stucom

**Dep. d'Informàtica**

Grupo DAM1

**Módulo 3 - Programació**

Profesores: Vicente Català, Jose Portugal

# Práctica 1 UF4 – Gestión de Tienda

Vamos a mejorar el programa de gestión de tienda en base a unos nuevos requerimientos.

## Documento funcional

### 1. Crear rol empleado para acceder a la aplicación

Se necesita un rol empleado con nombre y número de empleado que servirá como usuario de acceso a la aplicación.

### 2. Ampliar rol cliente con información número de cliente

Además del nombre del cliente se necesita su número de socio y un saldo para comprar productos

### 3. Habilitar login para empleado

Antes de acceder al menú se pedirá un numero de empleado y contraseña, si coincide con la almacenada se accederá al menú principal de la aplicación.

De momento se usarán valores fijos

Número de empleado: 123

Nombre empleado: test

Contraseña: test

```
Introduzca numero de empleado: 123
Introduzca contraseña: test
Login correcto

============================
Menu principal miTienda.com
============================
1) Contar caja
2) Añadir producto
3) Añadir stock
4) Marcar producto proxima caducidad
5) Ver inventario
6) Venta
7) Ver ventas
8) Ver venta total
9) Eliminar producto
10) Salir programa
Seleccione una opción:
```

### 4. Habilitar pago para cliente

En la opción 6) venta antes de finalizar venta, se efectuará un cargo en el saldo del cliente con la cantidad a pagar. Si el cliente no tiene saldo continuar la venta pero mostrar mensaje con la cantidad a deber por parte del cliente.

De momento se usarán valores fijos para todos los clientes.

Número cliente: 456

Saldo inicial en cuenta: 50.00€

```
Seleccione una opción: 6
Realizar venta, escribir nombre cliente
Maria
Introduce el nombre del producto, escribir 0 para terminar:
manzana
Producto añadido con éxito
Introduce el nombre del producto, escribir 0 para terminar:
manzana
Producto añadido con éxito
Introduce el nombre del producto, escribir 0 para terminar:
manzana
Producto añadido con éxito
Introduce el nombre del producto, escribir 0 para terminar:
fresa
Producto añadido con éxito
Introduce el nombre del producto, escribir 0 para terminar:
fresa
Producto añadido con éxito
Introduce el nombre del producto, escribir 0 para terminar:
0
Venta realizada con éxito, total: 83,20€
Cliente debe: -33,20€
```

## Test unitario

1. Verificar empleado con usuario y contraseña valida puede acceder al menú.
2. Verificar empleado con usuario o contraseña incorrecta no puede acceder al menú.
3. Verificar el cliente paga y se descuenta la cantidad de su saldo inicial.
4. Verificar el cliente paga superando su saldo mostrando mensaje cantidad a deber.

# Documento Técnico

1. Nueva clase Person
    a. Crear clase abstract Person en package model
    b. Definir atributo name protegido

2. Nueva interface Logable
    a. Crear interface Logable en package main
    b. Definir metodo login con entrada user(int) y password(String), devuelve boolean

3. Nueva clase Employee
    a. Crear clase Employee en package model
    b. Extender de clase Person
    c. Definir atributo employeeId
    d. Definir constantes finales USER = 123, PASSWORD = “test”
    e. Implementar interface Logable
    f. Implementar metodo login, si usuario(número empleado) y contraseña son como valores fijos (123, test) devolver true si no false.

4. Modificaciones clase Shop con Employee
    a. Modificar metodo main para incluir un nuevo metodo initSession() antes del menú principal
    b. Implementar metodo initSession() pidiendo numero de empleado y contraseña
    c. Crear objeto de tipo Employee e invocar metodo login() para validar sus datos
    d. Si login es false, pedir de nuevo datos, si es true continuar, con sus correspondientes mensajes al usuario.

5. Nueva interface Payable
    a. Crear interface Payable en package main
    b. Definir metodo pay con entrada amount(Amount), devuelve boolean

6. Nueva clase Client
    a. Crear clase Client en package model
    b. Extender de clase Person
    c. Definir atributo memberId(int) y balance(Amount) como saldo inicial
    d. Definir constantes finales MEMBER_ID= 456 , BALANCE = 50.00€
    e. Implementar interface Payable
    f. Implementar metodo pay, recibe de entrada la cantidad de la venta, restar del saldo inicial, si saldo final es positivo devuelve true, si no false

7. Modificaciones clase Sale con Client
    a. Modificar atributo client de clase String a clase Client
    b. Realizar las adaptaciones necesarias

8. Modificaciones clase Shop con Client
    a. Modificar metodo sale() para crear objeto de tipo Client e invocar metodo pay() con el total de la compra
    b. Si pay es true continuar, si es false continuar y mostrar mensaje con la cantidad a deber por parte del cliente.

# Diagrama de clases