PROGRAMACIÓN FUNCIONAL EN JAVA

5to Semestre – L.T.I. Fray Bentos – 04.2023

Tabla de contenido

1.- Ejercicio de Biblioteca y Libros.

1. Obtener libros por año mayor a 2018.
2. Obtener libros de programación.
3. Obtener el primer libro por temática programación.
4. Obtener los libros por temática programación.

1.1.- Libro.java

```
package libros;

import java.util.Objects;

public class Libro {

	private int id;
	private String nombre;
	private String autor;
	private String tematica;
	private int año;

	public Libro() {
		super();
	}

	public Libro(int id, String nombre, String autor, String tematica,
int año) {
		super();
		this.id = id;
		this.nombre = nombre;
		this.autor = autor;
		this.tematica = tematica;
		this.año = año;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
```

```java
	public String getAutor() {
		return autor;
	}

	public void setAutor(String autor) {
		this.autor = autor;
	}

	public String getTematica() {
		return tematica;
	}

	public void setTematica(String tematica) {
		this.tematica = tematica;
	}

	public int getAño() {
		return año;
	}

	public void setAño(int año) {
		this.año = año;
	}

	@Override
	public int hashCode() {
		return Objects.hash(id);
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Libro other = (Libro) obj;
		return id == other.id;
	}



	@Override
	public String toString() {
		return "Id: " + id + ", nombre: " + nombre + ", autor: " +
autor + ", temática: " + tematica + ", año: " + año + ".";
	}

}
```

1.2.- Biblioteca.java

```
package libros;

import java.util.ArrayList;
import java.util.List;

public class Biblioteca {

	private static List<Libro> libros = new ArrayList<Libro>();

	public static void agregarLibros() {
		libros.add(new Libro(1, "Programación 1", "Javier Cano",
"Programación", 2018));
		libros.add(new Libro(2, "Programación 2", "Javier Cano",
"Programación", 2019));
		libros.add(new Libro(3, "Base de datos SQL", "Jhon Base", "Base de
datos", 2017));
		libros.add(new Libro(4, "Base de datos NoSQL", "Esteban Quito",
"Base de datos", 2018));
		libros.add(new Libro(5, "Testing Funcional", "Ana Lizando",
"Testing", 2016));
		libros.add(new Libro(6, "Testing de Performance", "Ana Lizando",
"Testing", 2018));
	}

	public static void ejercicio1() {
		// Obtener libros por ano mayor a 2018
		System.out.print("Ejercicio 1: Libros mayores a 2018:\n\t");
		libros.stream()
				.filter(l -> l.getAño() > 2018)
				.forEach(System.out::println);
	}

	public static void ejercicio2() {
		// Obtener libros con nombre igual = "Programación".
		System.out.println("Ejercicio 2: Libros con nombre que contenga
'Programación:'");
		libros.stream()
				.filter(l ->
l.getNombre().toLowerCase().contains(("Programación").toLowerCase()))
				.forEach(l -> System.out.println("\t" + l));
	}

	public static void ejercicio3() {
		// Obtener el primer libro por tematica = 'Programación'
		System.out.println("Ejercicio 3: Primer Libro con temática
'Programación:'");
		libros.stream()
				.filter(l -> l.getTematica().equals("Programación"))
				.findFirst().ifPresent(l -> System.out.println("\t" +
l));
	}

	public static void ejercicio4() {
		// Obtener todos los libros por tematica = 'Programación'
		System.out.println("Ejercicio 4: Libros con temática
'Programación:'");
```

```
            libros.stream()
                        .filter(l -> l.getTematica().equals("Programación"))
                        .forEach(l -> System.out.println("\t" + l));
      }

}
```

1.3.- Main.java

```
package main;

import libros.Biblioteca;

public class Main {

      public static void main(String[] args) {
            // Ejericio de Biblioteca - Libros
            System.out.println("-- Ejercicio Libros: --");
            Biblioteca.agregarLibros();
            Biblioteca.ejercicio1();
            Biblioteca.ejercicio2();
            Biblioteca.ejercicio3();
            Biblioteca.ejercicio4();
      }

}
```

2.- Ejercicio de Veterinaria y Mascotas.

1. Todas las mascotas con categoría perro se muestren en pantalla.
2. Ver que la mascota no sea nula e imprimir los datos de la primera que encuentre.
3. Mostrar todas las mascotas mayores a 8 años.

2.1.- Mascota.java

```
package mascotas;

import java.util.Objects;

public class Mascota {

	private int id;
	private String nombre;
	private String raza;
	private int edad;
	private String categoria;

	public Mascota() {
		super();
	}

	public Mascota(int id, String nombre, String raza, int edad, String
categoria) {
		super();
		this.id = id;
		this.nombre = nombre;
		this.raza = raza;
		this.edad = edad;
		this.categoria = categoria;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public String getRaza() {
		return raza;
	}

	public void setRaza(String raza) {
```

```
            this.raza = raza;
      }

      public int getEdad() {
            return edad;
      }

      public void setEdad(int edad) {
            this.edad = edad;
      }

      public String getCategoria() {
            return categoria;
      }

      public void setCategoria(String categoria) {
            this.categoria = categoria;
      }

      @Override
      public int hashCode() {
            return Objects.hash(id);
      }

      @Override
      public boolean equals(Object obj) {
            if (this == obj)
                  return true;
            if (obj == null)
                  return false;
            if (getClass() != obj.getClass())
                  return false;
            Mascota other = (Mascota) obj;
            return id == other.id;
      }

      @Override
      public String toString() {
            return "Id: " + id + ", nombre: " + nombre + ", raza: " + raza + ",
edad: " + edad + ", categoría: " + categoria + ".";
      }

}
```

2.2.- Veterinaria.java

```
package mascotas;

import java.util.ArrayList;
import java.util.List;

public class Veterinaria {

      private static List<Mascota> mascotas = new ArrayList<Mascota>();

      public static void crearMascotas() {
```

```
            mascotas.add(new Mascota(1, "Pepe", "Caniche", 2, "Perro"));
            mascotas.add(new Mascota(2, "pipi", "Persa", 5, "Gato"));
            mascotas.add(new Mascota(3, "fifi", "Pitbull", 5, "Perro"));
            mascotas.add(new Mascota(4, "Gabito", "Bokser", 2, "Perro"));
            mascotas.add(new Mascota(5, "Garfield", "Naranja", 2, "Gato"));
            mascotas.add(new Mascota(6, "Pepe", "Cotorra", 9, "Loro"));
      }

      public static void ejericio1() {
            // Todas las mascotas con categoría perro se muestren en pantalla.
            System.out.println("Ejercicio 1: Mascotas con categoría: 'Perro'");
            mascotas.stream().filter(m ->
m.getCategoria().equalsIgnoreCase("Perro"))
                        .forEach(m -> System.out.println("\t" + m));
      }

      public static void ejericio2() {
            // Ver que la mascota no sea nulo e imprimir los datos de la
primera que encuentre
            System.out.println("Ejercicio 2: Mascota que no sea nula y listar
la primera.");
            mascotas.stream().filter(mascota -> mascota != null)
                        .findFirst()
                        .ifPresent(m -> System.out.println("\t" + m));
      }

      public static void ejericio3() {
            // Mostrar todas las mascotas mayores a 8 años
            System.out.println("Ejercicio 3: Listado de mascotas con edad mayor
a 8 años.");
            mascotas.stream()
                        .filter(m -> m.getEdad() > 8)
                        .forEach(m -> System.out.println("\t" + m));
      }
}
```

2.3.- Main.java

```
package main;

import mascotas.Veterinaria;

public class Main {

      public static void main(String[] args) {
            // Ejercicio de Veterinaria - Mascotas
            System.out.println("\n\n\n-- Ejercicio Mascostas: --");
            Veterinaria.crearMascotas();
            Veterinaria.ejericio1();
            Veterinaria.ejericio2();
            Veterinaria.ejericio3();
      }

}
```

3.- Ejercicio de Ventas y Celulares.

1. Obtener los celulares con precio menor a 200.
2. Obtener los celulares que no sean de color "Gray".
3. Obtener los celulares que su marca tenga un largo de caracteres menor o igual a 6.

3.1.- Cellphone.java

```
package celulares;

import java.util.Objects;

public class Cellphone {

	private int id;
	private int num_of_cameras;
	private double price;
	private String brand;
	private String model;
	private String color;

	public Cellphone() {
		super();
	}

	public Cellphone(int id, int num_of_cameras, double price, String brand,
String model, String color) {
		super();
		this.id = id;
		this.num_of_cameras = num_of_cameras;
		this.price = price;
		this.brand = brand;
		this.model = model;
		this.color = color;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public int getNum_of_cameras() {
		return num_of_cameras;
	}

	public void setNum_of_cameras(int num_of_cameras) {
		this.num_of_cameras = num_of_cameras;
	}

	public double getPrice() {
		return price;
	}
```

```
	public void setPrice(double price) {
		this.price = price;
	}

	public String getBrand() {
		return brand;
	}

	public void setBrand(String brand) {
		this.brand = brand;
	}

	public String getModel() {
		return model;
	}

	public void setModel(String model) {
		this.model = model;
	}

	public String getColor() {
		return color;
	}

	public void setColor(String color) {
		this.color = color;
	}

	@Override
	public int hashCode() {
		return Objects.hash(id);
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Cellphone other = (Cellphone) obj;
		return id == other.id;
	}

	@Override
	public String toString() {
		return "Id: " + id + ", num_of_cameras: " + num_of_cameras + ",
price: " + price + ", brand: " + brand + ", model: " + model + ", color: " +
color + ".";
	}

}
```

3.2.- Ventas.java

```java
package celulares;

import java.util.ArrayList;
import java.util.List;

public class Ventas {

	private static List<Cellphone> celulares = new ArrayList<Cellphone>();

	public static void crearCelulares() {
		celulares.add(new Cellphone(1, 4, 267, "Samsung", "A21s",
"White"));
		celulares.add(new Cellphone(2, 2, 269, "Xiaomi", "Note 9",
"Gray"));
		celulares.add(new Cellphone(3, 1, 199, "Huawei", "Y6p", "Black"));
		celulares.add(new Cellphone(4, 3, 140, "Huawei", "Y8p", "Blue"));
	}

	public static void ejercicio1() {
		// Obtener los celulares con precio menor a 200
		System.out.println("Ejercicio 1: Celulares con precio menor a
200:");
		celulares.stream()
				.filter(c -> c.getPrice() < 200)
				.forEach(c -> System.out.println("\t" + c));
	}

	public static void ejercicio2() {
		// Obtener los celulares que no sean de color "Gray"
		System.out.println("Ejercicio 2: Celulares de color distinto a
\"Gray\":");
		celulares.stream()
				.filter(c -> !c.getColor().equals("Gray"))
				.forEach(c -> System.out.println("\t" + c));
	}

	public static void ejercicio3() {
		// Obtener los celulares que su marca tenga un largo de caracteres
menor o igual a 6
		System.out.println("Ejercicio 3: Celulares con marca de largo de
caracteres menor o igual a 6:");
		celulares.stream()
				.filter(c -> c.getBrand().length() <= 6)
				.forEach(c -> System.out.println("\t" + c));
	}
}
```

3.3.- Main.java

```java
package main;

import celulares.Ventas;

public class Main {

	public static void main(String[] args) {
		// Ejercicio de Ventas - Celulares
		System.out.println("\n\n\n-- Ejercicio Celulares: --");
		Ventas.crearCelulares();
		Ventas.ejercicio1();
		Ventas.ejercicio2();
		Ventas.ejercicio3();
	}
}
```

4.- Ejercicio de Personas y Usuarios.

1. Buscar mascotas con más de 2 años.
2. Obtener primer registro de la lista.
3. Actualizar registro donde el nombre y edad de la mascota está vacío, llenando los campos correspondientes.

4.1.- Usuarios.java

```
package usuarios;

import java.util.Objects;

public class Usuario {

	private String cedula;
	private String nombre;
	private String apellido;
	private String Departamento;
	private String NombreMascota;
	private int edadMascota;

	public Usuario(String cedula, String nombre, String apellido, String
departamento, String nombreMascota, int edadMascota) {
		super();
		this.cedula = cedula;
		this.nombre = nombre;
		this.apellido = apellido;
		Departamento = departamento;
		NombreMascota = nombreMascota;
		this.edadMascota = edadMascota;
	}

	public String getCedula() {
		return cedula;
	}

	public void setCedula(String cedula) {
		this.cedula = cedula;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public String getApellido() {
		return apellido;
	}

	public void setApellido(String apellido) {
```

```
            this.apellido = apellido;
      }

      public String getDepartamento() {
            return Departamento;
      }

      public void setDepartamento(String departamento) {
            Departamento = departamento;
      }

      public String getNombreMascota() {
            return NombreMascota;
      }

      public void setNombreMascota(String nombreMascota) {
            NombreMascota = nombreMascota;
      }

      public int getEdadMascota() {
            return edadMascota;
      }

      public void setEdadMascota(int edadMascota) {
            this.edadMascota = edadMascota;
      }

      @Override
      public int hashCode() {
            return Objects.hash(cedula);
      }

      @Override
      public boolean equals(Object obj) {
            if (this == obj)
                  return true;
            if (obj == null)
                  return false;
            if (getClass() != obj.getClass())
                  return false;
            Usuario other = (Usuario) obj;
            return Objects.equals(cedula, other.cedula);
      }

      @Override
      public String toString() {
            return "Cédula: " + cedula + ", nombre: " + nombre + ", apellido: "
+ apellido + ", Departamento: " + Departamento + ", NombreMascota: " +
NombreMascota + ", edadMascota: " + edadMascota + ".";
      }

}
```

4.2.- Personas.java

```
package usuarios;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class Personas {

	private static List<Usuario> personas = new ArrayList<Usuario>();

	public static void crearPersonas() {
		personas.add(new Usuario("45678901", "Juan", "Perez", "Mercedes",
"Firulais", 9));
		personas.add(new Usuario("38545601", "Maria", "Molina",
"Montevideo", "Tina", 4));
		personas.add(new Usuario("27504513", "Pedro", "Sanchez", "Rivera",
"Marito", 6));
		personas.add(new Usuario("43218547", "Sandra", "Rodriguez",
"Colonia", "Frutilla", 2));
		personas.add(new Usuario("27005801", "Matias", "Roldan",
"Montevideo", "Hueso", 3));
		personas.add(new Usuario("45778605", "Juan", "Sanchez", "Rivera",
"Sandrito", 4));
		personas.add(new Usuario("45775605", "Jose", "Ramirez",
"Montevideo", "", 0));
	}

	public static void ejercicio1() {
		// Buscar mascotas con mas de 2 años.
		System.out.println("Ejercicio 1: Usuarios con mascotas que tienen
edad mayor a 2 años.");
		personas.stream()
				.filter(u -> u.getEdadMascota() > 2)
				.forEach(u -> System.out.println("\t" + u));
	}

	public static void ejercicio2() {
		// Obtener primer registro de la lista.
		System.out.println("Ejercicio 2: Obtener primer registro de la
lista:\n\t" + personas.stream().findFirst().get());
	}

	public static void ejercicio3() {
		// Actualizar registro donde el nombre y edad de la mascota esta
vacio, llenando los campos correspondientes.
		System.out.println("Ejercicio 3: Actualizar registro con nombre y
edad de mascota sea vacío:");
		Optional<Usuario> user = personas.stream()
				.filter(usuario ->
usuario.getNombreMascota().isEmpty()).findFirst();
		user.get().setNombreMascota("Michis");
		user.get().setEdadMascota(3);
		System.out.println("\t" + user.get());
	}
}
```

4.3.- Main.java

```java
package main;

import usuarios.Personas;

public class Main {

	public static void main(String[] args) {
		System.out.println("\n\n\n-- Ejercicio Usuarios: --");
		Personas.crearPersonas();
		Personas.ejercicio1();
		Personas.ejercicio2();
		Personas.ejercicio3();
	}
}
```

5.- Ejercicio de Prestamos a Clientes.

Crear una aplicación que evalúe si el cliente califica para sacar un préstamo y asignar cupo (préstamo que puede tomar) en relación al préstamo que puede sacar el cliente.

Las relaciones de entrada de este contrato son: Sueldos >= 100000 Prestamos hasta 50000

Se pide:

1. Evaluar cupo, evaluamos si gana más de 100K.
2. A los que ganan más de 100K les pusimos true (son pasibles de sacar préstamo).
3. Evaluar cuanto le vamos a prestar: Si en préstamo tiene 0, le vamos a prestar.

5.1.- Cliente.java

```
package clientes;

import java.util.Objects;

public class Cliente {

	private int id;
	private String nombre;
	private String apellido;
	private double sueldoNominal;
	private double prestamo;
	private boolean tieneDisponible;
	private double cupo;
	private double prestamoSolicitado;

	public Cliente(int id, String nombre, String apellido, double
sueldoNominal, double prestamo, boolean tieneDisponible, double cupo, double
prestamoSolicitado) {
		super();
		this.id = id;
		this.nombre = nombre;
		this.apellido = apellido;
		this.sueldoNominal = sueldoNominal;
		this.prestamo = prestamo;
		this.tieneDisponible = tieneDisponible;
		this.cupo = cupo;
		this.prestamoSolicitado = prestamoSolicitado;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNombre() {
```

```
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public String getApellido() {
		return apellido;
	}

	public void setApellido(String apellido) {
		this.apellido = apellido;
	}

	public double getSueldoNominal() {
		return sueldoNominal;
	}

	public void setSueldoNominal(double sueldoNominal) {
		this.sueldoNominal = sueldoNominal;
	}

	public double getPrestamo() {
		return prestamo;
	}

	public void setPrestamo(double prestamo) {
		this.prestamo = prestamo;
	}

	public boolean isTieneDisponible() {
		return tieneDisponible;
	}

	public void setTieneDisponible(boolean tieneDisponible) {
		this.tieneDisponible = tieneDisponible;
	}

	public double getCupo() {
		return cupo;
	}

	public void setCupo(double cupo) {
		this.cupo = cupo;
	}

	public double getPrestamoSolicitado() {
		return prestamoSolicitado;
	}

	public void setPrestamoSolicitado(double prestamoSolicitado) {
		this.prestamoSolicitado = prestamoSolicitado;
	}

	@Override
	public int hashCode() {
		return Objects.hash(id);
```

```
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Cliente other = (Cliente) obj;
		return id == other.id;
	}

	@Override
	public String toString() {
		return "Id: " + id + ", nombre: " + nombre + ", apellido: " +
apellido + ", sueldoNominal: " + sueldoNominal + ", prestamo: " + prestamo + ",
tieneDisponible: " + tieneDisponible + ", cupo: " + cupo
				+ ", prestamoSolicitado: " + prestamoSolicitado + ".";
	}

}
```

5.2.- Calificación.java

```
package clientes;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Calificacion {

	private static List<Cliente> listaClientes = new
ArrayList<Cliente>();

	public static void crearClientes() {
		listaClientes.add(new Cliente(1, "Juan", "Perez", 100000, 0,
false, 0, 15000));
		listaClientes.add(new Cliente(2, "Jose", "Pretado", 90000, 0,
false, 0, 30000));
		listaClientes.add(new Cliente(3, "Pablo", "todo", 110000, 0,
false, 0, 50000));
		listaClientes.add(new Cliente(4, "Rosa", "Morales", 180000, 0,
false, 0, 45000));
	}

	public static void evaluarPrestamo() {
		// Evaluar cupo, evaluamos si gana mas de 100K
		ArrayList<Cliente> clientesCalificados =
listaClientes.stream()
				.filter(a -> a.getSueldoNominal() >= 100000)
				.collect(Collectors.toCollection(() -> new
ArrayList<>()));
```

```
            // A los que ganan mas de 100K les pusimos true (son pasibles
de sacar prestamo)
            clientesCalificados.forEach(c -> c.setTieneDisponible(true));
            // Evaluar cuanto le vamos a prestar. Si en prestamo tiene 0,
le vamos a prestar.
            ArrayList<Cliente> prestamoDisponible =
clientesCalificados.stream()
                        .filter(a -> a.getPrestamo() == 0)
                        .collect(Collectors.toCollection(() -> new
ArrayList<>()));
            prestamoDisponible.forEach(c ->
c.setPrestamo(c.getPrestamoSolicitado()));
            //Listamos los clientes con prestamo aprobado.
            prestamoDisponible.forEach(System.out::println);
      }
}
```

5.3.- Main.java

```
package main;

import clientes.Calificacion;

public class Main {

      public static void main(String[] args) {
            System.out.println("\n\n\n-- Ejercicio Prestamo: --
");
            Calificacion.crearClientes();
            Calificacion.evaluarPrestamo();
      }

}
```

6.- Ejercicio de Automotora y Autos

Supongamos que partimos de la definición de una clase Auto con las siguientes características:

```
public class Auto {
	private String marca;
	private Double precio;
	private int anio;
	// get y set...
}
```

Se pide:

Crear una clase de prueba (void main) que implemente los siguientes puntos usando streams y/o Optional:

Se tiene la siguiente lista:

```
Lista<Auto> autos = new ArrayList<Auto>;
autos.add(new Auto ("Toyota", 12000,2016));
autos.add(new Auto ("Fiat",8000,2016));
autos.add(new Auto ("Nissan", 9800, 2010));
autos.add(new Auto ("Audi", 20000, 2010));
```

1. Obtener e imprimir en consola las marcas de todos los autos cuyo precio sea menor a 10000 y el año sea mayor a 2015.
2. Obtener el auto cuya marca sea Toyota y su año sea 2018. Considerar que puede no haber ninguno que cumpla la condición.
3. Obtener e imprimir en consola la suma de todos los precios de los autos del año 2010.
4. Obtener e imprimir en consola la cantidad de autos total.

6.1.- Auto.java

```java
package automotora;

public class Auto {

	private String marca;
	private double precio;
	private int anio;

	public Auto(String marca, double precio, int anio) {
		super();
		this.marca = marca;
		this.precio = precio;
```

```
            this.anio = anio;
      }

      public String getMarca() {
            return marca;
      }

      public void setMarca(String marca) {
            this.marca = marca;
      }

      public double getPrecio() {
            return precio;
      }

      public void setPrecio(double precio) {
            this.precio = precio;
      }

      public int getAnio() {
            return anio;
      }

      public void setAnio(int anio) {
            this.anio = anio;
      }

      @Override
      public String toString() {
            return "Auto - marca: " + marca + ", precio: " + precio + ", anio:
" + anio + ".";
      }

}
```

6.2.- Automotora.java

```
package automotora;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class Automotora {

      private static List<Auto> autos = new ArrayList<Auto>();

      public static void crearAutos() {
            autos.add(new Auto("Toyota", 12000, 2016));
            autos.add(new Auto("Fiat", 8000, 2016));
            autos.add(new Auto("Nissan", 9800, 2010));
            autos.add(new Auto("Audi", 20000, 2010));
      }
```

```
	public static void ejercicio1() {
		// Obtener e imprimir en consola las marcas de todos los autos cuyo precio sea menor a 10000 y el año sea mayor a 2015.
		System.out.print("Ejercicio 1: Autos con precio menor a 10000 y año mayor a 2015: \n\t");
		autos.stream()
				.filter(a -> a.getPrecio() < 10000 && a.getAnio() > 2015)
				.forEach(System.out::println);
	}

	public static void ejercicio2() {
		// Obtener el auto cuya marca sea Toyota y su año sea 2018. Considerar que puede no haber ninguno que cumpla la condición.
		System.out.println("Ejercicio 2: Encontrar Auto Toyota y año 2018:");
		Optional<Auto> optionalAuto = autos.stream()
				.filter(a -> a.getMarca().equals("Toyota") && a.getAnio() == 2018)
				.findFirst();

		System.out.println(optionalAuto.isPresent()
				? "\t" + optionalAuto.get()
				: "\tNo se encontró un auto Toyota del año 2018.");
	}

	public static void ejercicio3() {
		// Obtener e imprimir en consola la suma de todos los precios de los autos del año 2010
		System.out.println("Ejercicio 3: Listar todos los precios de los autos año 2010:");
		autos.stream().filter(a -> a.getAnio() == 2010).forEach(a -> System.out.println("\tPrecio: " + a.getPrecio()));
	}

	public static void ejercicio4() {
		// Obtener e imprimir en consola la cantidad de autos total.
		System.out.println("Ejercicio 4: Cantidad total de autos.");
		System.out.println("\tLa cantidad total de autos es: " + autos.stream().count() + ".");
	}

}
```

6.3.- Main.java

```java
package main;

import automotora.Automotora;

public class Main {

	public static void main(String[] args) {
		// Ejercicio de Automotora y Auto
		System.out.println("\n\n\n-- Ejercicio Autos: --");
		Automotora.crearAutos();
		Automotora.ejercicio1();
		Automotora.ejercicio2();
		Automotora.ejercicio3();
		Automotora.ejercicio4();
	}
}
```

7.- Ejercicio de Granja y Cultivos

Se tiene una clase en Java:

```
public class Cultivo{
	private String nombre;
	private String categoria;
	private double resistencia;
	// constructor, get y set...
}
```

Definir en Java la clase Principal (void main) con la siguiente lista:

```
List< Cultivo > cultivos= new ArrayList< Cultivo >();
cultivos.add(new Cultivo ("Tomate", "Fruta",40));
cultivos.add(new Cultivo ("Frutilla","Fruta",15));
cultivos.add(new Cultivo ("Lechuga", "Verdura", 45));
cultivos.add(new Cultivo ("Papa", "Hortaliza", 61));
cultivos.add(new Cultivo ("Zanahoria", "Hortaliza", 75));
```

Se pide realizar lo siguiente utilizando stream y/u optional

1. Obtener e imprimir en consola los nombres de los cultivos con resistencia menor a 70 y categoría “Fruta”.
2. Obtener e imprimir el promedio de las resistencias de los cultivos de categoría “Hortaliza”
3. Obtener e imprimir en consola el nombre del primer cultivo cuya categoría sea “Tubérculo”. Si no existiera, imprimir el mensaje de “No existe cultivo en la categoría”.

7.1.- Cultivo.java

```java
package cultivos;

public class Cultivo {

	private String nombre;
	private String categoria;
	private double resistencia;

	public Cultivo(String nombre, String categoria, double resistencia) {
		super();
		this.nombre = nombre;
		this.categoria = categoria;
		this.resistencia = resistencia;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
```

```
	}

	public String getCategoria() {
		return categoria;
	}

	public void setCategoria(String categoria) {
		this.categoria = categoria;
	}

	public double getResistencia() {
		return resistencia;
	}

	public void setResistencia(double resistencia) {
		this.resistencia = resistencia;
	}

	@Override
	public String toString() {
		return "Cultivo - nombre: " + nombre + ", categoria: " + categoria
+ ", resistencia: " + resistencia + ".";
	}

}
```

7.2.- Granja.java

```
package cultivos;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.OptionalDouble;

public class Granja {

	private static List<Cultivo> cultivos = new ArrayList<Cultivo>();

	public static void crearCultivos() {
		cultivos.add(new Cultivo("Tomate", "Fruta", 40));
		cultivos.add(new Cultivo("Frutilla", "Fruta", 15));
		cultivos.add(new Cultivo("Lechuga", "Verdura", 45));
		cultivos.add(new Cultivo("Papa", "Hortaliza", 61));
		cultivos.add(new Cultivo("Zanahoria", "Hortaliza", 75));
	}

	public static void ejercicio1() {
		// Obtener e imprimir en consola los nombres de los cultivos
con resistencia menor a 70 y categoría “Fruta”.
		System.out.println("Ejercicio 1: Nombre de cultivos con
resistencia menor a 70 y categoría \"Fruta\":");
		cultivos.stream()
```

```
                    .filter(c -> c.getCategoria().equals("Fruta") &&
c.getResistencia() < 70)
                    .forEach(c -> System.out.println("\t - " +
c.getNombre() + "."));
     }

     public static void ejercicio2() {
          // Obtener e imprimir el promedio de las resistencias de los
cultivos de categoría “Hortaliza”.
          System.out.println("Ejercicio 2: Promedio de las resistencias
de los cultivos de categoría “Hortaliza”.");

          OptionalDouble promedio = cultivos.stream()
                    .filter(c -> c.getCategoria().equals("Hortaliza"))
                    .mapToDouble(Cultivo::getResistencia)
                    .average();

          System.out.println(
                    promedio.isPresent()
                              ? "\tEl promedio de resistencia de los
cultivos de categoría 'Hortaliza' es de: " + promedio.getAsDouble() + "."
                              : "\tNo se encontraron cultivos de
categoría 'Hortaliza'.");

     }

     public static void ejercicio3() {
          // Obtener e imprimir en consola el nombre del primer cultivo
cuya categoría sea
          // “Tubérculo”. Si no existiera, imprimir el mensaje de “No
existe cultivo en la categoría”.
          System.out.println("Ejercicio 3: Primer cultivo de categoría
“Tubérculo”.");
          Optional<Cultivo> cultivoTuberculo = cultivos.stream()
                    .filter(c -> c.getCategoria().equals("Tubérculo"))
                    .findFirst();

          System.out.println(
                    cultivoTuberculo.isPresent()
                              ? "\tEl primer cultivo de categoría
“Tubérculo” es: " + cultivoTuberculo.get().getNombre() + "."
                              : "\tNo existe cultivo en la categoría
“Tubérculo”.");
     }

}
```

7.3.- Main.java

```java
package main;

import cultivos.Granja;

public class Main {

	public static void main(String[] args) {
		System.out.println("\n\n\n-- Ejercicio Cultivos: --");
		// Ejercicio de Granja y Cultivo
		Granja.crearCultivos();
		Granja.ejercicio1();
		Granja.ejercicio2();
		Granja.ejercicio3();
	}

}
```

8.- Ejercicio de IMC – Indice de Masa Corporal:

Se desea implementar un programa que calcule el IMC (Indice de Masa Corporal) en Java haciendo uso de una interfaz funcional.

Se pide implementar un método de la interfaz funcional, que dado dos parámetros de tipo double, realice los siguientes cálculos.

1. El resultado de aplicar el método para calcular el IMC de una persona de 44 kg de peso y 1.50 m de altura.
2. Ahora se quiere calcular el peso de una persona con IMC = 19.56 y peso de 44 kg

Entregar la clase Principal y un documento con breve explicación del punto 1.

Nota: la fórmula del IMC es peso sobre altura al cuadrado => IMC = peso / altura2

Ejemplo: para una persona que pesa 68 kg y mide 1.65 se calcula: $68 \div (1.65)^2 = 24.98$.

8.1.- CalculadoraIMC.java

```java
package imc;

public interface CalculadoraIMC {

    double calcularValor(double valorX, double valorY);

}
```

8.2.- IndiceMasaCorporal.java

```
package imc;

public class IndiceMasaCorporal {

	public static void ejercicio1() {
		// Calculando el IMC de una persona que pesa 44 kg y mide 1.50 m
		CalculadoraIMC calc = (peso, altura) -> peso / (altura * altura);
		double valorImc = calc.calcularValor(44, 1.5);
		System.out.println("\tI.M.C.: " + valorImc);
	}

	public static void ejercicio2() {
		// Calculando el peso de una persona con un IMC de 19.56 y un peso de 44 kg
		CalculadoraIMC calc = (imc, altura) -> imc * (altura * altura);
		double peso = calc.calcularValor(19.555555555555557, 1.5);
		System.out.println("\tPeso: " + peso);
	}

}
```

8.3.- Main.java

```
package main;

import imc.IndiceMasaCorporal;

public class Main {

	public static void main(String[] args) {
		// Ejercicio de I.M.C. - Indice Masa corporal.
		System.out.println("\n\n\n-- Ejercicio I.M.C.: --");
		IndiceMasaCorporal.ejercicio1();
		IndiceMasaCorporal.ejercicio2();
	}

}
```

9.- Ejercicio de Empresa y Empleados

Una empresa que cuenta con muchos empleados desea implementar un sistema de liquidación de sueldos. Se investiga el uso de la tecnología java streams para la solución por lo que se deben hacer pruebas con algunas de las principales funcionalidades.

Dada la siguiente clase Empleado:

```
public class Empleado {
	private int numero;
	private String nombre;
	private double valorJornal;
	private int fechaIngreso;
	private String nombreOficina;
	//constructor
	//getters y setters
}
```

Se necesitan implementar las siguientes funcionalidades usando streams a modo de prueba en el método main de la clase Principal que cuenta con los siguientes datos de prueba cargados:

```
List<Empleado> empleados = new ArrayList();
empleados.add(new Empleado(1,"Juan", 2000, 2000,"RRHH"));
empleados.add(new Empleado(1,"Maria", 5000,2005,"VENTAS"));
empleados.add(new Empleado(1,"Ana",1000, 2011,"RRHH"));
empleados.add(new Empleado(1,"Julia",2000, 2011,"RRHH"));
```

Se pide completar el método main agregando los siguientes requerimientos:

1. Obtener los empleados de la oficina “RRHH” que su valor de jornal sea mayor 1000. El resultado se debe guardar en una variable de tipo lista. Por último, se debe recorrer la lista resultante y mostrar su contenido.
2. Calcular el gasto en sueldos para el mes, para calcular el monto total se debe tomar el valor de cada jornal y agregarle un 30 % de impuestos. Los jornales de un trabajador son 30 en el mes. El valor final se debe guardar en una variable de tipo double.
3. Buscar si hay algún empleado con antigüedad mayor a 15 años. Se debe tomar precaución de que no existan casos de éxito para esta búsqueda. Imprimir por consola si se encontró o no algún caso que cumpla la condición.

Ejecutar la clase para probar las sentencias solicitadas y entregar una imagen que muestre los resultados.

9.1.- Empleado.java

```
package empresa;

public class Empleado {

	private int numero;
	private String nombre;
	private double valorJornal;
	private int fechaIngreso;
	private String nombreOficina;

	public Empleado(int numero, String nombre, double valorJornal, int fechaIngreso,
String nombreOficina) {
		super();
		this.numero = numero;
		this.nombre = nombre;
		this.valorJornal = valorJornal;
		this.fechaIngreso = fechaIngreso;
		this.nombreOficina = nombreOficina;
	}

	public int getNumero() {
		return numero;
	}

	public void setNumero(int numero) {
		this.numero = numero;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public double getValorJornal() {
		return valorJornal;
	}

	public void setValorJornal(double valorJornal) {
		this.valorJornal = valorJornal;
	}

	public int getFechaIngreso() {
		return fechaIngreso;
	}

	public void setFechaIngreso(int fechaIngreso) {
		this.fechaIngreso = fechaIngreso;
	}

	public String getNombreOficina() {
		return nombreOficina;
	}

	public void setNombreOficina(String nombreOficina) {
		this.nombreOficina = nombreOficina;
	}

	@Override
	public String toString() {
```

```
            return "Empleado - numero: " + numero + ", nombre: " + nombre + ",
valorJornal: " + valorJornal + ", fechaIngreso: " + fechaIngreso + ", nombreOficina: "
+ nombreOficina + ".";
      }

}
```

9.2.- Empresa.java

```
package empresa;

import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Empresa {

      public static List<Empleado> empleados = new ArrayList<Empleado>();

      public static void crearEmpleados() {
            empleados.add(new Empleado(1, "Juan", 2000, 2000, "RRHH"));
            empleados.add(new Empleado(1, "María", 5000, 2005, "VENTAS"));
            empleados.add(new Empleado(1, "Ana", 1000, 2011, "RRHH"));
            empleados.add(new Empleado(1, "Julia", 2000, 2011, "RRHH"));
      }

      public static void ejercicio1() {
            System.out.println("Ejercicio 1: Empleados \"RRHH\" y Jornal > 1000. Que
además "
                        + "se guarda en una nueva lista y son listados.");
            // Se crea e instancia la variable temporal.
            List<Empleado> resultado = new ArrayList<Empleado>();
            resultado = empleados.stream()
                        // Filtro la busqueda.
                        .filter(e -> e.getNombreOficina().equals("RRHH") &&
e.getValorJornal() > 1000)
                        // Asigno los empleados encontrados a la nueva lista.
                        .collect(Collectors.toList());
            // Se listan los empleados encontrados.
            resultado.forEach(e -> System.out.println("\t" + e));
      }

      public static void ejercicio2() {
            System.out.println("Ejercicio 2: Cálculo de gasto en sueldo + 30% de
impuestos.");
            double gasto = empleados.stream()
                        // Calcula el costo mensual de cada empleado con impuestos
                        .mapToDouble(e -> e.getValorJornal() * 30 * 1.3)
                        // Suma los costos mensuales de todos los empleados
                        .sum();
            NumberFormat formatoMoneda = new DecimalFormat("$ #,##0.00");
            System.out.println("\tGasto total: " + formatoMoneda.format(gasto));
      }

      public static void ejercicio3() {
            System.out.println("Ejercicio 3: Listar empleados con antigüedad mayor a
15 años.");
            List<Empleado> empleadosAntiguos = empleados.stream()
```

```
                        // Filtra los empleados con antigüedad mayor a 15 años
                        .filter(e -> (2023 - e.getFechaIngreso()) > 15)
                        // Convierte el flujo de empleados en una lista
                        .collect(Collectors.toList());
            // Listamos los empleados.
            if (!empleadosAntiguos.isEmpty()) {
                  System.out.println("\tSe encontraron los siguientes empleados con
antigüedad mayor a 15 años:");
                  empleadosAntiguos.forEach(e -> System.out.println("\t" + e));
            } else {
                  System.out.println("No se encontró ningún empleado con antigüedad
mayor a 15 años.");
            }
      }

}
```

9.3.- Main.java

```
package main;

import empresa.Empresa;

public class Main {

      public static void main(String[] args) {
            // Ejercicio de Empresa y Empleado.
            System.out.println("\n\n\n-- Ejercicio Empresa y Empleado: --");
            Empresa.crearEmpleados();
            Empresa.ejercicio1();
            Empresa.ejercicio2();
            Empresa.ejercicio3();
      }

}
```

10.- Ejercicio de Instituto y Alumnos.

Dada la definición de Alumno adjunta y la carga de una lista de alumnos, escribir utilizando java 8 los siguientes pedidos:

1. Obtener todos los alumnos de la lista.
2. Obtener la cantidad de elementos de la lista de alumnos.
3. Obtener los alumnos con nota mayor a 9.
4. Obtener los alumnos del curso PHP.
5. Obtener el primer alumno de la lista.
6. Obtener los alumnos cuyos nombres superen los 10 caracteres.
7. Obtener los alumnos cuyo nombre comienza con la letra A.

10.1.- Alumno.java

```
package instituto;

public class Alumno {
	private int id;
	private String cedula;
	private String nombres;
	private String apellidos;
	private String nombreCurso;
	private double nota;
	private int edad;

	public Alumno() {

	}

	public Alumno(int id, String cedula, String nombres, String apellidos,
String nombreCurso, double nota, int edad) {
		this.id = id;
		this.cedula = cedula;
		this.nombres = nombres;
		this.apellidos = apellidos;
		this.nombreCurso = nombreCurso;
		this.nota = nota;
		this.edad = edad;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getCedula() {
		return cedula;
```

```
	}

	public void setCedula(String cedula) {
		this.cedula = cedula;
	}

	public String getNombres() {
		return nombres;
	}

	public void setNombres(String nombres) {
		this.nombres = nombres;
	}

	public String getApellidos() {
		return apellidos;
	}

	public void setApellidos(String apellidos) {
		this.apellidos = apellidos;
	}

	public String getNombreCurso() {
		return nombreCurso;
	}

	public void setNombreCurso(String nombreCurso) {
		this.nombreCurso = nombreCurso;
	}

	public double getNota() {
		return nota;
	}

	public void setNota(double nota) {
		this.nota = nota;
	}

	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	@Override
	public String toString() {
		return "Id: " + id + ", cedula: " + cedula + ", nombres: " +
nombres + ", apellidos: " + apellidos + ", nombreCurso: " + nombreCurso + ",
nota: " + nota + ", edad: " + edad + ".";
	}

}
```

10.2.- Instituto.java

```
package instituto;

import java.util.ArrayList;
import java.util.List;

public class Instituto {

	private static List<Alumno> listaAlumnos = new ArrayList<>();

	public static void cargarAlumnos() {
		listaAlumnos.add(new Alumno(1, "1717213183", "Javier Ignacio",
"Molina Cano", "Java 8", 7, 28));
		listaAlumnos.add(new Alumno(2, "1717456218", "Lillian Eugenia",
"Gómez Álvarez", "Java 8", 10, 33));
		listaAlumnos.add(new Alumno(3, "1717328901", "Sixto Naranjoe",
"Marín", "Java 8", 8.6, 15));
		listaAlumnos.add(new Alumno(4, "1717567128", "Gerardo Emilio",
"Duque Gutiérrez", "Java 8", 10, 13));
		listaAlumnos.add(new Alumno(5, "1717902145", "Jhony Alberto",
"Sáenz Hurtado", "Java 8", 9.5, 15));
		listaAlumnos.add(new Alumno(6, "1717678456", "Germán Antonio",
"Lotero Upegui", "Java 8", 8, 34));
		listaAlumnos.add(new Alumno(7, "1102156732", "Oscar Darío",
"Murillo González", "Java 8", 8, 32));
		listaAlumnos.add(new Alumno(8, "1103421907", "Augusto Osorno",
"Palacio Martínez", "PHP", 9.5, 17));
		listaAlumnos.add(new Alumno(9, "1717297015", "César Oswaldo",
"Alzate Agudelo", "Java 8", 8, 26));
		listaAlumnos.add(new Alumno(10, "1717912056", "Gloria Amparo",
"González Castaño", "PHP", 10, 28));
		listaAlumnos.add(new Alumno(11, "1717912058", "Jorge León", "Ruiz
Ruiz", "Python", 8, 22));
		listaAlumnos.add(new Alumno(12, "1717912985", "John Jairo",
"Duque García", "Java Script", 9.4, 32));
		listaAlumnos.add(new Alumno(13, "1717913851", "Julio Cesar",
"González Castaño", "C Sharp", 10, 22));
		listaAlumnos.add(new Alumno(14, "1717986531", "Gloria Amparo",
"Rodas Monsalve", "Ruby", 7, 18));
		listaAlumnos.add(new Alumno(15, "1717975232", "Gabriel Jaime",
"Jiménez Gómez", "Java Script", 10, 18));
	}

	public static void ejercicio1() {
		// Obtener todos los alumnos de la lista
		System.out.print("Ejercicio 1: Lista todos los alumnos:\n");
		listaAlumnos.forEach(a -> System.out.println("\t" + a));
	}

	public static void ejercicio2() {
		// Obtener la cantidad de elementos de la lista de alumnos
		System.out.println("Ejercicio 2: Cantidad de alumnos en la
lista:\n\t El total es:" + listaAlumnos.stream().count() + ".");
```

```
	}

	public static void ejercicio3() {
		// Obtener los alumnos con nota mayor a 9
		System.out.println("Ejercicio 3: Alumnos con nota mayor a 9.");
		listaAlumnos.stream()
				.filter(alumno -> alumno.getNota() > 9)
				.forEach(alumno -> System.out.println("\t" +
alumno));
	}

	public static void ejercicio4() {
		// Obtener los alumnos del curso PHP
		System.out.println("Ejercicio 4: Alumnos del curso PHP.");
		listaAlumnos.stream()
				.filter(alumno ->
alumno.getNombreCurso().equals("PHP"))
				.forEach(alumno -> System.out.println("\t" +
alumno));
	}

	public static void ejercicio5() {
		// Obtener el primer alumno de la lista
		System.out.println("Ejercicio 5: Obtener el primer alumno de la
lista.\n\t "
				+ listaAlumnos.stream().findFirst().get());
	}

	public static void ejercicio6() {
		// Obtener los alumnos cuyos nombres superen los 10 caracteres
		System.out.println("Ejercicio 6: Alumnos cuyos nombres superen
los 10 caracteres.");
		listaAlumnos.stream()
				.filter(alumno -> alumno.getNombres().length() > 10)
				.forEach(alumno -> System.out.println("\t" +
alumno));
	}

	public static void ejercicio7() {
		// Obtener los alumnos cuyo nombre comienza con la letra A
		System.out.println("Ejercicio 7: Alumnos cuyo nombre comienza con
la letra A.");
		listaAlumnos.stream()
				.filter(alumno ->
alumno.getNombres().startsWith("A"))
				.forEach(alumno -> System.out.println("\t" +
alumno.toString()));
	}

}
```

10.3.- Main.java

```java
package main;

import instituto.Instituto;

public class Main {

	public static void main(String[] args) {
		System.out.println("\n\n\n-- Ejercicio Instituto y Alumno: --");
		Instituto.cargarAlumnos();
		Instituto.ejercicio1();
		Instituto.ejercicio2();
		Instituto.ejercicio3();
		Instituto.ejercicio4();
		Instituto.ejercicio5();
		Instituto.ejercicio6();
		Instituto.ejercicio7();
	}

}
```

11.- Ejercicio Supermercado y Articulo

Dada la definición de Alumno adjunta y la carga de una lista de alumnos, escribir utilizando java 8 los siguientes pedidos:

1. Obtener la cantidad de artículos de la lista
2. Obtener la cantidad de artículos de limpieza de la lista
3. Sumar los precios de todos los artículos
4. Sumar los precios de los artículos cuyo precio sea menor a 100
5. Verificar si algún precio es mayor o igual a 500
6. Imprimir el primer artículo de la categoría limpieza
7. Imprimir el nombre del primer artículo de la categoría otros

11.1.- Articulo.java

```
package supermercado;

public class Articulo {

	private long id;
	private Double precio;
	private String nombre;
	private Categoria categoria;

	public Articulo(long id, Double precio, String nombre, Categoria
categoria) {
		super();
		this.id = id;
		this.precio = precio;
		this.nombre = nombre;
		this.categoria = categoria;
	}

	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public Double getPrecio() {
		return precio;
	}

	public void setPrecio(Double precio) {
		this.precio = precio;
	}

	public String getNombre() {
		return nombre;
```

```
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public Categoria getCategoria() {
		return categoria;
	}

	public void setCategoria(Categoria categoria) {
		this.categoria = categoria;
	}

	@Override
	public String toString() {
		return "Articulo - id: " + id + ", precio: " + precio + ", nombre:
" + nombre + ", categoria: " + categoria + ".";
	}

	enum Categoria {
		LACTEOS, SALUD, BELLEZA, LIMPIEZA, COMESTIBLE, BEBIDA, OTROS
	}
}
```

11.2.- Supermercado.java

```
package supermercado;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import supermercado.Articulo.Categoria;

public class Supermercado {

	private static List<Articulo> articulos = new
ArrayList<Articulo>();

	public static void crearArticulos() {
		articulos.add(new Articulo(1, (double) 100, "Shampu",
Categoria.LIMPIEZA));
		articulos.add(new Articulo(2, (double) 40, "Fideos",
Categoria.COMESTIBLE));
		articulos.add(new Articulo(3, (double) 30, "Leche",
Categoria.LACTEOS));
		articulos.add(new Articulo(4, (double) 68, "Perfume",
Categoria.BELLEZA));
		articulos.add(new Articulo(5, (double) 500, "Vitaminas",
Categoria.SALUD));
		articulos.add(new Articulo(6, (double) 60, "Yoguth",
Categoria.LACTEOS));
```

```
          articulos.add(new Articulo(7, (double) 120, "Limpia pisos",
Categoria.LIMPIEZA));
          articulos.add(new Articulo(8, (double) 80, "Vino",
Categoria.BEBIDA));
          articulos.add(new Articulo(9, (double) 20, "Galletas",
Categoria.COMESTIBLE));
          articulos.add(new Articulo(10, (double) 30, "Harina",
Categoria.COMESTIBLE));
          // articulos.add(new Articulo(11, (double) 30, "Vinagre",
Categoria.OTROS));
     }

     public static void ejercicio1() {
          // Obtener la cantidad de artículos de la lista
          System.out.println("Ejercicio 1: Cantidad de artículos:\n\t El
total es: " + articulos.stream().count() + ".");
     }

     public static void ejercicio2() {
          // Obtener la cantidad de artículos de limpieza de la lista
          System.out.println("Ejercicio 2: Cantidad de artículos de
limpieza:\n\t El total es: "
                    + articulos.stream()
                              .filter(a ->
a.getCategoria().equals(Articulo.Categoria.LIMPIEZA))
                              .count()
                    + ".");
     }

     public static void ejercicio3() {
          // Sumar los precios de todos los artículos
          System.out.println("Ejercicio 3: La suma total del precio de
los artículos es:\n\t Total: "
                    + articulos.stream()
                              .mapToDouble(Articulo::getPrecio)
                              .reduce(0, Double::sum));
     }

     public static void ejercicio4() {
          // Sumar los precios de los artículos cuyo precio sea menor a
100
          System.out.println("Ejercicio 4: La suma total de los
artículos con precio menor a 100 es:\n\t Total: "
                    + articulos.stream()
                              .filter(a -> a.getPrecio() < 100)
                              .mapToDouble(Articulo::getPrecio)
                              .sum());
     }

     public static void ejercicio5() {
          // Verificar si algún precio es mayor o igual a 500
          System.out.println("Ejercicio 5: Verificar si algún precio es
mayor o igual a 500.");
```

```
          boolean algunPrecioMayorIgualA500 =
articulos.stream().anyMatch(a -> a.getPrecio() >= 500);

          if (algunPrecioMayorIgualA500) {
               System.out.println("\tExiste artículo con precio igual a
500.");
               articulos.stream().filter(a -> a.getPrecio() >=
500).forEach(a -> System.out.println("\t\t" + a));
          } else {
               System.out.println("\tNo existe artículo con precio
igual a 500.");
          }
     }

     public static void ejercicio6() {
          // Imprimir el primer artículo de la categoría limpieza
          System.out.println("Ejercicio 6: Imprimir el primer artículo
de la categoría \"LIMPIEZA\".");
          Optional<Articulo> primerArticuloLimpieza = articulos.stream()
                    .filter(a -> a.getCategoria() ==
Categoria.LIMPIEZA)
                    .findFirst();
          if (primerArticuloLimpieza.isPresent()) {
               System.out.println("\t" + primerArticuloLimpieza.get());
          } else {
               System.out.println("\tNo existe artículo con categoría
\"Limpieza\"");
          }
     }

     public static void ejercicio7() {
          // Imprimir el nombre del primer artículo de la categoría
otros
          System.out.println("Ejercicio 7: Imprimir el nombre del primer
artículo de la categoría \"OTROS\".");
          Optional<String> nombrePrimerArticuloOtros =
articulos.stream()
                    .filter(a -> a.getCategoria() == Categoria.OTROS)
                    .map(Articulo::getNombre)
                    .findFirst();
          if (nombrePrimerArticuloOtros.isPresent()) {
               System.out.println("\t" +
nombrePrimerArticuloOtros.get());
          } else {
               System.out.println("\tNo existe artículo con categoría
\"OTROS\"");
          }
     }

}
```

11.3.- Main.java

```java
package main;

import supermercado.Supermercado;

public class Main {

	public static void main(String[] args) {
		System.out.println("\n\n\n- Ejercicio Supermercado y Articulo: -");
		Supermercado.crearArticulos();
		Supermercado.ejercicio1();
		Supermercado.ejercicio2();
		Supermercado.ejercicio3();
		Supermercado.ejercicio4();
		Supermercado.ejercicio5();
		Supermercado.ejercicio6();
		Supermercado.ejercicio7();
	}
}
```

12.- Ejercicio Personas

Supongamos que partimos de la definición de una clase Persona y se pide: Crear una clase de nombre Principal (void main) que implemente los siguientes puntos usando streams y/o Optional.

1. Obtener e imprimir en consola los apellidos de todas las personas cuya edad sea menor a 50 y mayor a 30.
2. Obtener e imprimir el apellido de la primera persona cuyo nombre sea Juan y su edad sea 32. Considerar que puede no haber ninguna que cumpla la condición.
3. Obtener e imprimir en consola el promedio de todas las edades.

Realizar los puntos anteriores en una clase con nombre Principal y dentro del método main entregar la clase Principal y un documento con breve explicación del punto 3.

12.1.- Persona.java

```java
package personas;

public class Persona {

	private String nombre;
	private String apellido;
	private int edad;

	public Persona(String nombre, String apellido, int edad) {
		super();
		this.nombre = nombre;
		this.apellido = apellido;
		this.edad = edad;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public String getApellido() {
		return apellido;
	}

	public void setApellido(String apellido) {
		this.apellido = apellido;
	}
```

```
	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	@Override
	public String toString() {
		return "Persona - nombre: " + nombre + ", apellido: " +
apellido + ", edad: " + edad + ".";
	}

}
```

12.2.- PrincipalPersona.java

```
package personas;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.OptionalDouble;

public class PrincipalPersona {

	private static List<Persona> personas = new ArrayList<Persona>();

	public static void crearPersonas() {
		personas.add(new Persona("Juan", "Perez", 40));
		personas.add(new Persona("Ana", "Martinez", 32));
		personas.add(new Persona("Laura", "Lopez", 45));
		personas.add(new Persona("Marcos", "Olmos", 59));
	}

	public static void ejercicio1() {
		// Obtener e imprimir en consola los apellidos de
		// todas las personas cuya edad sea menor a 50 y mayor a 30.
		System.out.println("Ejercicio 1: Imprimir apellidos de"
				+ "personas menor a 50 y mayor a 30.");
		personas.stream().filter(
				a -> a.getEdad() > 30 && a.getEdad() < 50)
				.forEach(a -> System.out.println("\t- "
						+ a.getApellido() + "."));
	}

	public static void ejercicio2() {
		// Obtener e imprimir el apellido de la primera persona cuyo
		// nombre sea Juan y su edad sea 32. Considerar que puede no
		// haber ninguna que cumpla la condición.
		System.out.println("Ejercicio 2: Imprimir el apellido de la "
				+ "1ra persona que se llame Juan y su edad sea 32.");
		Optional<String> apellido = personas.stream()
```

```
                        .filter(p -> p.getNombre().equals("Juan")
                                    && p.getEdad() == 32)
                        .findFirst()
                        .map(Persona::getApellido);

            if (apellido.isPresent()) {
                  System.out.println("\tEl apellido es: " + apellido.get());
            } else {
                  System.out.println("\tNo hay resultado.");
            }
      }

      public static void ejercicio3() {
            // Obtener e imprimir en consola el promedio de todas las edades.
            System.out.println("Ejercicio 3: Imprimir el promedio de todas las
edades.");
            OptionalDouble promedioEdades = personas.stream()
                        .mapToInt(Persona::getEdad)
                        .average();
            promedioEdades.ifPresent(
                        promedio -> System.out.println(
                                    "\tEl promedio de edades es: " + promedio));
      }
}
```

12.3.- Main.java

```
package main;

import personas.PrincipalPersona;

public class Main {

      public static void main(String[] args) {
            System.out.println("\n\n\n-- Ejercicio Personas: --");
            PrincipalPersona.crearPersonas();
            PrincipalPersona.ejercicio1();
            PrincipalPersona.ejercicio2();
            PrincipalPersona.ejercicio3();
      }

}
```