	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesora:	Rocío Alejandra Aldeco Pérez
Asignatura:	Programación Orientada a Objetos
Grupo:	6
No de Práctica(s):	3
Integrante(s):	Ugalde Velasco Armando
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	
Semestre:	2021-1
Fecha de entrega:	16 de octubre de 2020
Observaciones:	

CALIFICACIÓN: _____

Práctica de Estudio 3: Utilerías y clases de uso general
Programación Orientada a Objetos Grupo 6
Facultad de Ingeniería
Departamento de Computación

Objetivo de la práctica: Utilizar bibliotecas propias del lenguaje para realizar algunas tareas comunes y recurrentes.

Realiza las siguientes actividades después de leer y revisar en clase la **Práctica de Estudio 3: Utilerías y clases de uso general**

1. Usando la clase **ArrayList** crea una lista de la clase **Persona** (la creada en la práctica anterior) donde cada persona tenga **nombre completo**, **edad** y **RFC**.
2. Deberás poder realizar búsquedas en base a la edad, esto significa que podrás tener una función que permita buscar en tu lista el número de personas con una edad particular, mayores a una edad o menores a una edad.
3. Cuando estés seguro de que tu programa es correcto, súbelo a **Alphagrader**. Recuerda que si no pasas los tests no obtendrás los puntos de ejecución.
4. Finalmente explica tu proceso y concluye. Recuerda incluir tu caratula, subir este documento en PDF a Schoology.

Para cumplir con el objetivo planteado, se desarrolló el método estático **menoresA40**:

```
static public int menoresA40(List<Persona> personas)
{
    return personas.stream()
            .map(Persona::calcularEdad)
            .reduce( identity: 0, (acc, edad) → edad < 40 ? acc + 1 : acc);
}
```

Método estático menoresA40

Como su nombre lo indica, éste retorna la cantidad de personas con una edad menor a **40**, presentes en una lista con instancias de **Persona**. Se utiliza el método **stream** para crear una secuencia de los elementos, y así habilitar la utilización de las utilidades funcionales **map** y **reduce**. En el método **map** se provee el método **calcularEdad** como argumento, el cual se encarga de "mapear" cada persona a su respectiva edad. Como se puede deducir, el método **map** es una función de orden mayor. La operación anterior es intermedia, ya que se ejecuta de forma *"floja",* por lo tanto, retorna otro objeto que implementa la interfaz **Stream**.

Finalmente, se ejecuta la operación terminal **reduce** sobre el objeto antes mencionado, proporcionando como argumento una función que, a partir de un *"acumulador"* cuyo valor inicial es cero, determina la cantidad de personas con edades menores a 40, aumentando el valor del acumulador si se cumple la condición en cada elemento.

En la clase principal, se creó una lista de personas utilizando la clase **ArrayList**. Posteriormente, se leyeron las cadenas correspondientes utilizando la clase **Scanner**, y se crearon las instancias de la clase **Persona** a partir de los datos anteriores, para finalmente añadir éstas a la lista utilizando el método **add**.

Por último, se utilizó el método estático ya analizado para obtener el número de personas menores a 40 años en la lista, y se imprimieron los resultados correspondientes. El número de personas con edad mayor o igual a 40 se obtuvo restando el número de personas menores a dicha edad del tamaño total de la lista.

```
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    List<Persona> personas = new ArrayList<>();
    while (sc.hasNext())
    {
        String[] datos = sc.nextLine().split( regex: ",");
        Persona persona = new Persona(datos[0], datos[1], datos[2], datos[3]);
        personas.add(persona);
    }
    int menores = Persona.menoresA40(personas);
    System.out.println("MAYORES A 40: " + (personas.size() - menores));
    System.out.println("MENORES A 40: " + menores);
}
```

Método main

CONCLUSIONES

Java es un lenguaje robusto que cuenta con una gran cantidad de utilidades y **APIs** que cumplen una amplia variedad de objetivos. Las listas en Java son un tipo de dato abstracto que puede implementarse de distintas formas, siendo **ArrayList** y **LinkedList** las más comunes. A su vez, éstas cuentan con ciertos métodos que nos permiten realizar muchas operaciones sin necesidad de *"reinventar la rueda".* En la práctica, por ejemplo, se utilizó la implementación de la interfaz **stream** para hacer uso de las utilidades funcionales **map** y **reduce**, que facilitan en gran medida la realización de ciertas tareas y proveen un estilo de código más declarativo.

La diferencia más notable entre los arreglos y las listas es que los primeros no pueden cambiar de tamaño una vez que son inicializados. En cambio, las listas sí poseen esta capacidad: en el ejercicio realizado se insertaron elementos arbitrariamente en la lista de personas, sin necesidad de establecer un tamaño concreto. La forma en la que se logra lo anterior, como se mencionó, varía dependiendo de la implementación que se utilice, por ejemplo, **ArrayList** utiliza internamente un arreglo que cambia de tamaño, y **LinkedList** utiliza la estructura de lista ligada.