Clean Code Inspection to Team 3 / Software Juniors

By: Team 2 KillChain

Recommendations & corrections.

BillingSystem

```
}catch (IOException e) {


}
```

No message for the exception.

DressManager

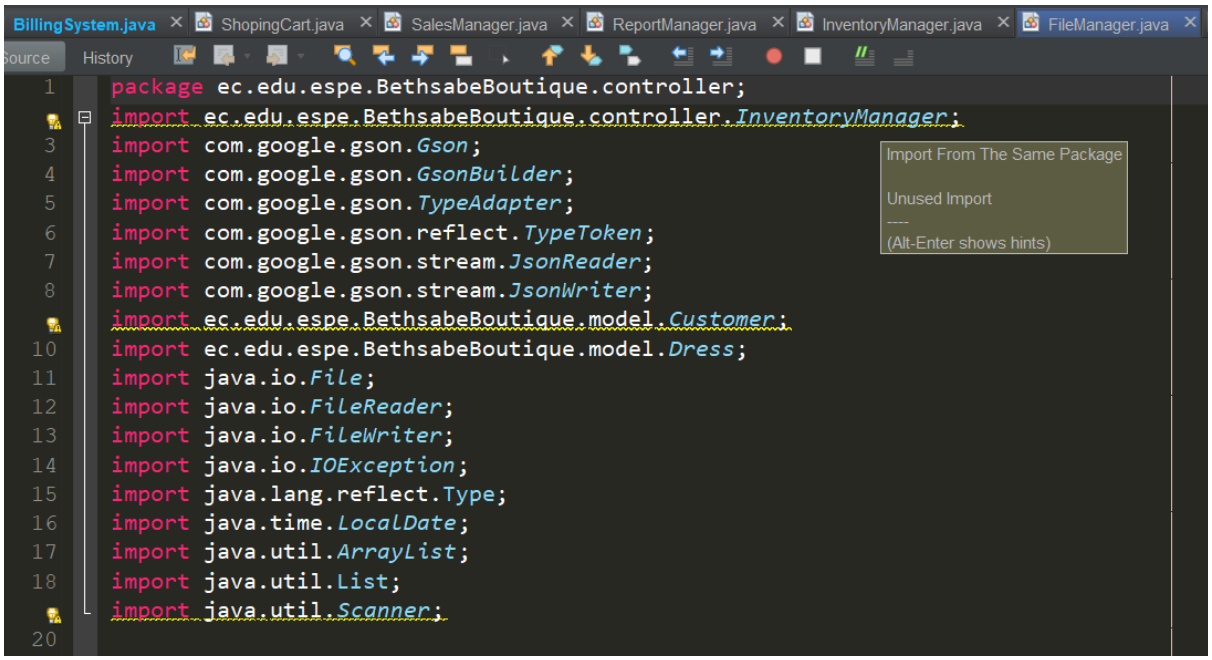
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ec.edu.espe.BethsabeBoutique.controller;

import ec.edu.espe.BethsabeBoutique.model.Dress;
import java.util.ArrayList;
import java.util.Scanner;
```

Default header.

```
    if (dressFounded == false) {
        System.err.println("No hay ningun vestido con el nombre: " + name + " en la base de datos");
        System.out.println( x: "Revise que el nombre este escrito correctamente e intelo nuevamente");
    }
    return null;
}
```

Is not recommended to return null, but the first thing to correct here is the message that has an spelling error.

FileManager



Imported classes without use.

```
public void createCsv(ArrayList<Dress> dressList, String reportName) {
    //Creates the folder "Reportes" in root
    File file = new File(pathname: "Reportes");
    file.mkdir();
    String[] reportsList = file.list(); //Obtains all the reports in "Reportes"
```

Some comments that are not obligatory.

```
} catch (IOException e) {
    e.printStackTrace();
}
```

Is better to put a message that the customer can understand.

```
public void createBill(){

}
```

Useless function.

InventoryManager

```
public void addDress() {

    Dress dress = dressManager.getDressInformation();
    dress.setDateReceived(dateReceived: LocalDate.now());
    getDressList().add(e: dress);
    fileManager.createJson(dressList);
}

public void editDress() {
    Dress dress = dressManager.searchDress();
    if (dress != null) {
        Dress updatedDress = dressManager.getUpdatedDressInformation();
        dress.setName(name: updatedDress.getName());
        dress.setBrand(brand: updatedDress.getBrand());
        dress.setSize(size: updatedDress.getSize());
        dress.setQuantity(quantity: updatedDress.getQuantity());
        System.out.println("Vestido -" + dress.getName() + "- editado exitosamente");
    }
}

public void deleteDress() {
    Dress dress = dressManager.searchDress();
    if (dress != null) {
        getDressList().remove(o: dress);
        System.out.println("Vestido -" + dress.getName() + "- Eliminado");
        fileManager.createJson(dressList);
    }
}
```

Declaration of a Dress object multiple times in the same class instead of declaring one global.

ReportManager

```
1  package ec.edu.espe.BethsabeBoutique.controller;
2
3  import ec.edu.e                    outique.model.Dress;
4  import java.uti
5  import java.util.List;
6  import java.util.Scanner;
7
8  /**
9   *
10  * @author Caetano Flores, Juniors, DCCO-ESPE
11  */
12 public class ReportManager {
13     Scanner scanner = new Scanner(source: System.in);
14     FileManager fileManager = new FileManager();
15
16     public void createReport(ArrayList<Dress> dressList){
17         String reportName;
18         System.out.println(x: "Ingrese el nombre del reporte: ");
19         reportName = scanner.next();
21         fileManager.createCsv(dressList, reportName);
22     }
23 }
```

Unused Import

(Alt-Enter shows hints)

Unused import.

SalesManager

```
String name, email, idCard;
```

More than one variable declared in the same line.

ShoppingCart

```
package ec.edu.espe.BethsabeBoutique.controller;

import ec.edu.espe.BethsabeBoutique.model.Dress;
import java.awt.BorderLayout;
import java.util.ArrayList;
import java.util.Scanner;

/**
 *
 * @author Caetano Flores, Juniors, DCCO-ESPE
 */
public class ShopingCart {
    private float totalPrice;
    private ArrayList<Dress> cart
    Scanner scanner = new Scanner
    InventoryManager inventoryManager = new InventoryManager();
    DressManager dressManager = new DressManager();
    SalesManager salesManager = new SalesManager();
```

Variable inventoryManager is never read

(Alt-Enter shows hints)

Class & Import without use.

```
public Dress searchOnCart(Dress dressToSearch) {

        for (Dress dress : getCart()) {
            if (dress.getName().equalsIgnoreCase(anotherString: dressToSearch.getName())) {
                return dress;
            }
        }

    return null;
}
```

Bad Indentation.

```
int cartIndex = 1, option = 0;
```

More than 1 variable declared in the same line.

Customer

```
    /**
     * @param email the email to set
     */
    public void setEmail(String email) {
        this.email = email;
    }

    /**
     * @return the idCard
     */
    public String getIdCard() {
        return idCard;
    }

    /**
     * @param idCard the idCard to set
     */
    public void setIdCard(String idCard) {
        this.idCard = idCard;
    }
    private String name;
    private String email;
    private String idCard;
```

Declaration of attributes after the methods.

Dress

```
    private LocalDate dateReceived;
    //private LocalDate dateSold;

    public Dress(String name, String brand, String size, float price, int quantity) {
        this.name = name;
        this.brand = brand;
        this.size = size;
        this.price = price;
        this.quantity = quantity;
    }

    // Getters and setters

    // ...

    public LocalDate getDateReceived() {
        return dateReceived;
    }

    public void setDateReceived(LocalDate dateReceived) {
        this.dateReceived = dateReceived;
    }

    /*public LocalDate getDateSold() {
        return dateSold;
    }*/

    /*public void setDateSold(LocalDate dateSold) {
        this.dateSold = dateSold;
    }*/
```

Commented code & unnecessary comment about getters and setters.

PD: The class SalesSystem that is declared in the Class Diagram is not in the program, we think it's the SalesManager, so, consider renaming the class in the diagram or in the program, and check the attributes name are according to the diagram because there are some changed names, also add the classes ShoppingCard and BillingSystem to the diagram.