ORACLE Academy

Conceptos fundamentales de Java

Sección 7, parte 1: Creación de un proyecto de inventario

Solución del proyecto

Visión general

Este proyecto avanzará al ritmo del usuario en las secciones 4, 5, 6 y 7 del curso. Después de cada sección, se podrán realizar más aportaciones hasta que se cree una aplicación Java completa para mantener el inventario. En cada parte, tome como base la última parte de modo que se cumplan tanto los requisitos anteriores como los nuevos. Incluya todas las partes en un paquete denominado inventario.

Cree un programa de inventario que se pueda utilizar para una serie de productos distintos (CD, DVD, software, etc.).

Tema(s):

- Modificación de programas
  - Creación de métodos estáticos (sección 7.3)
  - Uso de parámetros en un método (sección 7.1)
  - Devolver un valor desde un método (sección 7.1)
- Adición de métodos (comportamientos) a una clase existente (sección 7.2)
- Implementación de una interfaz de usuario (secciones 5.1, 5.2, 6.2)

1. Abra el programa de inventario que se ha actualizado en la **sección 6: Creación de un proyecto de inventario**

2. Ahora va a modificar el código para que la clase main no realice ningún procesamiento, sino que solo llame a métodos estáticos cuando sea necesario.

   a) Cree un método estático en la clase **ProductTester** después del final del método main denominado displayInventory. Este método no devuelve ningún valor y acepta la matriz de productos como parámetro. *Recuerde* que, al transferir una matriz como un parámetro, debe utilizar el nombre de clase como el tipo de datos, un conjunto de corchetes vacíos y, a continuación, el nombre de la matriz (*ClassName*[] *arrayName*).

   b) Copie el código que muestra la matriz del método main en el nuevo método displayInventory.

   c) Donde ha eliminado el código de visualización de main, sustitúyalo con una llamada al método en el método displayInventory. No olvide incluir la lista correcta de argumentos para que coincida con la lista de parámetros en el método que está llamando.

   d) Ejecute y pruebe el código.

   e) Cree un método estático en la clase **ProductTester** después del final del método main denominado addToInventory. Este método no devuelve ningún valor y acepta la matriz de productos y el escáner como parámetros.

   f) Copie el código que añade los valores a la matriz del método main en el nuevo método addToInventory.

   g) Para resolver los errores que haya en el código, mueva las variables locales necesarias (tempNumber, tempName, tempQty, tempPrice) del método main a la parte superior del método addInventory.

h) Agregue una llamada al método en main al método addToInventory() donde ha eliminado el bucle for.

i) Ejecute y pruebe el código.
j) Cree un método en **ProductTester** que devolverá un valor entero denominado getNumProducts() que acepta el escáner como parámetro. Mueva todo el código que obtiene el número máximo de productos del usuario a este método, ponga una llamada al método en main en su nuevo método. Almacenará el valor devuelto en la variable maxSize, por lo que tendrá que declarar 2, uno en el método main y otro en el método getNumProducts(). Puede eliminar el valor inicial de -1 de la declaración en main.

k) Ejecute y pruebe el código.

3. Cree dos nuevos métodos en la clase **Product**, uno que permitirá al usuario sumar al número de unidades en existencias (addToInventory), y otro que permitirá al usuario restar del número de unidades en existencias (deductFromInventory). Ambos métodos deben aceptar un parámetro (cantidad) que contiene el número de elementos que se van a sumar o restar. Colóquelos debajo de los constructores.

4. Modifique la clase **ProductTester** para que el usuario pueda ver, modificar o descatalogar los productos a través de una interfaz de usuario que se basa en un sistema de menús.
    a) Muestre un sistema de menús que mostrará opciones y devolverá la opción de menú introducida por el usuario.

        i. El método se debe llamar getMenuOption, debe devolver un valor entero y utilizar un objeto Scanner como parámetro. Escriba el código debajo de main.

        ii. El menú se debe parecer a lo siguiente:

```
1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit
Please enter a menu option:
```

        iii. Solo se deben aceptar números entre 0 y 4, cualquier otra entrada debe obligar a que se vuelva a presentar la solicitud al usuario. **Recuerde** que al agregar una sentencia try catch, debe inicializar la variable en algo que produzca un error en la condición while.

    b) Cree un método que muestre el valor de índice de la matriz y el nombre de cada producto, lo que permite al usuario seleccionar el producto que se quiere actualizar (sumar/restar).

        i. El método se debe llamar getProductNumber, debe devolver un valor entero y utilizar la matriz de productos y un objeto Scanner como parámetros. Debe tener una única variable local denominada productChoice de tipo entero que se inicializa en -1. Escriba el código debajo de main.

        ii. Se debe utilizar un bucle FOR tradicional para mostrar el valor de índice y el nombre del producto. Utilice la longitud de la matriz para terminar el bucle. Se puede acceder al nombre de cada producto mediante su método getter correspondiente.

        iii. Solo se debe permitir que el usuario introduzca valores entre 0 y 1, inferiores a la longitud de la matriz. Todas las entradas deben tener el manejo de errores adecuado.

    c) Cree un método que agregue valores de existencias a cada producto identificado.
        i. El método se debe llamar addInventory, no debe tener valor de retorno y debe utilizar la matriz de productos y un objeto Scanner como parámetros. Debe tener dos variables locales denominadas productChoice que no tengan que inicializarse y otra denominada updateValue que se debe inicializar en -1. Ambas variables locales deben poder almacenar valores enteros. Escriba el código debajo de main.

ii. Agregue una llamada al método en el método getProductNumber, transfiera los parámetros correctos y guarde el resultado en la variable productChoice.

iii. El usuario debe ver el mensaje "¿Cuántos productos quiere agregar?" y solo se le permitirá introducir valores positivos a partir de 0. Todas las entradas deben tener el manejo de errores y los mensajes de error correspondientes.

iv. Una vez que se ha agregado el valor de actualización válido, los niveles de existencias del producto seleccionado se deben actualizar con el método addToInventory que ha creado anteriormente. La variable productChoice se utiliza para identificar el valor de índice del producto en la matriz y updateValue es la cantidad de existencias que se va a agregar.

d) Cree un método que reste valores de existencias a cada producto identificado.
   i. Siga el mismo procedimiento que llevó a cabo para agregar existencias pero asigne el nombre deductInventory al método. Las restricciones en sus entradas son que el valor debe ser 0 o superior y no puede ser superior a la cantidad actual de existencias del producto. Todas las entradas deben tener el manejo de errores y los mensajes de error correspondientes. Utilice el método deductFromInventory para realizar el cambio en el objeto de producto en la matriz.

e) La última opción de menú que implementar es la capacidad de marcar existencias como descatalogadas.
   i. El método se debe llamar discontinueInventory, no debe tener valor de retorno y debe utilizar la matriz de productos y un objeto Scanner como parámetros. Debe tener una única variable local denominada productChoice que almacena un valor entero y no se tenga que inicializar. Escriba el código debajo de main.

   ii. Agregue una llamada al método en el método getProductNumber, transfiera los parámetros correctos y guarde el resultado en la variable productChoice.

   iii. Ahora debe utilizar el método setActive para definir el valor Activo en false para el objeto de producto seleccionado.

f) Ahora, es necesario crear un método que lo reúna todo.
   i. El método se debe llamar executeMenuChoice, no debe tener valor de retorno y debe utilizar las opciones de menú, la matriz de productos y un objeto Scanner como parámetros. No requiere variables locales. Escriba el código debajo de main.

   ii. Utilice una sentencia switch para ejecutar los métodos que ha creado en este ejercicio. Para cada sentencia case, utilice una sentencia de salida que muestre uno de los siguientes encabezados antes de ejecutar el método correspondiente:

```
View Product List
Add Stock
Deduct Stock
Discontinue Stock
```

g) La etapa final de este ejercicio es actualizar el método main para hacer uso de la nueva funcionalidad. Actualice el código para que el método main coincida con el siguiente código:

```
public static void main(String[] args) {
    //create a Scanner object for keyboard input
    Scanner in = new Scanner(System.in);
    int maxSize, menuChoice;

    maxSize = getNumProducts(in);
    if(maxSize ==0) {
        //Display a no products message if zero is entered
        System.out.println("No products required!");
    }else {
        Product[] products = new Product[maxSize];
        addToInventory(products, in);
        do {
            menuChoice = getMenuOption(in);
            executeMenuChoice(menuChoice, products, in);
        }while(menuChoice!=0);
    }//endif

}//end method main
```

h) Ejecute y pruebe el código.

5. Guarde el proyecto.