MANIPULANDO ARRAYS DE TIPOS PRIMITIVOS

1. Crea un programa que rellene un array de 100 posiciones con número aleatorios entre el -1 y el -1000 ambos inclusive. A continuación, realizará un recorrido que vaya sumando todos los números aleatorios almacenados en el array. Por último, se imprimirá el mensaje "La suma de todos los elementos del array es XXXX"

2. Crea un programa que le pida al usuario las notas del alumnado de una clase con 10 matriculados. Una vez termine la introducción de notas, se debe calcular la media aritmética de todas ellas.

   La media aritmética se calcula como la suma de todas las notas dividida entre el número de matriculados.

3. Crea un programa que le pida al usuario el número de alumnos/as matriculados de una clase y a continuación crea un array con tantas posiciones como número de matriculados haya. Ahora debemos recorrer el array rellenándolo con notas enteras aleatorias entre 0 y 100. Por último, tenemos que mostrar los siguientes mensajes:
   La nota más baja de la clase es XXX y está en la posición POS del array.
   La nota más alta de la clase es XXX y está en la posición POS del array.
   La nota media de la clase es XXX.

MANIPULANDO ARRAYS DE OBJETOS

1. Crea un programa que cree un array y lo rellene con las siguientes cadenas:
   {"5432-FNT", "9823-TRH", "3451-LPU", "1937-YHN", "8474-PMG", "8183-TGF", "0293-SDR"}

   Estas cadenas simbolizan matrículas de coches buscados por la policía. Nuestro programa debe pedirle al usuario una matrícula y comprobar si está contenida en el array de matrículas. En el caso de que se encuentre debe imprimir "Llama a la Poli" y, en caso contrario, imprimirá "Matrícula no encontrada"

2. Aprovechando la clase Articulo del ejercicio 44, vamos a crear lo siguiente:

Carrito
+ carrito: array de Articulo + numArticulos: entero + inicializado: lógico
+ inicializaCarrito (numPosiciones: entero) + getNumArticulos () devuelve entero + guardaArticulo (nuevoArtic: Articulo) + muestraArticulos () + buscaArticuloPorId (idABuscar: entero) devuelve Articulo + getPrecioTotalCarrito () devuelve real

Teniendo en cuenta las siguientes restricciones:

- La propiedad *numArticulos* llevará la cuenta del número de artículos introducidos en el carro.
- El método *inicializaCarrito* crea un array de referencias a Articulo con *numPosiciones* posiciones y lo guarda en la propiedad *carrito.* Este método debe ser llamado en primer lugar, antes que cualquier otro del objeto *Carrito*, para garantizarlo usaremos la propiedad lógica *inicializado*, que se pondrá a true cuando ejecutemos este método.
- El método *guardaArticulo* almacena el artículo que se recibe como parámetro en la siguiente posición libre del array. Pero además realizará las siguientes comprobaciones:
    - Si *inicializado* es falso entonces imprimirá el mensaje “Error: carrito aún no inicializado” y terminará el método.
    - Si el número de artículos del carro es igual que el tamaño del array se imprimirá el mensaje “Error: carrito lleno”.
    - Si el objeto que se intenta guardar en el carro es igual a **null** (que funcionará como nuestra marca de "no encontrado") entonces se imprimirá el mensaje “Error: no se puede guardar un artículo nulo”

- El método *muestraArticulos* recorre las posiciones rellenas del array mostrando los artículos que contiene. De cada artículo se mostrará un mensaje como el siguiente: *“Id=XX, nombre=NNNNN, precio=PPP.PP€”.* En el caso de que no tenga ningún artículo se mostrará el mensaje: “No hay artículos en el carrito”. En el caso de que el carrito aún no haya sido inicializado se mostrará “Error: carrito aún no inicializado” y terminará el método.
- El método *buscaArticuloPorId* devuelve el artículo cuya propiedad *id* coincida con la que se recibe como parámetro. En el caso de que no lo encuentre, devuelve un **null**. En el caso de que el carrito aún no haya sido inicializado se mostrará “Error: carrito aún no inicializado”, terminando el método y devolviéndose un **null**.
- El método *getPrecioTotalCarrito* devuelve la suma de los precios de todos los productos del carrito. En el caso de que no haya ningún producto, se devuelve 0.0. En el caso de que el carrito aún no haya sido inicializado se mostrará “Error: carrito aún no inicializado”, terminando el método y devolviéndose un -1.0.

Por último, crea una clase *PruebaCarrito* que cree un objeto Carrito, lo inicialice y pruebe todos los métodos de la clase Carrito.