Métricas y Calidad de Software

Cesar Solano c.solanor@uniandes.edu.co

Refactorización 2

Al ser Java se aprovecha el paradigma de la programación orientada a objetos:

1. Extraer funciones y separación de código: Se identifican los pasos que hay para calcular los inventarios para establecer los métodos que se separarán.

```java
public static void main(String[] args) {
    String csvFileProducts = "./data/products.csv";
    String csvFileSales = "./data/sales.csv";
    String csvFileOrders = "./data/orders.csv";

    System.out.println(csvFileProducts);

    Store store = new Store();

    store.setProductsFromCsv(csvFileProducts);
    store.setSalesFromCsv(csvFileSales);
    store.setOrdersFromCsv(csvFileOrders);

    store.updateInventory();

    System.out.println(store.printInventory());
}
```

2. Combinar funciones – Encapsular records: Se crea una nueva clase Store, que encapsulará los atributos necesarios para hacer las operaciones.

3.
```
package refactoring.problema3;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class Store {
    private ArrayList<Product> products;
    private ArrayList<Sale> sales;
    private ArrayList<Order> orders;
    private static final String CSV_SPLIT_BY = ",";

    public Store(){}

    public void setProductsFromCsv(String productsCsv) {
        this.products = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new
FileReader(productsCsv))) {
            String line = br.readLine();

            while ((line = br.readLine()) != null) {
                String[] data =
line.split(CSV_SPLIT_BY);

                // Access the product data
                int itemId = Integer.parseInt(data[0]);
                String item = data[1];
                int quantity =
Integer.parseInt(data[2]);

                this.products.add(new Product(itemId,
item, quantity));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void setSalesFromCsv(String salesCsv) {
        this.sales = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new
FileReader(salesCsv))) {
            String line = br.readLine();
            while ((line = br.readLine()) != null) {
                String[] data =
line.split(CSV_SPLIT_BY);

                int saleId =
Integer.parseInt(data[0].trim());
                String saleDate = data[1].trim();
                int itemId =
Integer.parseInt(data[2].trim());
                int quantity =
Integer.parseInt(data[3].trim());

                Sale sale = new Sale(saleId, saleDate,
itemId, quantity);
                this.sales.add(sale);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void setOrdersFromCsv(String ordersCsv) {
        this.orders = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new
FileReader(ordersCsv))) {
            String line = br.readLine();

            while ((line = br.readLine()) != null) {
                String[] data =
line.split(CSV_SPLIT_BY);

                int orderId =
Integer.parseInt(data[0].trim());
                String orderDate = data[1].trim();
                int itemId =
Integer.parseInt(data[2].trim());
                int quantity =
Integer.parseInt(data[3].trim());

                this.orders.add(new Order(orderId,
orderDate, itemId, quantity));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void updateInventory(){
        for (Order order : orders) {
            Product item =
products.get(order.getItemId());
            item.setQuantity(item.getQuantity() +
order.getQuantity());
        }

        for (Sale sale : sales) {
            Product item =
products.get(sale.getItemId());
            item.setQuantity(item.getQuantity() -
sale.getQuantity());
        }
    }
    public String printInventory(){
        StringBuilder sb= new StringBuilder();
        for (Product product : products) {
            sb.append(product.getItem());
            sb.append(" ");
            sb.append(product.getQuantity());
            sb.append("\n");
        }
        return sb.toString();
    }
}
```