fijo
La persona con dni: 84736291X, nombre: Paula es profesor con id: 25 es fijo
La persona con dni: 52647382W, nombre: Sergio es profesor con id: 26 no es fijo
La persona con dni: 63728459R, nombre: Elena es profesor con id: 27 es fijo
La persona con dni: 19283746D, nombre: Diego es profesor con id: 28 es fijo
La persona con dni: 76543219P, nombre: Alicia es profesor con id: 29 es fijo
Lista de alumnos:
La persona con dni: 36458358G, nombre: Luca es estudiante con id: 1 y asignaturas y notas: {Matematicas=8.4, Educacion Fisica=6.7, Historia=2.3, Geografia=8.7, Programacion=7.1, Fisica=3.8}
La persona con dni: 28374651H, nombre: Nia es estudiante con id: 2 y asignaturas y notas: {Matematicas=9.2, Educacion Fisica=8.8, Historia=6.1, Geografia=3.4, Programacion=7.5, Fisica=5.7}
La persona con dni: 84726319J, nombre: Santiago es estudiante con id: 3 y asignaturas y notas: {Matematicas=6.9, Educacion Fisica=8.5, Historia=9.7, Geografia=7.3, Programacion=9.8, Fisica=1.8}
La persona con dni: 37492015K, nombre: Valeria es estudiante con id: 4 y asignaturas y notas: {Matematicas=3.7, Educacion Fisica=1.1, Historia=6.5, Geografia=7.4, Programacion=3.9, Fisica=4.2}
La persona con dni: 83749206L, nombre: Mateo es estudiante con id: 5 y asignaturas y notas: {Matematicas=2.0, Educacion Fisica=3.5, Historia=8.7, Geografia=1.7, Programacion=3.4, Fisica=8.5}
La persona con dni: 94736281M, nombre: Sofía es estudiante con id: 6 y asignaturas y notas: {Matematicas=7.4, Educacion Fisica=9.8, Historia=6.2, Geografia=8.1, Programacion=5.9, Fisica=4.4}
La persona con dni: 10293847N, nombre: Martín es estudiante con id: 7 y asignaturas y notas: {Matematicas=1.6, Educacion Fisica=4.0, Historia=5.9, Geografia=8.7, Programacion=2.0, Fisica=2.8}
La persona con dni: 56473829O, nombre: Camila es estudiante con id: 8 y asignaturas y notas: {Matematicas=5.7, Educacion Fisica=9.0, Historia=8.6, Geografia=1.5, Programacion=4.5, Fisica=9.4}
La persona con dni: 83749205P, nombre: Darío es estudiante con id: 9 y asignaturas y notas: {Matematicas=3.1, Educacion Fisica=2.8, Historia=8.7, Geografia=6.4, Programacion=8.0, Fisica=6.9}
La persona con dni: 38475612Q, nombre: Ariana es estudiante con id: 10 y asignaturas y notas: {Matematicas=5.4, Educacion Fisica=3.9, Historia=6.1, Geografia=5.8, Programacion=4.3, Fisica=2.7}
La persona con dni: 93847561R, nombre: Emiliano es estudiante con id: 11 y asignaturas y notas: {Matematicas=8.3, Educacion Fisica=6.1, Historia=4.0, Geografia=7.4, Programacion=3.1, Fisica=8.0}
La persona con dni: 83749204S, nombre: Isabella es estudiante con id: 12 y asignaturas y notas: {Matematicas=4.7, Educacion Fisica=3.0, Historia=9.6, Geografia=5.3, Programacion=5.4, Fisica=1.1}
La persona con dni: 64738291T, nombre: Gabriel es estudiante con id: 13 y asignaturas y notas: {Matematicas=7.2, Educacion Fisica=5.0, Historia=2.7, Geografia=3.5, Programacion=8.6, Fisica=6.0}
La persona con dni: 94857362U, nombre: Luciana es estudiante con id: 14 y asignaturas y notas: {Matematicas=9.1, Educacion Fisica=1.9, Historia=3.8, Geografia=8.3, Programacion=5.7, Fisica=8.9}
La persona con dni: 28374658V, nombre: Sebastián es estudiante con id: 15 y asignaturas y notas: {Matematicas=4.1, Educacion Fisica=5.3, Historia=1.1, Geografia=3.2, Programacion=5.1, Fisica=6.6}
La persona con dni: 48392017W, nombre: Catalina es estudiante con id: 16 y asignaturas y notas: {Matematicas=3.9, Educacion Fisica=6.2, Historia=7.6, Geografia=8.2, Programacion=5.6, Fisica=5.9}
La persona con dni: 39485712X, nombre: Bruno es estudiante con id: 17 y asignaturas y notas: {Matematicas=2.2, Educacion Fisica=9.0, Historia=3.6, Geografia=1.5, Programacion=3.7, Fisica=8.6}
La persona con dni: 29384751Y, nombre: Renata es estudiante con id: 18 y asignaturas y notas: {Matematicas=1.9, Educacion Fisica=8.3, Historia=1.7, Geografia=8.0, Programacion=9.4, Fisica=1.6}
La persona con dni: 10293846Z, nombre: Thiago es estudiante con id: 19 y asignaturas y notas: {Matematicas=7.7, Educacion Fisica=4.5, Historia=9.5, Geografia=7.0, Programacion=3.8, Fisica=1.5}

Instituto con id: 2
La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 es fijo
Lista de profesores:
La persona con dni: 29384756T, nombre: Victor es profesor con id: 32 es fijo
La persona con dni: 37584921Z, nombre: Patricia es profesor con id: 33 es fijo
La persona con dni: 84716253L, nombre: David es profesor con id: 34 es fijo
La persona con dni: 56473892N, nombre: Sara es profesor con id: 35 no es fijo
La persona con dni: 90817254S, nombre: Pablo es profesor con id: 36 es fijo
```

Ahora, sin embargo, el resultado ha mejorado, pero hay que hacer alguna modificación.

1. Habría que modificar en el fichero Profesor.java, en el método toString(). Habría que convertir el true o false en “es fijo” o “no es fijo”. **Intenta NO** usar el método tradicional:

   If (this.fijo) {String fijo = “es fijo”;}

   Pista oculta :

   Busca una forma *funcional* y sencilla.

```
public String toString() {
    // Ejercicio 1.
    // Busca una manera eficiente de convertir el boolean "fijo" en: "es fijo" o "no es fijo".

    return super.toString()+" es profesor con id: "+this.id+" "+;this.fijo;
}
```

Ejercicio 2.

Los institutos han decidido comprobar la base de datos en común. Para ello, han decidido comprobar que pueden obtener el id del instituto, su director, el numero de alumnos y profesores.

Ejemplo:

```
Instituto con id 1 y Director:
La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 es fijo
Estudiantes: 19
Profesores: 10

Instituto con id 2 y Director:
La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 es fijo
Estudiantes: 20
Profesores: 10

Instituto con id 3 y Director:
La persona con dni: 00000003L, nombre: Pedro es profesor con id: 3 es fijo
Estudiantes: 20
Profesores: 10
```

Ejercicio 3.

Para este ejercicio, te proporciono un código que tendrás que añadir a tu código. Estas líneas son fijas, no se pueden modificar para realizar el ejercicio.

```
List<Instituto> listaFusion = new
ArrayList<>(GeneradordeDatos.listaInstitutos.get().values().stream().t
oList());

listaFusion.addAll(GeneradordeDatos.listaInstitutos.get().values().str
eam().toList());
```

Este código genera una lista de institutos duplicada.

Consejo, imprime al director del instituto para poder ver que es lo que imprime, pero sin imprimir todo el instituto, al contener muchos datos no se visualizará bien.

El objetivo es utilizar distinct para imprimir únicamente aquellos que no se repiten. Sin embargo, podrás comprobar que al usar distinct no elimina los duplicados.

Para ello deberás comprender como funciona distinct y que es lo que necesita para poder comparar si son ***iguales*** los institutos de la lista.

Puedes resolverlo como quieras, pero la idea es que solo utilizando distinct deberías poder lograrlo, solo debes buscar que falla en el código proporcionado para comparar.

Sin funcionar:

```
Antes:
La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 es fijo
La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 es fijo
La persona con dni: 00000003L, nombre: Pedro es profesor con id: 3 es fijo
La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 es fijo
La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 es fijo
La persona con dni: 00000003L, nombre: Pedro es profesor con id: 3 es fijo
```

Objetivo:

```
Despues:
La persona con dni: 00000003L, nombre: Pedro es profesor con id: 3 es fijo
La persona con dni: 00000002L, nombre: Luna es profesor con id: 2 es fijo
La persona con dni: 00000001L, nombre: Elena es profesor con id: 1 es fijo
```

Ejercicio 4.

Los siguientes ejercicios pueden resultar complejos si no analizamos bien los pasos a realizar, debido a la cantidad de datos anidados y el uso repetido de Maps.

Antes de continuar con el ejercicio 1, tenemos otro problema.

```
{Matematicas=5.031442404, Educacion Fisica=9.014974,
Historia=6.578206, Geografia=8.834224, Programacion=6.05507928,
Fisica=7.0976963}
```

Las notas de las asignaturas tienen demasiados decimales, modifica el mapadeInstitutos y cambia las notas de todos los alumnos a un valor que como máximo tenga un decimal. Recuerda guardar los cambios en el mapadeInstitutos para que se refleje en el archivo txt del final.

```
{Matematicas=5.0, Educacion Fisica=9.0, Historia=6.6, Geografia=8.8,
Programacion=6.1, Fisica=7.1}
```

La idea de este ejercicio es manipular Maps encadenados. Para el redondeo puedes usar Math o hacer operaciones. Recuerda que el valor de la nota es Float. Si al operar los datos te devuelve siempre el primer digito decimal a 0, prueba a hacer las operaciones con números declarados Float. Ejemplo: Usar 10f o 10.0 en vez de 10.

Si quieres centrarte en manipular los datos y no en operar los decimales puedo revelarte el método que he realizado para obtener un decimal.

Pista Invisible:

Ejercicio 5.

AVISO: Las notas de los alumnos son aleatorias, pueden darse casos en los cuales ningún alumno cumpla la condición, para probar que funciona puedes cambiar los criterios o ejecutar hasta que se cumpla la condición.

Los Institutos han decidido premiar a aquellos que han obtenido buenos resultados, para ello debemos filtrar a los alumnos dependiendo las notas.

Solo premiaran a aquellos alumnos que hayan logrado obtener un 9 o más en alguna asignatura, pero además deben haber obtenido un 5 o más en cada una de sus asignaturas.

EJEMPLO:

Alumno1 Notas -> 5.5, 7.6 , 8.9 -> No apto.

Alumno2 Notas -> 5.5, 9.2 , 5 -> Apto.

Alumno3 Notas -> 4.3, 10.0 , 10.0 -> No apto.

Para finalizar exporta toda la información de cada alumno que será recompensado a un fichero: AlumnosPremiados.txt

Resultado: **(Puede variar, las notas se generan aleatoriamente en cada ejecución).**

```
La persona con dni: 83749205P, nombre: Dario es estudiante con id: 9 y asignaturas y notas:
{Matematicas=7.7, Historia=8.0, Educacion Fisica=9.5, Geografia=6.0, Programacion=7.2,
Fisica=7.5}


La persona con dni: 56473827W, nombre: Liliana es estudiante con id: 42 y asignaturas y
notas: {Matematicas=9.6, Historia=9.5, Educacion Fisica=5.1, Geografia=9.2, Programacion=
9.5, Fisica=8.3}
```

Ejercicio 6.

Los institutos quieren participar en una competencia para descubrir cual logra mejores resultados. El objetivo es obtener la media de cada instituto por cada asignatura. Para ello deberemos comprobar todas las notas de todos los estudiantes de cada instituto por cada asignatura.

Además, podríamos obtener las medias totales del instituto. Mediante la suma de las medias de cada asignatura.

Ejemplo:

```
Medias del Instituto con id: 1
Educacion Fisica 5.6
Fisica 5.2
Geografia 5.9
Historia 5.9
Matematicas 5.4
Programacion 5.4

Total: 5.5666666


Medias del Instituto con id: 2
Educacion Fisica 7.0
Fisica 5.3
Geografia 5.1
Historia 5.6
Matematicas 5.7
Programacion 6.7

Total: 5.9


Medias del Instituto con id: 3
Educacion Fisica 5.1
Fisica 5.6
Geografia 5.8
Historia 5.4
Matematicas 7.1
Programacion 5.4

Total: 5.7333336
```

Ejercicio 1.

Antes de continuar al final, debemos comprobar que el ejercicio 1 se muestra parecido al archivo Info.txt proporcionado en la carpeta comprimida.

Ejercicio 7.

Final.

Para terminar, un ejercicio sencillo. Para realizar el ejercicio hay que copiar el código proporcionado:

```
//Ejercicio 7
Map<String, Integer> oculto = new
HashMap<>(GeneradordeDatos.finalizar.get());
```

Este código nos proporcionara un map con letras y números.

El objetivo es ordenar el map por sus valores (Integer), e imprimir por pantalla los String **sin separación entre ellos.**

Debemos tener cuidado, solo queremos sacar una letra, veremos que en el map hay valores con mas caracteres, deberemos imprimir solo uno.

Ejemplo aaa -> a

Una vez correctamente realizado obtendremos el mensaje oculto.

Ejemplo orientativo:

```
mensaje,  oculto.
```