UNIVERSIDAD DE LAS FUERZAS ARMADAS

ESPE

Topic: Pitfalls Inspections Grupal

Assignments N°14

Object-oriented programming

NRC: 9642

Team 6: Code Warriors

Members: Ediosn Verdesoto

Leonel Tipan

ING. Jorge Edison Lascano.

```java
private int idCardNumber;
```

Embedded types in name. Could be just idCard or CardNumber

```java
    if (productIndex < catalog.getInventory().getHardwareComponents().size()) {
        HardwareComponent hardwareComponent = catalog.getInventory().getHardwareComponents().get(index: pr
        System.out.println(x: "Ha seleccionado el siguiente componente de hardware:");
        System.out.println(x: catalog.infoForClient(hardwareComponent));
        System.out.print(s: "Ingrese la cantidad a comprar: ");
        int quantity = scanner.nextInt();

        if (quantity <= 0 || quantity > hardwareComponent.getQuantity()) {
            System.out.println(x: "Cantidad invalida o insuficiente.");
            return;
        }

        hardwareComponent.setQuantity(hardwareComponent.getQuantity() - quantity);

        System.out.println(x: "Compra realizada exitosamente.");

    } else {
        Clothing clothing = catalog.getInventory().getClothes().get(productIndex - catalog.getInventory(
        System.out.println(x: "Ha seleccionado la siguiente prenda de ropa:");
        System.out.println(x: catalog.infoForClient(clothing));
        System.out.print(s: "Ingrese la cantidad a comprar: ");
        int quantity = scanner.nextInt();

        if (quantity <= 0 || quantity > clothing.getQuantity()) {
            System.out.println(x: "Cantidad invalida o insuficiente.");
            return;
        }

        clothing.setQuantity(clothing.getQuantity() - quantity);
        System.out.println(x: "Compra realizada exitosamente.");
```

```java
public void showInventory(String dataReaded){
    Gson gson = new Gson();
    Type Inventory = new TypeToken<Inventory>(){}.getType();
    Inventory savedInventory = gson.fromJson(json: dataReaded, typeOfT: Inventory);
    System.out.println(x: savedInventory);
}

public void viewPurchaseRegister(PurchaseRegister purchaseRegister){
    File file = new File(path + fileName + ".json");
    String dataReaded;
    if (file.exists()){
        dataReaded = readData();
        showPurchaseRegister(dataReaded);
    }
    else{

    }

}

public void showPurchaseRegister(String dataReaded){
    Gson gson = new Gson();
    Type PurchaseRegister = new TypeToken<PurchaseRegister>(){}.getType();
    PurchaseRegister savedPurchaseRegister = gson.fromJson(json: dataReaded, typeOfT: PurchaseRegister);
    System.out.println(x: savedPurchaseRegister.getPurchases());
}
```

```
        HardwareComponent purchasedHardwareComponents = new HardwareComponent();
        purchasedHardwareComponents.setId();
        System.out.println(x: "Ingrese el nombre del componente: ");
        purchasedHardwareComponents.setName(name: keyboardInput.nextLine());
        System.out.println(x: "Ingrese el modelo del componente: ");
        purchasedHardwareComponents.setModel(model: keyboardInput.nextLine());
        System.out.println(x: "Ingrese el cantidad del componente: ");
        purchasedHardwareComponents.setQuantity(quantity: keyboardInput.nextInt());
        keyboardInput.nextLine();
        System.out.println(x: "Ingrese el costo del componente: ");
        purchasedHardwareComponents.setIndividualCost(individualCost: keyboardInput.nextDouble());
        keyboardInput.nextLine();
        System.out.println(x: "Ingrese el precio de venta del componte: ");
        purchasedHardwareComponents.setIndividualPrice(individualPrice: keyboardInput.nextDouble());
        keyboardInput.nextLine();
        System.out.println(x: "");
        ui.messagePurchase();
        System.out.println(x: purchasedHardwareComponents);
        return purchasedHardwareComponents;
    }

    public Clothing createPurchaseClothing(){
        Clothing purchaseClothings = new Clothing();
        purchaseClothings.setId();
        System.out.println(x: "Ingrese el nombre de la prenda: ");
        purchaseClothings.setName(name: keyboardInput.nextLine());
        System.out.println(x: "Ingrese el modelo de la prenda: ");
        purchaseClothings.setModel(model: keyboardInput.nextLine());
        System.out.println(x: "Ingrese el cantidad de la prenda: ");
        purchaseClothings.setQuantity(quantity: keyboardInput.nextInt());
        keyboardInput.nextLine();
        System.out.println(x: "Ingrese el costo de la prenda: ");
        purchaseClothings.setIndividualCost(individualCost: keyboardInput.nextDouble());
        keyboardInput.nextLine();
        System.out.println(x: "Ingrese el precio de venta de la prenda: ");
        purchaseClothings.setIndividualPrice(individualPrice: keyboardInput.nextDouble());
        keyboardInput.nextLine();
```

Oddball solution. ShowInventory and showPurchaseRegister, and many more are very similar. Maybe refactor using a method extraction to just call the method and change the variables.

```
public Purchase() {

}
```

```
public SalesRegister() {
}
```

```
public PurchaseRegister() {
}
```

Useless methods.