



**Argentina
programa**


Programación Orientada a Objetos con JAVA

Guía V –Parte 2

Ejercicios

EJERCICIOS DE APRENDIZAJE

En este módulo vamos a continuar modelando los objetos con el lenguaje de programación Java, pero ahora vamos a utilizar las colecciones para poder manejarlas de manera más sencilla y ordenada.

	VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.
---	--

1) Un directorio telefónico posee una lista de Cientes de los que interesa conocer: dni, nombre, apellido, ciudad y dirección. El directorio está compuesto por el número de teléfono y los datos del Cliente (Directorio: <teléfono, Cliente> donde el teléfono no es un atributo del cliente. Cuando agendamos un cliente al directorio telefónico lo agendamos con su número de teléfono, que es único)

El directorio telefónico posee además las siguientes funcionalidades:

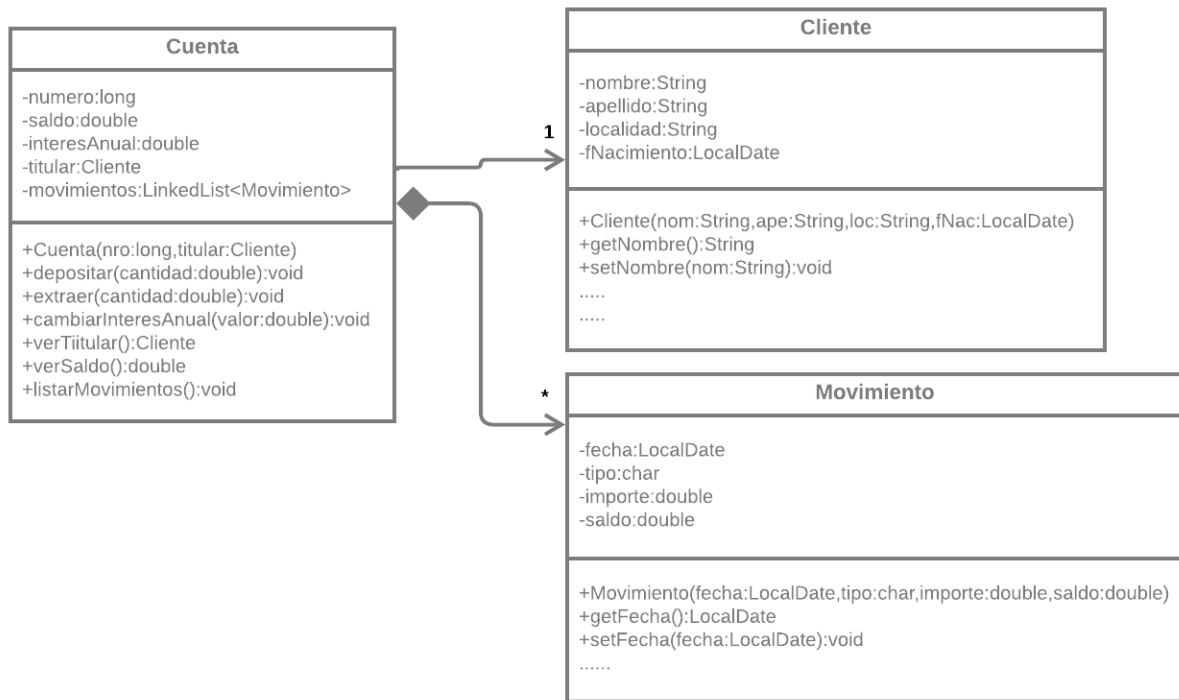
- **agregarCliente(nroTel, Cliente):void** → que permite registrar un nuevo cliente con su respectivo nro de teléfono. Siendo el nro del teléfono la clave del mismo.
- **buscarCliente(nroTel):Cliente** → que en base al nro de teléfono retorna el Cliente asociado al mismo.
- **buscarTeléfono(apellido):List** → que en base a un apellido nos devuelve una lista con los nros. de teléfono asociados a dicho apellido.
- **buscarClientes(ciudad):List** → que en base a una ciudad nos devuelve una lista con los Clientes asociados a dicha ciudad.
- **borrarCliente(telefono):void** que en base a un nro de teléfono elimina el cliente del directorio.

Luego desde el método main de una clase Test se pide:

- Crear un directorio.
- Cargar 5 clientes al directorio a través de su método **agregarCliente**, con sus respectivos nros de teléfono.
- Probar el resto los métodos que posee directorio.

Importante: Armar el modelo UML representado las clases necesarias. Implementar en java.

2) Dado el siguiente modelo implementar según la consigna.



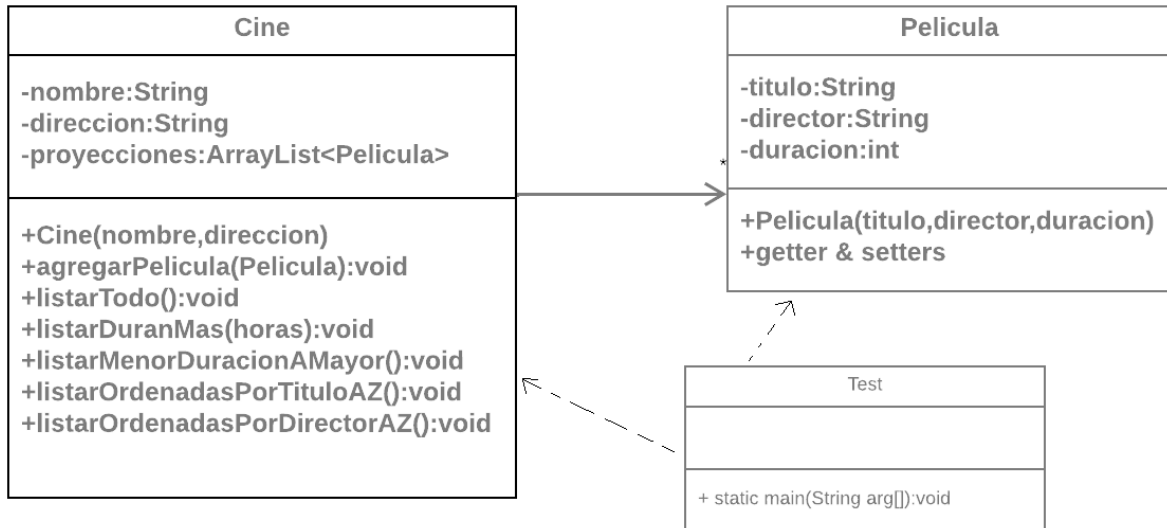
Descripción de los métodos de la clase Cuenta:

- **depositar(cantidad)** → Recibe la cantidad de dinero que incrementará el saldo de la cuenta y registra el movimiento “D”(Depósito)
- **extraer(cantidad)** → Recibe la cantidad de dinero a extraer que disminuirá el saldo de la cuenta y registra el movimiento “E” (Extracción). Ojo!!! Solo extraerá dinero si hay saldo suficiente.
- **cambiarInteresAnual(valor)** → Establecerá el nuevo interés anual de la cuenta.
- **verTitular(): Cliente** → Retornará el Cliente titular de la cuenta
- **verSaldo(): double** → Retornará el saldo actual de la cuenta.
- **listarMovimientos(): void** → Mostrará por consola todos los movimientos en el orden en el que fueron ejecutados.

Desde el método main de una clase Test, se pide:

- Crear el cliente Juan Lucero de la localidad de Merlo, nacido el 25/07/1978
- Crear una cuenta nro 1234 para el cliente creado en el punto anterior.
- Depositar en la cuenta 60.000 (sesenta mil) pesos.
- Extraer de la cuenta 10.000 (diez mil) pesos.
- Mostrar por consola el saldo de dicha cuenta.
- Mostrar usando el método de cuenta todos los movimientos.

3) Dado el siguiente modelo, implementar según la consigna.



Descripción de los métodos de la clase Cine:

agregarPelicula(Pelicula):void → Recibe una Película y la agrega a la lista que posee el Cine.

listarTodo():void → Muestra por consola todas las Películas registradas en el Cine.

listarDuranMas(horas):void → Reciba una cantidad de horas y muestra por consola todas las películas que tengan una duración mayor a la recibida por parámetro.

listarMenorDuracionAMayorDuracion():void → Lista las Películas ordenadas de menor a mayor.

listarOrdenadasPorTituloAZ():void → Lista las Películas ordenadas alfabéticamente por el título.

listarOrdenadasPorDirectorAZ():void → Lista las Películas ordenadas alfabéticamente por el director.

Desde el método main de una clase Test, se pide:

- Crear el Cine ROMA SRL, ubicado en Rivadavia 325.
- Definir un bucle que crea un objeto Película pidiéndole al usuario todos sus datos y guardándolos en el objeto Película. Después, esa Película se guarda en el Cine usando su método *agregarPelicula* y se le pregunta al usuario si quiere crear otra Película o no.
- Pedir al Cine que liste las películas que duren más de x cantidad de horas que el usuario ingresará por teclado.
- Pedir al cine que liste todas las películas.
- Pedir al cine que liste las películas ordenadas por título.
- Pedir al cine que liste las películas ordenadas por director.
- Pedir al cine que liste las películas ordenadas por duración.

4) Basados en el ejemplo anterior, ahora nos piden que las Películas no deben repetirse en el Cine, para ello en lugar de utilizar un ArrayList, utilizaremos un HashSet; pero ojo!!! Para que el HashSet sepa cuando una Película está repetida, es decir, tienen el mismo título, bastará con sobrescribir los métodos *equals* y *hashCode* en la clase Película.

5) Un Colegio nos pide un sistema para llevar un registro de los alumnos que se encuentran cursando actualmente. Los datos que necesita de un Alumno son: legajo, apellido, nombre y año que cursa. El sistema debe permitir agregar alumnos al Colegio sin que se repitan, quitar alumnos en base a su legajo, listar los alumnos ordenados alfabéticamente por su apellido y listar los alumnos cuyos apellidos comiencen por un carácter en particular. Luego desde el método main de una clase Test, nos piden:

- Instanciar un Colegio.
- Definir un bucle que crea un objeto Alumno pidiéndole al usuario todos sus datos y guardándolos en el objeto Alumno. Después, ese Alumno se guarda en el Colegio usando el método de Colegio que usted implementó y se le pregunta al usuario si quiere crear otro Alumno o no.
- Quitar uno de los alumnos cargados.
- Listar los alumnos ordenados por apellido.
- Listar los alumnos cuyo apellido comience con la letra que el usuario ingresará por teclado.

Importante: Armar el modelo UML representando las clases necesarias. Implementar en java.

6) La tienda RopaLinda SA, nos contrata para que armemos una aplicación que permita crear un Catálogo con los distintos productos que ofrecerá a sus clientes. La tienda posee 3 tipos de tipos de productos: Ropa, Electrodomesticos y Perfumeria. Todos los Productos tienen código, descripción, marca, precioLista y stock; la Ropa además posee como atributo: tipoDeTela; los Electrodomesticos poseen como atributo adicional: consumoEnW y los productos de Perfumería: tamañoEnCC (Tamaño en centímetros cúbicos). Todos los productos tienen un método calcularPrecioPublico, pero que cada tipo de producto implementará de la siguiente forma:

Los Electrodomésticos incrementarán un 25% el precioLista.

La Ropa incrementará un 40% el precioLista.

Los artículos de Perfumeria incrementarán un 20% por cada 100cc.

El Catalogo deberá permitir:

- a) Agregar productos
- b) Informar la cantidad de productos que posee de cada categoría: Ropa, Perfumería y Electrodomésticos.
- c) Listar los productos que son Electrodomesticos.
- d) Listar los productos que son Ropa.
- e) Listar los productos que son de Perfumería.

Luego desde el main, desde una clase Test se pide:

- Instanciar un Catálogo.
- Crear manualmente 2 productos de cada categoría.
- Agregar los productos creados al Catálogo.
- Solicitar al Catálogo la cantidad de productos por categoría.
- Solicitar al Catálogo un listado de los productos de cada categoría.

Importante: Armar el modelo UML representado las clases necesarias. Implementar en java. Haciendo uso de la herencia y polimorfismo.