Ejercicio 1

Crear una lista de todos los números del 1 a 100 que sean divisibles por un número que introduzca el usuario. Este número tiene que estar comprendido entre 2 y 7. Pinta la lista por pantalla.

Ejercicio 2

Crear una lista de enteros que tenga 10 números aleatorios entre 1 y 50. Mostrar la lista por pantalla.

A continuación, pedir por teclado un limite superior e inferior. Eliminar todos aquellos elementos de la lista que estén fuera de los limites incluidos estos. Mostrar el resultado por pantalla

- Crear lista
- Mostrar lista
- Pedir limites
- Eliminar elementos
- Mostrar lista

Ejercicio 3

Tenemos un taller de reparación de coches y queremos realizar un programa con el que gestionar los coches que entran, salen y están en el taller.

Los datos de cada coche que queremos registrar son: propietario, matricula, marca y si está reparado.

Las opciones que queremos dar son:

1.- Entrada nueva reparación: Se registran todos los datos del coche que se recepciona. Se recepciona como reparado = false

2.- Salida de coche reparado: Se busca el coche en el taller por el nombre de propietario. Localizado el coche, se cambia a reparado = true y se entrega al propietario, saliendo del taller

3.- Listado de vehículos: Se listan todos los vehículos en el taller, mostrando todos los parámetros

4.- Salir: salir del programa

Ejercicio 4

Un cine de un pueblo pequeño nos propone hacer una aplicación para controlar las personas de una cola de un cine en los grandes estrenos de películas. El grupo de personas esperan en una cola para sacar una entrada, que tendremos que calcular en base a su edad.

- Menores de 10 años : 1€
- Entre 11 y 17 años : 2€
- Mayores de 18 años hasta 65: 7€
- Mayores de 65 : 2€

Almacenamos una clase persona con los atributos:

- Nombre : String
- Edad : int generado de forma aleatoria entre 5 y 90 años

A medida que los ciudadanos solicitan entradas, se le asigna a la cola. A medida que van entrando, se les saca de la cola.

De cada sesión se almacena:

- Titulo de la película
- Fecha :dd/mm/aa-hh:mm
- Una listas de personas que entran en cada sesión
- Métodos:
    - double cantidadTotalRecaudada()

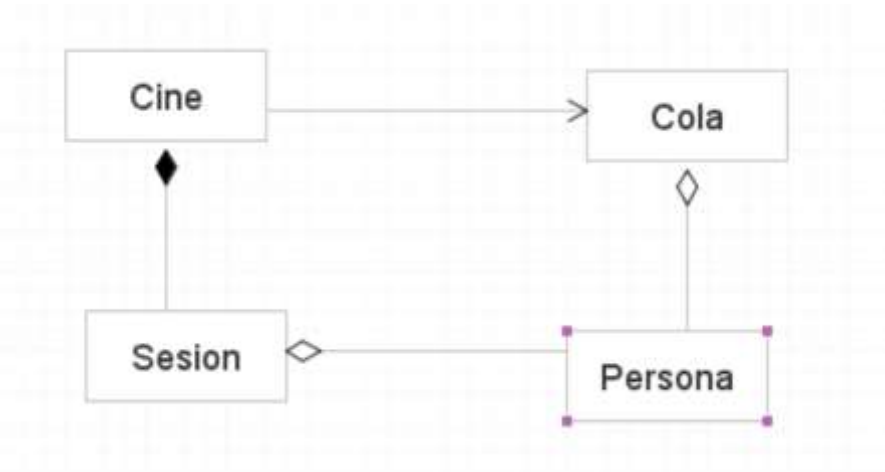
- void añadirPersona(Persona p) : Tener en cuenta que añadir una persona a la sesión implica sacarla de la cola

En la clase Cola se requieren los siguientes métodos:

- void añadirPersona(Persona p)
- Persona sacarPersona()

Métodos de la clase cine

- void añadirPersonaenLaCola(Persona p)
- void crearSesion()
- void añadirPersonaSesion()
- void listarCola()
- void listarSesiones()



Ejercicio 5

Un supermercado nos pide que hagamos una pequeña aplicación que almacene los productos pasados por un escáner. La aplicación debe almacenar productos(clase), cada producto al crearse contiene un precio (estos dos generados aleatoriamente). El nombre del producto será básico (producto1, producto2, producto3, etc.). el precio ya viene con impuestos incluidos.

Crear una clase ticket que contenga entre 1 y 8 productos. Los productos se escogen de forma aleatoria. El constructor de la clase ticket recibe una lista de productos, de los cuales selecciona de forma aleatoria entre 1 y 8 productos máximo. También se debe tener un método mostrar con todo lo vendido en cada ticket y el precio final. Ejemplo:

**********Cantidad****Precio*****Total			
Producto1	5	3.5	17.5
Producto2	7	2.5	17.5
Precio final			35

Ejercicio 6

Un amigo funcionario nos pide que le hagamos una aplicación para sus informes. Debemos gestionar informes que están formados de un código numérico y una tarea que pueden ser ADMINISTRATIVO, EMPRESARIAL y PERSONAL. Debe comprobarse que la tarea es alguna de estas.

Nuestro amigo quiere que seamos capaz de agregar y eliminar informes en forma de pila (el último en entrar, es el primero en salir).

Main:

- Agrega 6 informes
- Muestra su contenido
- Elimina 4 informes
- Muestra su contenido
- Agrega 3 informes
- Muestra su contenido