Tienda UPM

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Practica de Programación Orientada a Objetos 2025-2026

Universidad Politécnica de Madrid

E.T.S. de Ingeniería en Sistemas Informáticos

Departamento de Sistemas Informáticos

PRACTICA - POO 25-26

Entrega E1:

El cliente solicita un módulo de tickets que permita crear y gestionar productos, aplicar descuentos por categoría y emitir factura de un ticket ordenada alfabéticamente por nombre de producto.

Los productos están definidos por un identificador en el sistema (dos productos no son iguales si tienen diferente ID), numero positivo, que los define dentro del sistema, un nombre no vacío que y de menos de 100 caracteres, una categoría (MERCH, PAPELERIA, ROPA, LIBRO, ELECTRONICA) y un precio que es un número superior a 0 sin límite superior. No podrá haber dos productos con el mismo id en el sistema. No existirán mas de 200 productos diferentes en el sistema para esta versión.

Cuando se genera un ticket se aplican descuentos de manera automática cuando hay más de un producto de la misma categoría, los descuentos son MERCH(0%), PAPELERIA(5%), ROPA(7%), LIBRO(10%), ELECTRONICA(3%).

En la primera versión de la aplicación se arranca con una lista de productos vacía Ticket vacío y se insertan los productos de manera progresiva. Se permite en cualquier momento reiniciar el proceso e iniciar un nuevo ticket. Al imprimir el ticket, se imprime por pantalla y se inicia un nuevo ticket hasta que se cierra la aplicación. Al incorporar un producto al ticket, modificarlo o borrarlo, se debe imprimir el importe provisional del ticket, aplicando los descuentos actuales. El borrado de un producto borrara todas las apariciones del ticket. Para esta versión se asume que los tickets no tendrán más de 100 productos.

Los comandos para implementar son los siguientes:

- *prod add <id> "<nombre>" <categoria> <precio> (agrega un producto con nuevo id)*
- *prod list (lista productos actuales)*
- *prod update <id> campo valor (campos: nombre|categoria|precio)*
- *prod remove <id>*
- *ticket new (resetea ticket en curso)*
- *ticket add <prodId> <cantidad> (agrega al ticket la cantidad de ese producto)*
- *ticket remove <prodId> (elimina todas las apariciones del producto, revisa si existe el id)*
- *ticket print (imprime factura)*
- *help (lista los comandos)*
- *echo "<texto>" (imprime el texto en el valor texto)*
- *exit*

PRACTICA - POO 25-26

Entrada de prueba:

```
help
echo "Agrego Libro"
prod add 1 "Libro POO" BOOK 25
echo "Agrego Camiseta"
prod add 2 "Camiseta talla:M UPM" CLOTHES 15
echo "Listo Productos"
prod list
echo "Actualizo Nombre y Precio del Libro"
prod update 1 NAME "Libro POO V2"
prod update 1 PRICE 30
echo "inserto libro repetido y lo borro"
prod add 3 "Libro POO repetido Error" BOOK 25
prod remove 3
echo "Agrego un producto al ticket e imprimo el ticket"
ticket add 1 2
ticket print
ticket new
echo "Agrego dos productos al ticket e imprimo el ticket"
ticket add 1 2
ticket add 2 1
ticket print
ticket new
echo "Agrego un producto al ticket e inicio un nuevo ticket"
echo "Agrego un producto al ticket e imprimo un nuevo ticket"
ticket add 2 1
ticket new
ticket add 1 2
ticket print
exit
```

PRACTICA - POO 25-26

Salida esperada:

```
Welcome to the ticket module App.
Ticket module. Type 'help' to see commands.
tUPM> help
Commands:
  prod add <id> "<name>" <category> <price>
  prod list
  prod update <id> NAME|CATEGORY|PRICE <value>
  prod remove <id>
  ticket new
  ticket add <prodId> <quantity>
  ticket remove <prodId>
  ticket print
  echo "<texto>"
  help
  exit

Categories: MERCH, STATIONERY, CLOTHES, BOOK, ELECTRONICS
Discounts if there are ≥2 units in the category: MERCH 0%, STATIONERY 5%, CLOTHES 7%, BOOK 10%,
ELECTRONICS 3%.

tUPM> echo "Agrego Libro"
echo "Agrego Libro"

tUPM> prod add 1 "Libro POO" BOOK 25
{class:Product, id:1, name:'Libro POO', category:BOOK, price:25.0}
prod add: ok

tUPM> echo "Agrego Camiseta"
echo "Agrego Camiseta"

tUPM> prod add 2 "Camiseta talla:M UPM" CLOTHES 15
{class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
prod add: ok

tUPM> echo "Listo Productos"
echo "Listo Productos"

tUPM> prod list
Catalog:
  {class:Product, id:1, name:'Libro POO', category:BOOK, price:25.0}
  {class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
prod list: ok

tUPM> echo "Actualizo Nombre y Precio del Libro"
echo "Actualizo Nombre y Precio del Libro"

tUPM> prod update 1 NAME "Libro POO V2"
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:25.0}
prod update: ok

tUPM> prod update 1 PRICE 30
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0}
prod update: ok

tUPM> echo "inserto libro repetido y lo borro"
echo "inserto libro repetido y lo borro"

tUPM> prod add 3 "Libro POO repetido Error" BOOK 25
{class:Product, id:3, name:'Libro POO repetido Error', category:BOOK, price:25.0}
prod add: ok
```

```
tUPM> prod remove 3
{class:Product, id:3, name:'Libro POO repetido Error', category:BOOK, price:25.0}
prod remove: ok

tUPM> echo "Agrego un producto al ticket e imprimo el ticket"
echo "Agrego un producto al ticket e imprimo el ticket"

tUPM> ticket add 1 2
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 60.0
Total discount: 6.0
Final Price: 54.0
ticket add: ok

tUPM> ticket print
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 60.0
Total discount: 6.0
Final Price: 54.0
ticket print: ok

tUPM> ticket new
ticket new: ok

tUPM> echo "Agrego dos productos al ticket e imprimo el ticket"
echo "Agrego dos productos al ticket e imprimo el ticket"

tUPM> ticket add 1 2
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 60.0
Total discount: 6.0
Final Price: 54.0
ticket add: ok

tUPM> ticket add 2 1
{class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 75.0
Total discount: 6.0
Final Price: 69.0
ticket add: ok

tUPM> ticket print
{class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 75.0
Total discount: 6.0
Final Price: 69.0
ticket print: ok

tUPM> ticket new
ticket new: ok

tUPM> echo "Agrego un producto al ticket e inicio un nuevo ticket"
echo "Agrego un producto al ticket e inicio un nuevo ticket"

tUPM> echo "Agrego un producto al ticket e imprimo un nuevo ticket"
echo "Agrego un producto al ticket e imprimo un nuevo ticket"
```

```
tUPM> ticket add 2 1
{class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
Total price: 15.0
Total discount: 0.0
Final Price: 15.0
ticket add: ok

tUPM> ticket new
ticket new: ok

tUPM> ticket add 1 2
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 60.0
Total discount: 6.0
Final Price: 54.0
ticket add: ok

tUPM> ticket print
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
Total price: 60.0
Total discount: 6.0
Final Price: 54.0
ticket print: ok

tUPM> exit
Closing application.
Goodbye!

Process finished with exit code 0
```

Entregables

- Código fuente y empaquetado (jar) comprimidos en un zip subido a moodle.
- Diagrama UML del modelo propuesto en un formato legible (PNG, JPG, SVG,…). Se deberá justificar el diseño de clases y uso de librerías (si lo hubiera).

Consideraciones

- Para poder defender el proyecto el código entregado no debe tener errores de compilación.
- Se debe respetarse el formato de entrada de comandos y salida mostrados en el enunciado de la práctica.
- No se pueden crear más comandos de los pedidos en el enunciado.
- Para poder defender el proyecto el código debe ser capaz de ejecutar sin errores todos los comandos del ejemplo propuesto. Este punto será verificado en la defensa.