

WS32 - GROUPS

DATE: 03th february 2021

TEAM 2:

Mateo Landazuri
Andy Calderon
Leonel Mantuano
Pablo Bustillos
Richar Maisincho
Sebastian Palacios

- Attributes in different sizes are written with camelcases

```
public class Person {

    private String name;
    private String lastname;
    private String telephoneNumber;
    private String Email;
    private String ID;

    public Person(String name, String lastname, String telephoneNumber, String Email, String ID) {
        this.name = name;
        this.lastname = lastname;
        this.telephoneNumber = telephoneNumber;
        this.Email = Email;
        this.ID = ID;
    }

    private Customer client;
    private int numberOfBill = 1;
    private float priceOfWorkforce;
    private float discount;

    - Unnecessary comments
    /**
     * @return the Email
     */
    public String getEmail() {
        return Email;
    }

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

    /**
     * @return the ID
     */
    public String getID() {
        return ID;
    }

    /**
     * @param ID the ID to set
     */
    public void setID(String ID) {
        this.ID = ID;
    }

61    String name ;
62    int quanity;
63    >>>>> e4b3f82c2e54cd426697e6d1e44a137df95afedb
64    /**
65    }
66 }
```

- Comments not deleted

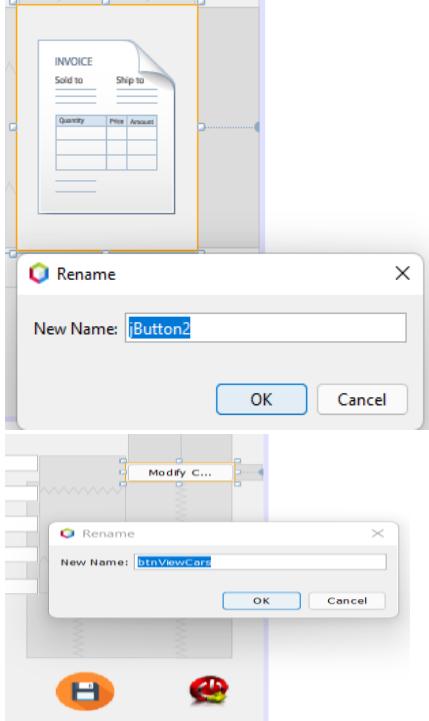
```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.mechanic.utils;

import com.mongodb.BasicDBObject;

```

- Don't change variable names



- Capital letters are not used

```

public class ValueProduct {

    BasicDBObject document = new BasicDBObject();

    public DBObject addProduct(String name, int quantity, int price) {
        document.put("Name", name);
        document.put("Quantity", quantity);
        document.put("Price", price);
        return null;
    }
}

```

- Class names must be singular nouns

```

181     DB db = mongo.getDB(dataBase);
182     DBCollection dbCollection = db.getCollection(collection);
183     BasicDBObject searchedName = new BasicDBObject();
184     BasicDBObject updateData = new BasicDBObject();
try {

```

The screenshot shows a Java code editor with a package structure on the left and a code editor on the right.

Package Structure:

- ec.edu.espe.mechanic.model
 - Accessories.java
 - Administrative.java
 - Bill.java
 - Car.java
 - Customer.java
 - Launderer.java
 - Mechanic.java
 - Motorcycle.java
 - Person.java
 - Product.java
 - Repair.java
 - Service.java
 - SpareParts.java
- SQLDBManager.java
- ec.edu.espe.mechanic.view
 - FrmBill.java
 - FrmCar.java
 - FrmCarManage.java
 - FrmCustomerRecord.java
 - FrmCustomerManage.java
 - FrmEmployers.java
 - FrmLogin.java
 - FrmMotorcycles.java
 - FrmProducts.java
 - FrmRegister.java
 - FrmWorkManage.java
 - FrmWorkerRecord.java

Code Editor (FrmMotorcycles.java):

```

  ...
  DBCollection collection;
  BasicDBObject document = new BasicDBObject();
  MongoClient mongo = createConnection();

  /**
   * Creates new form GUIMotorcycles
   */
  public FrmMotorcycles() {
    initComponents();
    this.setLocationRelativeTo(null);
  }

  /**
   * This method is called from within the const
  
```

-**Single responsibility** (The connection to the database must be made by a single class, not two.)

The screenshot shows a Java code editor with a package structure on the left.

Package Structure:

- ec.edu.espe.mechanic.utils
 - Connection.java
 - DBManager.java
 - FileManager.java
 - MySQLDBManager.java
 - OperationMongoDB.java
 - Oracle.java
 - Persistence.java
 - SQLDBManager.java

-Open / Closed Principle

Change the name of the array related to the project and the records it manages

```

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
  JSONArray jrr = new JSONArray();
  Object ob = null;
  JSONParser Jp = new JSONParser();
  //fetch file--
  
```

- The graphical view interface has 3 additional buttons that do not explain the operation (methods declared without using).

CUSTOMER PERSONAL INFORMATION

Name:	<input type="text" value="Mateo"/>	Row:	<input type="text" value="1"/>
Last Name:	<input type="text" value="Landazuri"/>	Column:	<input type="text" value="1"/>
Telephone Number:	<input type="text" value="123465678"/>	New Data:	<input type="text" value="1"/>
Email:	<input type="text" value="aaaaaaa.com"/>		
ID:	<input type="text" value="001"/>		

Name	Last Name	Telephone N...	Email	ID

TEAM 3:

Paul Saltos
 Javier Paucar
 Cristian Arroba
 Melissa Gómez
 José Guzmán

```

public class Person {

    private String name;
    private String lastname;
    private String telephonenumber;
    private String Email;
    private String ID;
  
```

-variables must be written with lowercase

Start Page X Accesories.java X

Source History

```

1 package ec.edu.espe.mechanic.model;
2
3 /**
4 * @author Sigma Programmers
5 */
6 public class Accesories extends Product {
7
8     public Accesories(String name, int quanty, int price) {
9         super(name, quanty, price);
10    }
11
12 }
  
```

- Classes must be singular

```

55     */
56     public void setPrice(int price) {
57         this.price = price;
58     }
59
60     /**
61      * name ;
62      * quanty;
63     >>>> e4b3f82c2e54cd426697e6dle44a137df95afedb
64     */
65 }
66

```

- uncommented code

- btnReturn : JButton
- discount : JTextField
- jButton1 : JButton
- jButton2 : JButton
- jLabel1 : JLabel
- jLabel10 : JLabel
- jLabel11 : JLabel
- jLabel12 : JLabel
- jLabel13 : JLabel
- jLabel14 : JLabel
- jLabel2 : JLabel

-Must have specific names

TEAM 4:

Angel Guaman
 Sebastián Caisatoa
 Sebastián Tayo
 Antony Morales
 José Sánchez

Open closed principle

Declaration of variable

```

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    JSONArray jrr = new JSONArray();
    Object ob = null;
    JSONParser Jp = new JSONParser();
    //fetch file-->
}

```

The second variable starts with mayus, and not in lower Camel case.

Parameterized constructor is used to initialize object - constructor P

```
private int year;
private String Registration;
private String Plate;
private String trademark;
private String model;
private float mileage;

public Vehicle(int year, String Regis
```

Registration, String plate, String trademark, String model, float mileage)

Unnecessary comments

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    | Look and feel setting code (optional)
    //</editor-fold>
    //</editor-fold>
```

/* Create and display the form */

java.awt.EventQueue.invokeLater(new Runnable {

Unprogrammed code

```
private void txtLastNameActionPerformed(java.awt.event.ActionEvent evt) {
    |     // TODO add your handling code here:
}

private void txtIDActionPerformed(java.awt.event.ActionEvent evt) {
    |     // TODO add your handling code here:
}

private void txtNameActionPerformed(java.awt.event.ActionEvent evt) {
    |     // TODO add your handling code here:
}
```

It calls the mongo db Connection in a button.

```
private void btnShowActionPerformed(java.awt.event.ActionEvent evt) {
    DBCursor cursor = null;
    DB db = mongo.getDB("Person");
    DBCollection dbCollection = db.getCollection("Worker");
    try {
        cursor = dbCollection.find();
        String[] columnNames = {"Name", "LastName", "Phone", "Code"};
        DefaultTableModel modelTable = new DefaultTableModel(columnNames, 0);
        while (cursor.hasNext()) {
           DBObject obj = cursor.next();
            String name = (String) obj.get("Name");
            String lastName = (String) obj.get("LastName");
            String phone = (String) obj.get("Phone");
            String Code = (String) obj.get("Code");
            modelTable.addRow(new Object[]{name, lastName, phone, Code});
        }
        tblSpeakers.setModel(modelTable);
        cursor.close();
    } catch (Exception ex) {
        System.out.println("Error printing tables");
    }
}
```

Interface segregation principle

table don't have information of form

Employee Registration

Name	<input type="text"/>	Row	<input type="text"/>
Last Name	<input type="text"/>	Column	<input type="text"/>
Phone	<input type="text"/>	fact	<input type="text"/>
Code	<input type="text"/>		

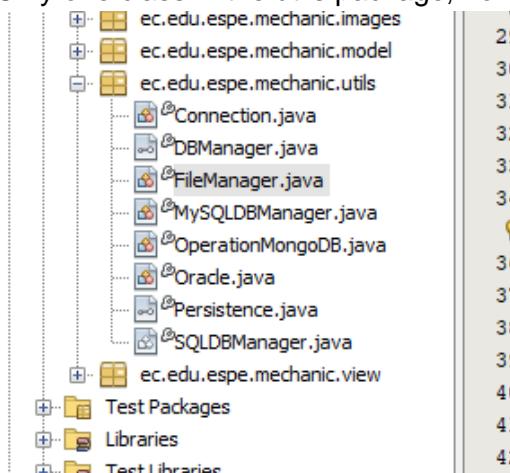
Name	Last Name	Phone	code

BILL

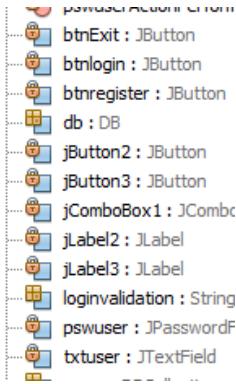
Name	<input type="text"/>	Vehicle	<input type="text"/>
Lastname	<input type="text"/>		
Repair	<input type="checkbox"/>		
Washer	<input type="checkbox"/>		
Other	<input type="text"/>		
Other			
Discount	<input type="text"/>		
TOTAL	<input type="text"/>		

Single Responsibility

Only one class in the utils package, no more than two

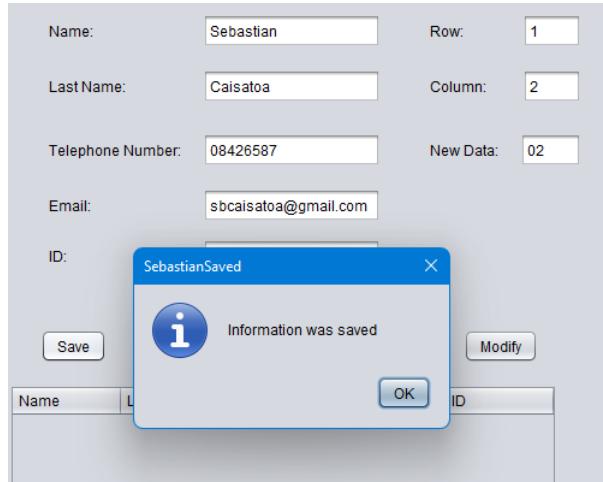


Undeclared Buttons



Segregation principle

Program clients must know what data is entered and what methods are actually to be used



TEAM 5:

Alvarez Michelle
 Correa Kerly
 Eivar Jaime
 Garcia Mayerly
 Teca Camila
 Teran Melanie

```

BasicDBObject document = new BasicDBObject();

public DDBObject addVehicle(int Name, String LastName, String Phone, String Code) {

```

The name is declared as a int, it should be String - Bad quality code

Mechanic - Apache NetBeans IDE 12.5

```

package ec.edu.espe.mechanic.controller;
import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;

/**
 * @author Sigma Programmers
 */
public class ValueProduct {
    BasicDBObject document = new BasicDBObject();

    public DBObject addProduct(String name, int quanty, int price) {
        document.put("Name", name);
        document.put("Quanty", quanty);
        document.put("Price", price);
        return null;
    }
}

```

Mechanic - Apache NetBeans IDE 12.5

```

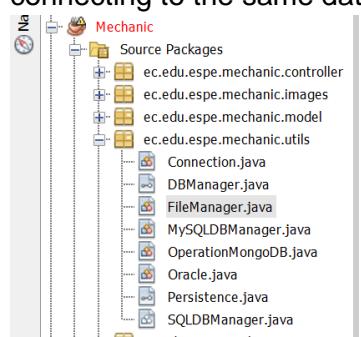
package ec.edu.espe.mechanic.controller;
import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;

/**
 * @author Sigma Programmers
 */
public class ValueVehicle {
    BasicDBObject document = new BasicDBObject();

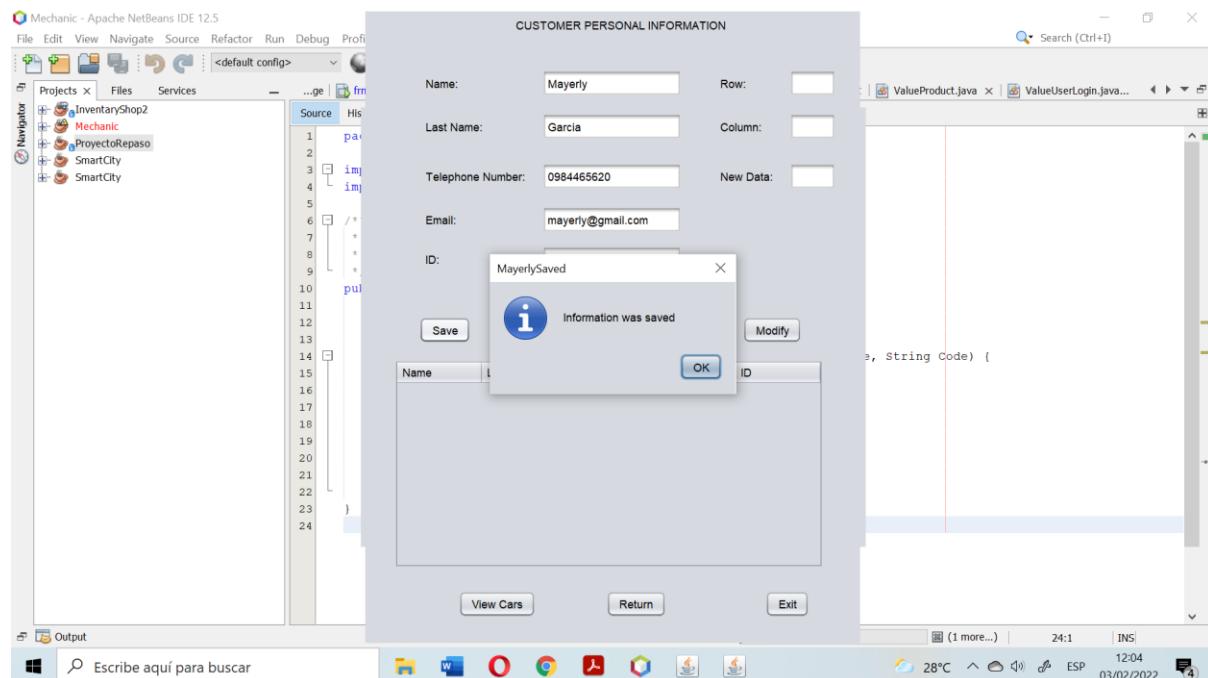
    public DBObject addVehicle(int year, String Registration, String Plate, String trademark, String model, float mileage) {
        document.put("Year", year);
        document.put("Registration", Registration);
        document.put("Plate", Plate);
        document.put("Trademark", trademark);
        document.put("Model", model);
        document.put("Mileage", mileage);
        return null;
    }
}

```

In the utils package there should be one class to the database, but no more than two classes connecting to the same database. (**Single Responsibility**)



The Interface segregation principle is broken because the clients of the program should know what data is entered and which methods should actually be used.
Added or saved data is not displayed in the table



Single Responsibility Principle broken because has not relation with the data of the class:

A screenshot of a web-based application titled "Employee Registration". The form contains four input fields: "Name", "Last Name", "Phone", and "Code", each with an associated text input box. Below the inputs are four buttons: "Insert", "Delete", "Delete All", and "Modify". At the bottom of the form is a table with columns "Name", "Last Name", "Phone", and "Code", and a "Return" button. A red circle highlights a section of the form containing the labels "Row", "Column", and "fact", which are not logically related to the employee registration data.

Single Responsibility Principle broken because the "add vehicle" object is not related to the "EmployeeController" class.

```

1 package ec.edu.espe.mechanic.controller;
2
3 import com.mongodb.BasicDBObject;
4 import com.mongodb.DBObject;
5
6 /**
7 *
8 * @author Sigma Programmers
9 */
0 public class EmployeeController {
1
2     BasicDBObject document = new BasicDBObject();
3
4     public DBObject addVehicle(int Name, String LastName, String Phone, String Code) {
5
6         document.put("Name", Name);
7         document.put("LastName", LastName);
8         document.put("Phone", Phone);
9         document.put("Code", Code);
0
1         return null;
2     }
3 }

```

Bad declaration of the person Method of the Person class, missing to declare id

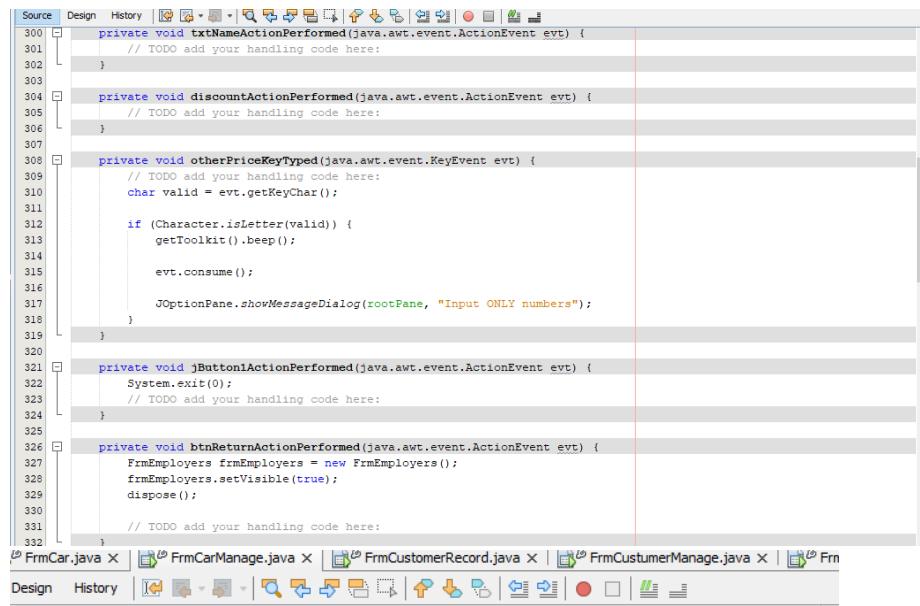
```

26
27     public Person(int id, String name, String lastName, String plate) {
28         this.id = id;
29         this.name = name;
30         this.lastName = lastName;
31         this.plate = plate;
32     }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

Person person = new Person(name,lastName,plate);

Bad names for button functions, FrmBill does not perform any function and comments not deleted



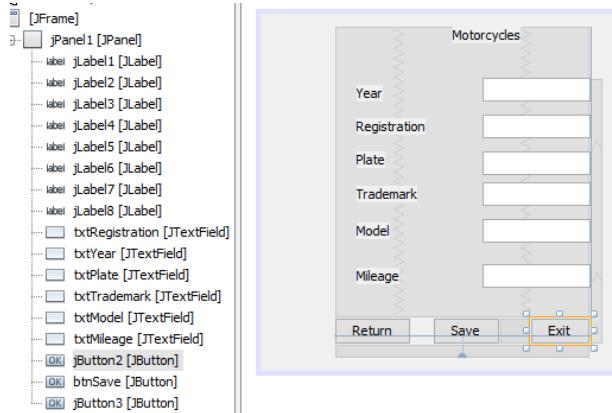
```

300 private void txtNameActionPerformed(java.awt.event.ActionEvent evt) {
301     // TODO add your handling code here:
302 }
303
304 private void discountActionPerformed(java.awt.event.ActionEvent evt) {
305     // TODO add your handling code here:
306 }
307
308 private void otherPriceKeyTyped(java.awt.event.KeyEvent evt) {
309     // TODO add your handling code here:
310     char valid = evt.getKeyChar();
311
312     if (Character.isLetter(valid)) {
313         Toolkit.getDefaultToolkit().beep();
314
315         evt.consume();
316
317         JOptionPane.showMessageDialog(rootPane, "Input ONLY numbers");
318     }
319 }
320
321 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
322     System.exit(0);
323     // TODO add your handling code here:
324 }
325
326 private void btnReturnActionPerformed(java.awt.event.ActionEvent evt) {
327     FrmEmployers frmEmployers = new FrmEmployers();
328     frmEmployers.setVisible(true);
329     dispose();
330
331     // TODO add your handling code here:
332 }

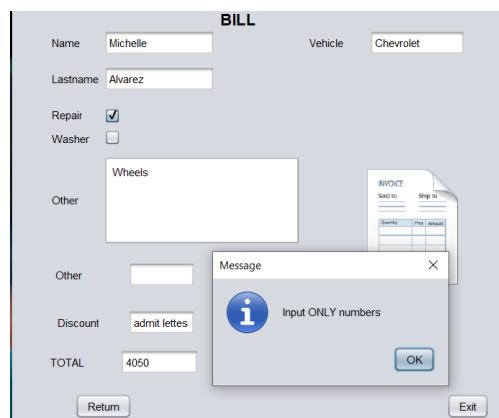
```

The code is part of a Java application named 'FrmCustomerRecord.java'. It includes imports for `java.awt.event` and `JOptionPane`. The file is part of a project with other files like 'FrmCar.java', 'FrmCarManage.java', 'FrmCustomerRecord.java', 'FrmCustomerManage.java', and 'FrmBill.java'. The code contains several methods with bad names: `txtNameActionPerformed`, `discountActionPerformed`, `otherPriceKeyTyped`, `jButton1ActionPerformed`, and `btnReturnActionPerformed`. The code is mostly empty with TODO comments.

Unnecessary commented code (FrmCarManage)



Bad names for Return and Exit buttons (FrmMotorcycles)

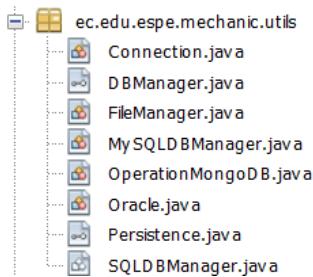


The FrmBill does not follow the **Liskov Substitution Principle** because it does not represent a window or public methods into the program. There are two fields call "Other" and in one of them it is not clear which is the information that you have to right. The field discount and TOTAL are not clear. They must be calculated into the program.

The image shows the Java code for the FrmMotorcycles class. The code includes several commented-out sections of code, indicated by the `/* */` and `*/ */` markers. One such section is at the top:

```
34     * This method is called from within the constructor to initialize the form.
35     * WARNING: Do NOT modify this code. The content of this method is always
36     * regenerated by the Form Editor.
37     */
38     @SuppressWarnings("unchecked")
39     // Generated Code
208     private void txtMileageActionPerformed(java.awt.event.ActionEvent evt) {
210         // TODO add your handling code here:
211     }
212     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
214         System.exit(0);
215         // TODO add your handling code here:
216     }
217     private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
219
220         FrmEmployers frmEmployers = new FrmEmployers();
221         frmEmployers.setVisible(true);
222         dispose();
223
224         // TODO add your handling code here:
225     }
226 }
```

In the FrmMotorcycle there are some buttons without name so it is difficult to understand how the program works. **Unclean code.**



Each class should be responsible for only one part of the system's functionality. We can see there are many classes that allow the connection to the database, so it is not clear which class do a specific job. So in this part of the program does not follow the **Single Responsibility Principle**.

TEAM 6:

Alex Andrango

Anderson Almache

Dylan Asumaza

Cristhian Altamirano

Andrea Tapia

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
    // TODO add your handling code here:
}
```

Change button variable name Exit

```
public class Customer extends Person {

    public Customer(String name, String lastname, String telephoneNumber, String Email, String ID) {
        super(name, lastname, telephoneNumber, Email, ID);
    }

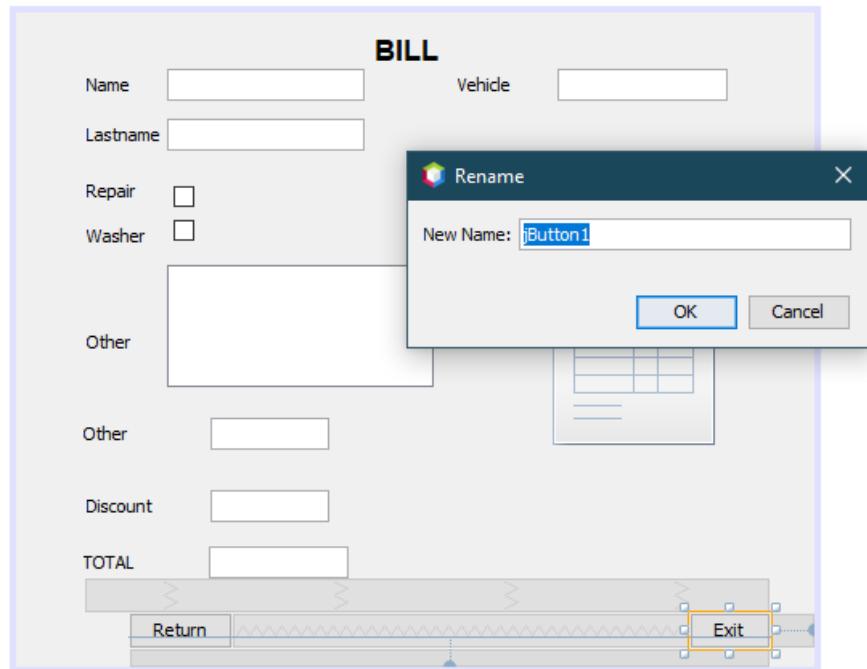
}
```

Camel Case

```
public class Launderer extends Washer {

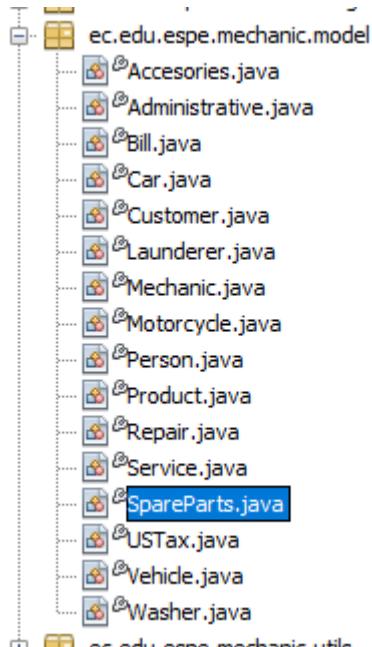
    public Launderer(String specialits, float waitTime) {
        super(specialits, waitTime);
    }

}
```



Name of the

buttons



some classes are written in plural

Default comments are not deleted

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.mechanic.utils;

import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.MongoClient;

/**
 *
 * @author SigmaProgramers
 */
public class OperationMongoDB {

```

TEAM 7:

Alan Andrade

Katherin Bravo

Darling Cruz

Jhon Guitarra

Alejandro de la Cruz

```

private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    String dataToUpdate = "Do you want to update " + txtDataToUpdate.getText() + "?";
    if (txtDataToUpdate.getText().isEmpty() || txtNewData.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "FILL ALL THE FIELDS");
    } else {
        int selection = JOptionPane.showConfirmDialog(null, dataToUpdate, "Speaker Updating",
            JOptionPane.YES_NO_CANCEL_OPTION);
        switch (selection) {
            case 0:
                JOptionPane.showMessageDialog(null, "Information was updating", txtDataToUpdate.getText() + "Updated",
                    JOptionPane.INFORMATION_MESSAGE);
                updateCars(mongo, "Vehicles", "Cars", txtDataToUpdate.getText(), txtNewData.getText(), cmbField.getSelectedItem().toString());
                txtDataToUpdate.setText("");
                txtNewData.setText("");
                cmbField.setSelectedIndex(0);
                break;
            case 1:
                JOptionPane.showMessageDialog(null, "Information was NOT saved", txtDataToDelete.getText() + "NOT deleted",
                    JOptionPane.INFORMATION_MESSAGE);
                txtDataToDelete.setText("");
                txtNewData.setText("");
                cmbField.setSelectedIndex(0);
                break;
            default:
                JOptionPane.showMessageDialog(null, "Action was cancelled", txtDataToDelete.getText() + "Cancelled",
                    JOptionPane.INFORMATION_MESSAGE);
                break;
        }
    }
}

private void btnReturnActionPerformed(java.awt.event.ActionEvent evt) {

```

the button contains code

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

```

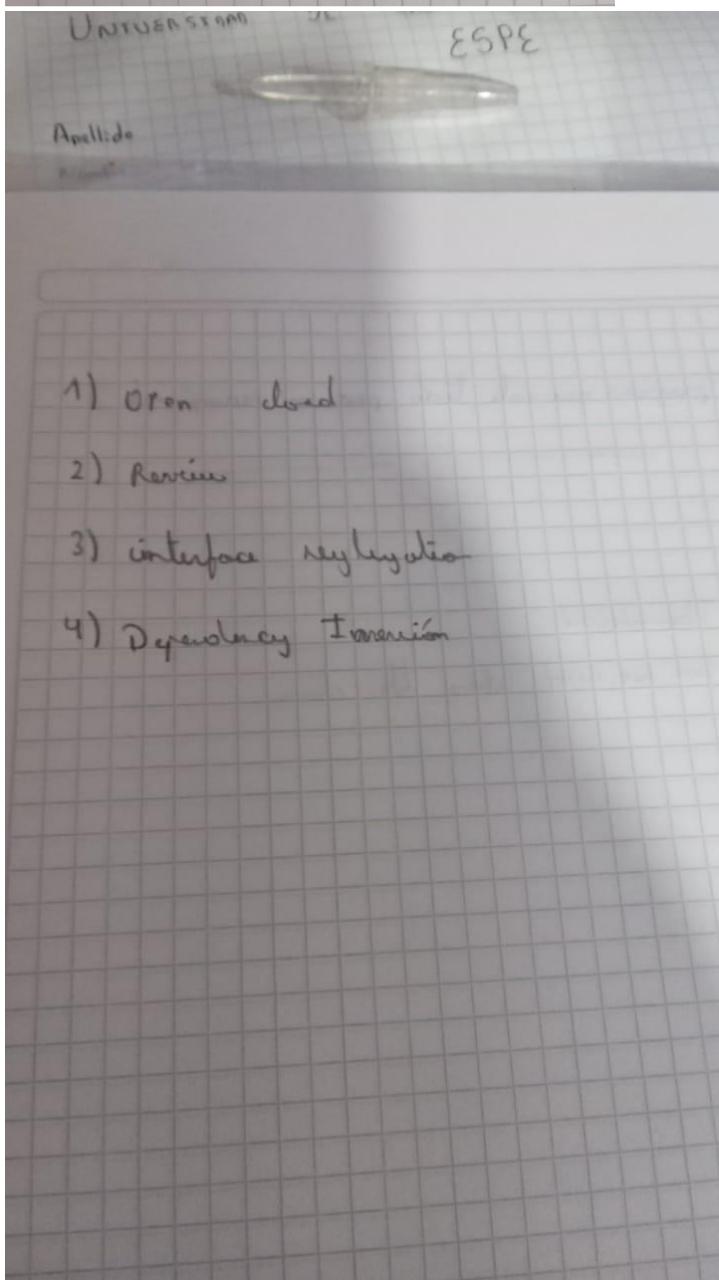
QUIZZES about SOLID Principles

ANSWERS:

1. Open /closed principle (Open/close)
2. Liskov Substitution
3. Interface Segregation
4. Dependency Inversion

5. Single Responsibility

- 1.- Open / Close .
- 2.- True
- 3.- Inter face Segregation
- 4.- Dependency
- 5.- Segregated Interface



Mayerly García

OPP 7A90

- Open principle
- Close principle.

- | | | |
|-----------------------|---|------|
| { Data members | } | True |
| { Method declarations | | |
| { Method definitions | | |

- Should be forced to depend on methods that it does not use (interface segregation principle)

- Dependency Inversion

- Interface Segregation

Name Altamirano Cristhian

NRC: 7490

What is

- 1) Open , closed, principle.
- 2) Review
- 3) Interface segregation principle
- 4) Dependency , Abstract principle
- 5) Single, Responsability, principle.

Día MES Año
Alejandro De la Cruz 2490

"El deseo de escribir aumenta a medida que se escribe"

- 1) Close / Open Jdk
- 2) Reuse
- 3) Interface Segregation principle
- 4) Dependency Inversion
- 5) Single , Responsibility y principio

Name: Pablo Bustillos

1º Open Clase principal

2º True

3º Interface segregation principle

4º Dependency inversion

5º Interface segregation principle

→ Sebastián Caísatoca

1. Open / Close Principle

2. True

3. Interface Segregation

4. Dependency Inversion

5. Interface Segregation

Name = Joel Zeus

Course : OOP - 7490

Date = 03 / 02 / 2022

1) Open / closed principle

2) Review

3) Interface segregation

4) Dependency abstract principle.

5) Single responsibility

Name: Salma Villegas

1. Open - Close Principle

2. True

3. Liskov Substitution Principle.

4. Dependency Inversion Principle

5. Single Responsibility Principle.

MELISSA GOMEZ

1.- OPEN - CLOSE

2.- LISKOV

3.- DEPENDENCY INVERSION

4.- INTERFACE SEGREGATION

5.- SINGLE RESPONSIBILITY

Michelle Alvarez

OOP 7490

1. Open / Close Principle
2. True
3. Interface Segregation
4. Dependency Inversion
5. Interface Segregation

Name: Leanel Mantuano

- 1) open close principle
- 2) True
- 3) Interface segregation
- 4) Dependency Inversion
- 5) Interface segregation

Andrango Alex

OOP 7490

1. Open / close principle
2. Liskov segregation principle
3. Interface segregation principle
4. Dependency inversion principle
5. Single responsibility principle

Name: Andy Josue Calderón Merchan
DDP 7490

1. Open / Clase Principa l
2. True
3. Interface Segregation
4. Dependency Inversion
5. Interface Segregation

Katherin Bravo

1. Open - Close Principle
2. - True
3. - Interface Segregation
4. - Dependency Inversion
5. - Interface Segregation

Melanie Terán

OOP

NAC: 7490

- 1. Open / Close principle
- 2. Liskov Substitution Principle
- 3. Interface Segregation principle
- 4. Dependency inversion principle
- 5. Single responsibility principle

Andrea Tapia 7490.

1. Open/Closed principle
2. Review
3. Interface Segregation principle
4. Dependency Abstract principle
5. Single responsibility principle

Quiz about SOLID Principles

Name: Andrade Alan

NRC: 7490

- 1: Close - Open - Principle
- 2: Review
- 3: Interface Segregation principle.
- 4: Dependency Abstract principle
- 5: Single Responsibility

Darling Cruz

- 1 Open - Close Principle
- 2 True
- 3 You have segregation
- 4 Dependency
- 5 aggregation interface

Jalme Eiow

OOP 7490

- 1.) Open close principal
- 2.) True.
- 3.) Interface Segregation principle
- 4.) Dependency Inversion
- 5.) Interface Segregation principle
- 6.)

RICHAR MAISINCHO

Open - close pricipal

True

Interface segregation

Dependencia inversion

Interface segregation

Mateo Landaizuri.

7490

DOP.

1. Open / close principle

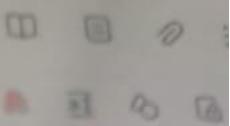
2. True

3. Interface segregation

4. Dependency Inversion principle

5. Interface Segregation

Quiz



Quiz
Name: José Francisco Sánchez M. Shiquero

1 Open Close

2 Review

3 Abstraction Core Ideas

4 Refinement ↗

5



Name: Daniel Lincango

NRC: 7490

- (1) Open/Closed
- (2) Liskon
- (3) Interface
- (4) Dependence, Abstract principle
- (5) Single Responsibility

DD MM AA

N. Benjamin Cadena

9490 - OOP,

Thursday, February 3rd 2022

1. -

Open, closed, principle

2. -

Review

3. -

Interface segregation

4. -

Dependency Inversion

5. -

Single Responsibility principle

Normal

NAME: JAVIER PAUCAR

1. OPEN / CLOSE PRINCIPLE

2. REVIEW

3. INTERFACE SEGREGATION

4. DEPENDENCY , ABSTRACT PRINCIPLE

5.

Name: Diego Sebastián Palacios Condó

1 Open / Close principal

2. Liskov segregation

3. Interface segregation

4. Dependency inversion

5. Interface segregation

Name: Anderson Almache

NRC: 7490

Answers:

1. Open...
2. Yes
3. Interface Segregation
4. Dependency, Abstract
5. Single

Mateo Maldonado

- What is the principles should be ready for functionality

Single responsibility principle

- When we have an inheritance the father with ever the children need. parent classes need it.

Dependency inversion

- No client should be post to depend methods that don't use

Interface segregation principle

- I have to organize the system in two

Dependency inversion principle

- This principle says that component do the must do and no other action

Interface segregation principle

Dylan Asumura

7490 OPP

1. Open / Close principle
2. Review
3. Interface Segregation principle
4. Dependency Abstract principle
5. Single responsibility principle

Kerly Correa

1. Abstract, Open / Close
2. Review (True)
3. Interfaces
4. Dependency
5. Single, Segregation

Name: Angel Guaman

NQF: 74.90

- 1) open closed
- 2) Review
- 3) interface segregation
- 4) dependency inversion
- 5)

Eayo Sebastian

1. What is the principle that the software entities should be able to extend but not to modify?

Inheritance / Open / closed Principle

2. -

3. -

Interface segregation Principle

4. The derived layers should not depend

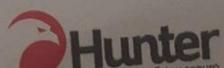
Dependency inversion Principle

5. A component should only do what it has to.

Interface segregation

1. Open / Closed Principle
2. Single Responsibility Principle
3. Interface Segregation Principle
4. Dependence inversion Principle
5. Liskov Substitution

Camilo Tecc



- 1) Open / close
- 2) Liskov
- 3) Interface
- 4) Dependence , Abstract principle
- 5) Single responsibility

CS Escaneado con CamScanner

Alexander Ruano

- Name: Anthony Morales
- Class name: OOP
- 1) Open / close principle
 - 2) Yes or True
 - 3) Interface segregation principle
 - 4) Dependency principle
 - 5) Single responsibility