**Simulación de una Máquina Expendedora**

En este proyecto, crearemos una simulación de una máquina expendedora que vende diferentes tipos de productos. Los usuarios podrán comprar productos si tienen suficiente dinero y si el producto está disponible en la máquina.

## Clase Producto

La clase Producto tendrá los siguientes atributos:

- `nombre`: El nombre del producto.
- `codigo`: El código del producto en cuestión. Deberá ser único y **deberá ser un número**. Deberán ser ordenados y empezando por el 1 (es decir, el primer producto deberá tener el código 1, el segundo e 2, el tercero el 3, etc)
- `precio`: El precio del producto.
- `cantidad`: La cantidad de productos disponibles.

**Métodos:**

- `public Producto(nombre, codigo, precio, cantidad)`: Constructor para inicializar los atributos.
- `public ??? reducirCantidad()`: Reduce la cantidad de productos en 1 **si quedan unidades disponibles**. Devuelve `true` si la operación ha sido exitosa o `false` si no hay stock.
- `public ??? toString()`: Devuelve la información del producto (nombre, codigo, precio, cantidad disponible).

## Clase MaquinaExpendedora

La clase MaquinaExpendedora tendrá los siguientes atributos:

- `producto1, producto2, producto3`: Nuestra máquina expendedora solamente tendrá 3 productos disponibles.
- `saldo`: El saldo actual del usuario.

**Métodos:**

- `public MaquinaExpendedora(???)`: Constructor que recibe los productos y los almacena en la máquina. Inicializa el saldo.
- `public void insertarDinero(cantidad)`: Inserta dinero en la máquina para realizar una compra. Mostrará un mensaje que indique la cantidad de dinero insertado.
- `public int solicitarCodigoProducto()`: Solicitará el código del producto al usuario y lo devolverá. **NOTA:** Habrá que emplear la clase **Scanner** para ello.

- `public void comprarProducto(codigoProducto)`: Permite al usuario comprar un producto según el código del producto y restando el saldo. Si puede comprar el producto, mostrará el nombre del producto que ha sido comprado. Si no tiene suficiente dinero o no hay stock, muestra un mensaje de error. **NOTA:** se deberá emplear la estructura de control **SWITCH** para la selección del producto.
- `private ??? obtenerProductoAleatorio()`: Deberá devolver un producto aleatorio disponible en la máquina expendedora.
- `public ??? mostrarMenu()`: Muestra un mensaje con el menú de productos disponibles, con sus códigos, nombres y precios. **NOTA**: Hay que hacer uso del **toString()** de la clase Producto.
- `public ??? mostrarSaldo()`: Muestra un mensaje por pantalla indicando que devuelve el saldo restante y vacía el saldo de la máquina expendedora.
- `public ??? toString()`: Devuelve un mensaje con la información del saldo actual.

## Consideraciones:

- Todos los **atributos** deberán ser **privados**.
- Deberás emplear **accesores y mutadores** (getters y setters) para aquellos atributos que sean necesarios (no me vale crearlos todos, penalizará los que no se utilicen)
- Deberás emplear la clase **Scanner para seleccionar** el tipo de producto que quieres comprar.
- El código del producto aleatorio será el número “9”.
- El código para devolver el saldo será el número “0”.
- Deberás mostrar toda la información que se muestra en el ejemplo del Resultado Esperado.
    - Información de saldo inicial
    - Menú de productos
    - Petición del código de producto
    - Información de saldo actual después de cada compra
    - etc
- Recuerda que deberás crear una clase Main que permita mostrar todas las funcionalidades requeridas. No hace falta que hagáis ningún tipo de elemento recursivo ni tampoco un bucle. Se puede solicitar un código de producto 4 o 5 veces en el main para simular el ejemplo que se muestra en el resultado esperado.
- **Opcional**: Si se desea que la ejecución continúe hasta que se obtenga un 0, sí será necesario la implementación de un elemento de **recursividad**. Esto **sumaría un punto** extra en el examen, con la posibilidad de obtener un 11 sobre 10.

# Resultado esperado:

**Has insertado 3.0$.**

**Saldo actual: 3.0$.**

**=== MENÚ DE PRODUCTOS ===**
**Elige entre los distintos productos disponibles:**
**1: Producto: Agua | Precio: 1.5$ | Stock: 10**
**2: Producto: Refresco | Precio: 2.0$ | Stock: 5**
**3: Producto: Chocolate | Precio: 1.2$ | Stock: 7**
**9: Producto: Aleatorio | Precio: el correspondiente al producto | Stock: ?**
**0: Devuelve el saldo restante**

**Introduce el codigo del producto deseado: 1**
**Has comprado Agua.**

**Saldo actual: 1.5$.**

**=== MENÚ DE PRODUCTOS ===**
**Elige entre los distintos productos disponibles:**
**1: Producto: Agua | Precio: 1.5$ | Stock: 10**
**2: Producto: Refresco | Precio: 2.0$ | Stock: 5**
**3: Producto: Chocolate | Precio: 1.2$ | Stock: 7**
**9: Producto: Aleatorio | Precio: el correspondiente al producto | Stock: ?**
**0: Devuelve el saldo restante**

**Introduce el codigo del producto deseado: 9**
**No tienes suficiente saldo para comprar el producto: Refresco (cuesta 2.0$ y dispones de 1.5$)**

**Saldo actual: 1.5$.**

**=== MENÚ DE PRODUCTOS ===**
**Elige entre los distintos productos disponibles:**
**1: Producto: Agua | Precio: 1.5$ | Stock: 10**
**2: Producto: Refresco | Precio: 2.0$ | Stock: 5**
**3: Producto: Chocolate | Precio: 1.2$ | Stock: 7**
**9: Producto: Aleatorio | Precio: el correspondiente al producto | Stock: ?**
**0: Devuelve el saldo restante**

**Introduce el codigo del producto deseado: 7**
**No has elegido un codigo de producto valido.**

**Saldo actual: 1.5$.**

**=== MENÚ DE PRODUCTOS ===**
**Elige entre los distintos productos disponibles:**
**1: Producto: Agua | Precio: 1.5$ | Stock: 10**
**2: Producto: Refresco | Precio: 2.0$ | Stock: 5**
**3: Producto: Chocolate | Precio: 1.2$ | Stock: 7**
**9: Producto: Aleatorio | Precio: el correspondiente al producto | Stock: ?**
**0: Devuelve el saldo restante**

**Introduce el codigo del producto deseado: 0**
**Devolviendo 1.5$ de saldo restante, que tengas un buen dia.**