EJERCICIOS COLECCIONES

Ejercicio1. ArrayList de Personas

- Crearás una clase *Persona* con los atributos DNI, nombre y edad.
- Usarás un ArrayList para guardar las personas, llamado *listaPersonas*;
- El array estará ordenado en todo momento, según la edad. ¿Cuando ordenamos?
- Cuando haya que imprimir el arrays, hacerlo con un foreach usando System.out.println(persona)
- Hacer un menú. Puedes eliminar la opción 4 y mostrar siempre la lista al principio.
- Meter en código un par de personas:

MENU
1. Añadir persona
2. Eliminar persona
3. Detalles de persona
4. Listar todas las personas

```
55555555A – Mortadelo – 18
44444444A – Filemon – 17
```

- Añadir ya desde el programa una persona con DNI 11111111A y ver que se ordena y se coloca en primer lugar.

MEJORA: crear el método
`eliminarPersona(String dni, ArrayList<Persona> lista)—>boolean`
para eliminar una persona. Devuelve ***`true`*** si ha ido bien o `false` si no existe dicha persona.

Ejercicio2. ArrayList del Ferry

Se quiere implementar el control de los vehículos que entran y salen en un Ferry. Para ellos, crearemos dos clases:

- la clase *Vehiculo*, con sus aspectos comunes como matrícula, propietario. Incluye los métodos que considere oportunos.
- la clase Ferry, la cual contrendrá dos atributos: nombre del ferry y la lista de vehículos cargados. Implementar esta lista con un ArrayList. Los métodos serán:
  - `boolean embarcar(String matricula, String propietario)`. Creara un nuevo objeto de tipo vehículo y lo añadirá a la lista del ferry. Devuelve `true` si ha ido bien y `false` en caso de que el vehículo ya exista*.
  - `boolean desembarcar(String matricula)`. Eliminará de la lista del ferry el vehículo con dicha matrícula. Devuelve `true` en caso correcto y `false` si el vehículo no estaba en la lista*.
  - void listarVehiculos( ). Imprime la lista de vehículos ordenada por matrícula.**

- `Vehiculo buscarVehiculo (String matricula)`. Devolverá el vehículo buscado si existe e imprimirá sus datos, en otro caso devolverá null.

```
** EJERCICIO 2 - ARRAYLIST FERRY **
**********************************
1. Embarcar vehiculo
2. Desembarcar vehiculo
3. Datos de un vehiculo embarcado
4. Listar vehiculos embarcados
5. Salir
Elige una opcion >
```

* Para que se pueda ordenar, debes implementar el método compareTo(…).

* (HACER SI HAS VISTO HERENCIA) Para ver si una persona existe, puedes usar el método de ArrayList `contains(…)`. Pero, para poder usar dicho método, tu clase Vehiculo debe implementar el método `equals(…)`, para que el arraylist sepa cuando dos vehículos son iguales.

Ejercicio3. HashMap de Agenda ❖

Crea un programa que implemente una agenda de teléfonos. El contenido de la agenda lo guardaremos con un HashMap donde asociaremos el nombre (clave) a número de telefono (valor). Por tanto, definiremos el mapa de la forma HashMap<String,String>.

NOTA: este programa se debería o podría hacerse definiendo una nueva clase Contacto, pero para practicar HashMap y simplificar, no lo vamos a hacer.

Meter el código dos contactos:

```
Emergencias       - 112
Violencia Genero - 016
AA Carmen         - 660112233
```

Hacer un menú con las siguientes opciones:

```
** EJERCICIO 3 - AGENDA **
*********************************
         Emergencias --> 112
    Violencia Genero --> 016
           AA Carmen --> 660112233
*********************************
1. Añadir contacto
2. Eliminar contacto
3. Modificar contacto
4. Ver contacto
5. Salir
Elige una opcion >
```

Ejercicio4. HashMap de Almacen

Crea un programa que maneje el inventario de productos de una tienda. Para ellos, tendremos dos clases:

- la clase Producto tendrá las propiedades de: *descripcion*, *precio, unidades*.
- la clase Almacen tendrá dos atributo: *ultimoCodigo* y *mapaProductos*, que será un HashMap del tipo `HashMap<Integer,Producto>`. El primer campo (la clave) será el código del producto y el segundo campo (el valor) será el producto en sí. Los métodos de esta clase serán:
  - void añadirProductos(Producto p)
  - void modificarPrecio(Integer codigo, double precio)
  - void añadirUnidades(Integer codigo, int unidades)
  - Producto obtenerProducto(Integer codigo)
  - void venderProducto(Integer codigo, int unidades). Cuando queden 10 unidades, mostrar un mensaje avisando de reponer!!!
  - void listarAlmacen()

PROGRAMA PRINCIPAL

Ejecutar este código sin usar menú para ahorrar tiempo.

```
// TODO code application logic here

Almacen mialmacen=new Almacen();

mialmacen.añadirProducto(new Producto("iphone11",900,50));
mialmacen.añadirProducto(new Producto("galaxy12",800,30));
mialmacen.añadirProducto(new Producto("ps3",400,100));

mialmacen.añadirProducto(new Producto("Satisfyer",50,300));

mialmacen.modificarPrecio(1001, 950);

mialmacen.añadirUnidades(1003,100);

mialmacen.venderProducto(1002,1);

mialmacen.venderProducto(1002,20);

mialmacen.venderProducto(1004, 500);

mialmacen.listarAlmacen();
```

(OPCIONAL) Hacer un programa principal con un menú que tenga las opciones vistas anteriormente.

La lista de acciones a realizar

- Introducir en tiempo de compilación los productos de la imagen

```
mialmacen.añadirProducto(new Producto("iphone11",900,50));
mialmacen.añadirProducto(new Producto("galaxy12",800,30));
mialmacen.añadirProducto(new Producto("ps3",400,100));
```

- Añadir el producto Satisfyer, de 50€ y 300 unidades.
- Listar el almacen para ver los códigos
- Subir el precio del iphone a 950€

```
mialmacen.modificarPrecio(1001, 950);
```

- Añadir 100 unidades de la ps3
- Vender un galaxy12
- Vender 20 galaxy12 (debe dar warning)
- Vender 500 Satisfyer (debe dar error)
- Listar el almacen

La salida final del programa debe ser esta:

```
Vendidas 1 de galaxy12
Vendidas 20 de galaxy12  WARNING!! menos de 10 unidades

ERROR!!. Unidades insuficientes de Satisfyer

LISTA DEL ALMACEN
**************************
--------------------------------
Codigo:1001
iphone11          950,00€   50 unidades
--------------------------------
Codigo:1002
galaxy12          800,00€   9 unidades
--------------------------------
Codigo:1003
ps3               400,00€   200 unidades
--------------------------------
Codigo:1004
Satisfyer          50,00€   300 unidades
**************************
```