

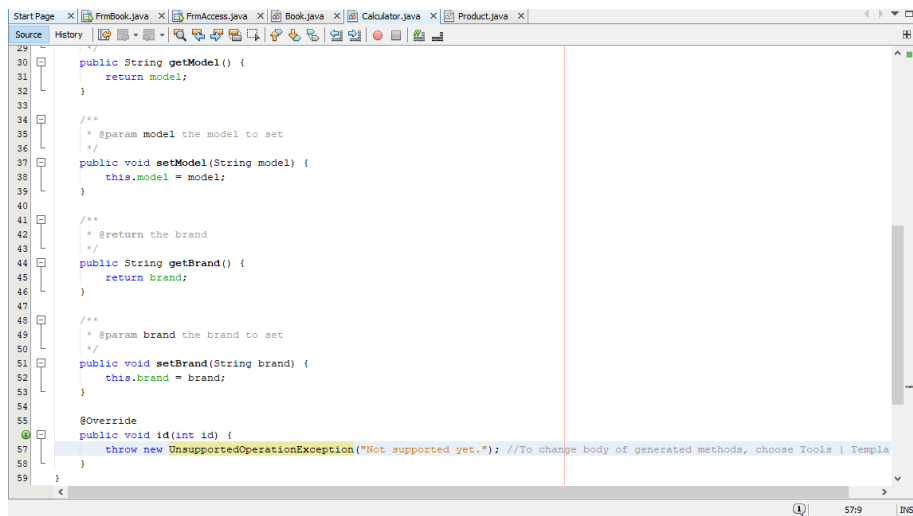
**Name:** Cajas Karla, Campoverde Carlos, Chablay Esteban, Chamorro Myckel

**Team:** EMCL.JAVA

**NRC:** 3682

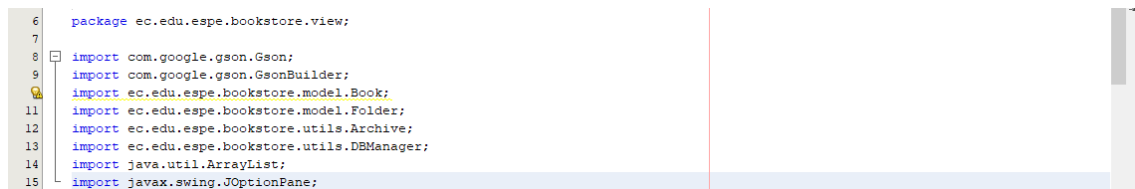
**Evaluated team:**

- contains in classes override unnecessary



```
29  
30 public String getModel() {  
31     return model;  
32 }  
33  
34 /**  
35  * @param model the model to set  
36  */  
37 public void setModel(String model) {  
38     this.model = model;  
39 }  
40  
41 /**  
42  * @return the brand  
43  */  
44 public String getBrand() {  
45     return brand;  
46 }  
47  
48 /**  
49  * @param brand the brand to set  
50  */  
51 public void setBrand(String brand) {  
52     this.brand = brand;  
53 }  
54  
55 @Override  
56 public void id(int id) {  
57     throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates  
58 }  
59 }
```

- use of unnecessary libraries



```
6 package ec.edu.espe.bookstore.view;  
7  
8 import com.google.gson.Gson;  
9 import com.google.gson.GsonBuilder;  
10 import ec.edu.espe.bookstore.model.Book;  
11 import ec.edu.espe.bookstore.model.Folder;  
12 import ec.edu.espe.bookstore.utils.Archive;  
13 import ec.edu.espe.bookstore.utils.DBManager;  
14 import java.util.ArrayList;  
15 import javax.swing.JOptionPane;
```

- empty method, it is not encapsulated



```
Start Page x FrmBook.java x FrmAccess.java x Book.java x Calculator.java x Product.java x Customer.java x Folder.java x Inventory.java x
Source History
18 private ArrayList<Book> books = new ArrayList<>();
19
20 public void addBook(Book book) {
21     books.add(book);
22 }
23
24 public void removeProduct(Book book) {
25     //int id = chicken.getId();
26     books.remove(book);
27 }
28
29 //sobrecarga de funciones -> overloading
30 public void removeProduct(int id) {
31     Iterator iterator = books.iterator();
32     while(iterator.hasNext()) {
33         Book book = (Book)iterator.next();
34         if(id == book.getId()) {
35             iterator.remove();
36         }
37     }
38 }
39
40 public void resetIterator() {
41 }
42
43 String title;
44 String author;
45 public Book nextBook() {
46     return new Book(title, author, id, id, id);
47 }
48 }
```

- about unnecessary writing

```
16 private int id;
17 private ArrayList<Book> books = new ArrayList<>();
18
19 public void addBook(Book book) {
20     books.add(book);
21 }
22
23 public void removeProduct(Book book) {
24     //int id = chicken.getId();
25     books.remove(book);
26 }
27
28 //sobrecarga de funciones -> overloading
29 public void removeProduct(int id) {
30     Iterator iterator = books.iterator();
31     while(iterator.hasNext()) {
32         Book book = (Book)iterator.next();
33         if(id == book.getId()) {
34             iterator.remove();
35         }
36     }
37 }
38
39 public void resetIterator() {
40 }
```

- unnecessary methods



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

```
etory
addBook(Book book)
nextBook() : Book
removeProduct(Book book)
removeProduct(int id)
resetIterator()
author : String
books : ArrayList<Book>
id : int
title : String

35
36
37
38
39 public void resetIterator() {
40
41 }
42 String title;
43 String author;
44 public Book nextBook() {
45     return new Book(title, author, id, id, id);
46 }
47
48
```

- repeating code in classes can create a function

```
144 private void btnSaveMouseClicked(java.awt.event.MouseEvent evt) {
145     String tipType = txtTipType.getText();
146     String color = txtColor.getText();
147     float price = Float.parseFloat(txtPrice.getText());
148     Integer amount = Integer.parseInt(txtAmount.getText());
149
150     if(color.isEmpty() || tipType.isEmpty()){
151         JOptionPane.showMessageDialog(rootPane, "Debe llenar todos los campos");
152     }else{
153         for( int id=1; id <= amount; id++){
154             Pen pens = new Pen(color,tipType,id,price,amount);
155             listPen.add(pens);
156             jsonProduct = gson.toJson(pens);
157             Archive.writeArchive(jsonProduct, "Pens.json");
158             dbManager = new DBManager("pens");
159             dbManager.insert(jsonProduct);
160         }
161
162         FrmPrincipal principal = new FrmPrincipal();
163         principal.setVisible(true);
164         dispose();
165     }
166 }
```