

WS32 - SOFTWARE PRINCIPLES GROUPS

DATE: 03th february 2021

TEAM 1:

Salma Villegas

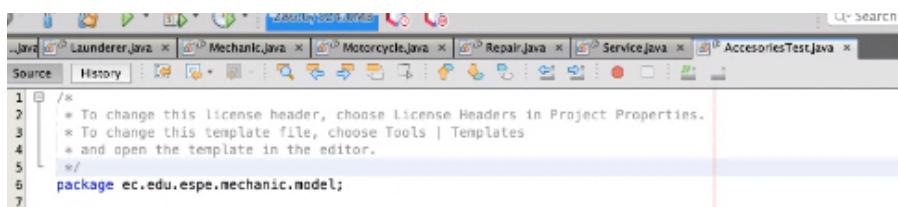
Alexander Ruano

Daniel Lincango

Mateo Maldonado

Leandro Quinga

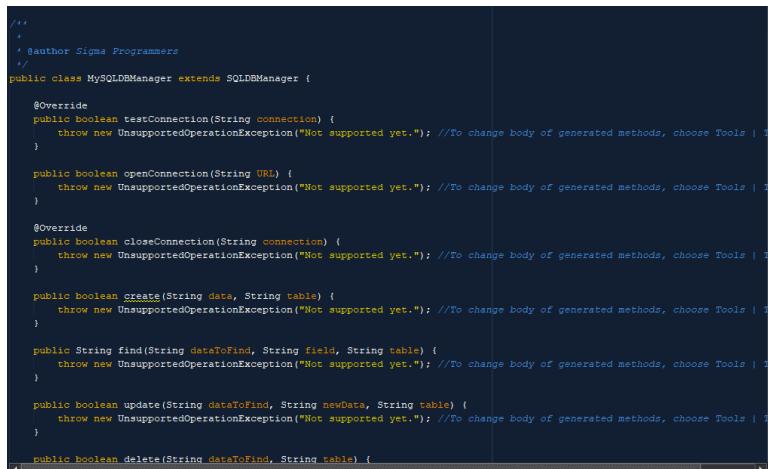
Joel Zeas



The screenshot shows a Java code editor with several tabs at the top: LaundryJava, Mechanic.java, Motorcycle.java, RepairJava, Service.java, and AccessoriesTest.java. The main window displays the following 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.
 */
package ec.edu.espe.mechanic.model;
```

Innecesary comments, unclean code



The screenshot shows a Java code editor with the following code:

```
/*
 * Author Sigma Programmers
 */
public class MySQLDBManager extends SQLDBManager {

    @Override
    public boolean testConnection(String connection) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

    public boolean openConnection(String URL) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

    @Override
    public boolean closeConnection(String connection) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

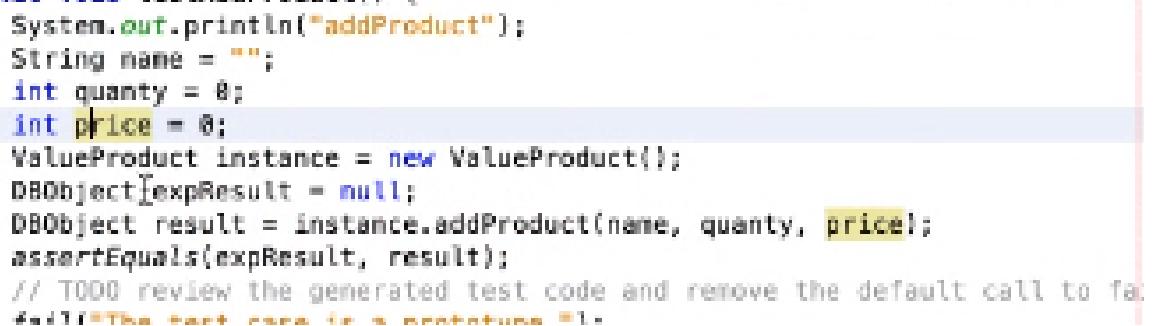
    public boolean create(String data, String table) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

    public String find(String dataToFind, String field, String table) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

    public boolean update(String dataToFind, String newData, String table) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }

    public boolean delete(String dataToFind, String table) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates
    }
}
```

Unclean Code



The screenshot shows a Java code editor with the following code:

```
System.out.println("addProduct");
String name = "";
int quantity = 0;
int price = 0;
ValueProduct instance = new ValueProduct();
DBObject expResult = null;
DBObject result = instance.addProduct(name, quantity, price);
assertEquals(expResult, result);
// TODO review the generated test code and remove the default call to fail
// fail();  
The part para la ejecución
```

```

/*
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;
    }
}

```

do not use the format Camelcase

```

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

```

Refactor jButton

```

boolean create(String Data, String table, BasicDBObject document);

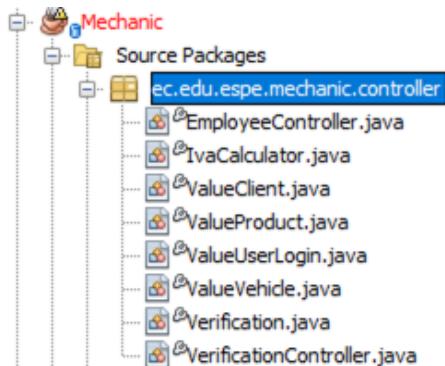
String find(String DataBase, String dataToFind, String field, String table);

boolean update(String DataBase, String dataToFind, String newData, String field, S.

```

Don't use camelcase

- Very redundant class names, difficult to differentiate



- **Single Responsibility Principles:** there is no relationship between the “document” variable and the “addVehicle” class. It is also a very general term.

```

BasicDBObject document = new BasicDBObject();

public DBObject addVehicle(int Name, String LastName, String Phone, String Code)

    document.put("Name", Name);
    document.put("LastName", LastName);
    document.put("Phone", Phone);
    document.put("Code", Code);

    return null;
}

```

- Use an else variable that replaces the return and returns an actual data

```

public class ValueClient {

    public boolean create(Customer customer) {
        boolean created = false;
        String personData;

        Persistence persistence;
        persistence = new FileManager();

        if (persistence.create(customer.toString(), "Data.txt")) {
            JOptionPane.showMessageDialog(null, customer + " was saved");
        }
        return created;
    }
}

```

- Use lower case for variables

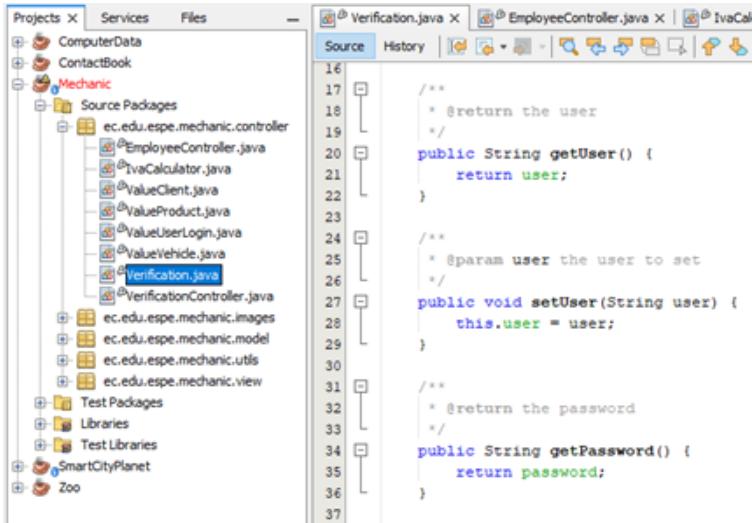
```

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);
}

```

- Getters and setters must be programmed in the ".model" package, no in the ".controller" package



- **Open/Closed Principle:** Very general variable and method names, also the user validation programming is usually done in the .view package and it must be open to extend it

```

public boolean login(Verification verification, String user, String password) {

    boolean logged = false;
    String readLine;
    Persistence persistence;
    persistence = new FileManager();
    readLine = persistence.read("", "Users.json");

    Gson gson = new Gson();

    verification = gson.fromJson(readLine, Verification.class);

    if (user.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(null, "FILL ALL THE FIELDS");
    } else if (user.equals(verification.getUser()) == false && password.equals(verification.getPassword()) == false) {
        ...
    }
}

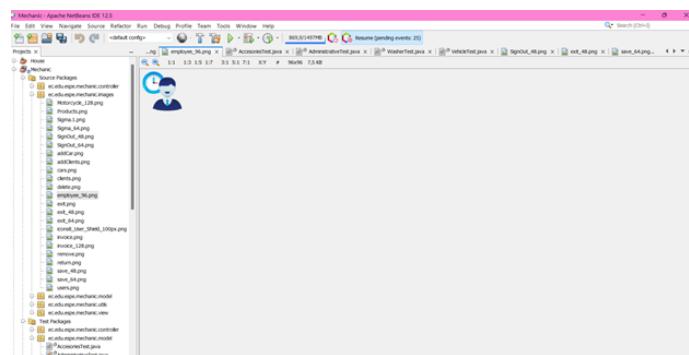
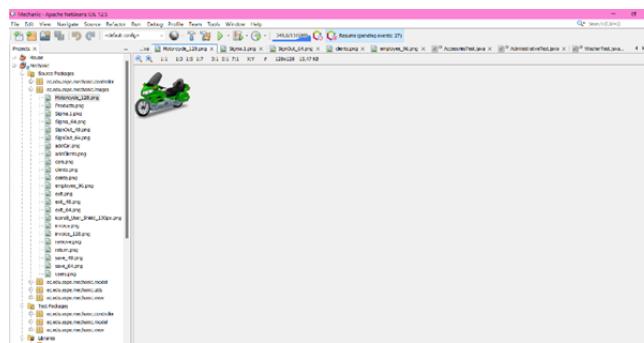
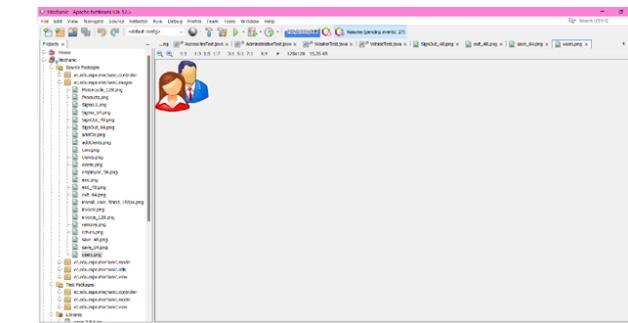
```

Source Packages

Package: ec.edu.espe.mechanic.images

Nothing to mention, each class has only one image.

Used Principles: Single Responsibility Principle (each class have only one image)

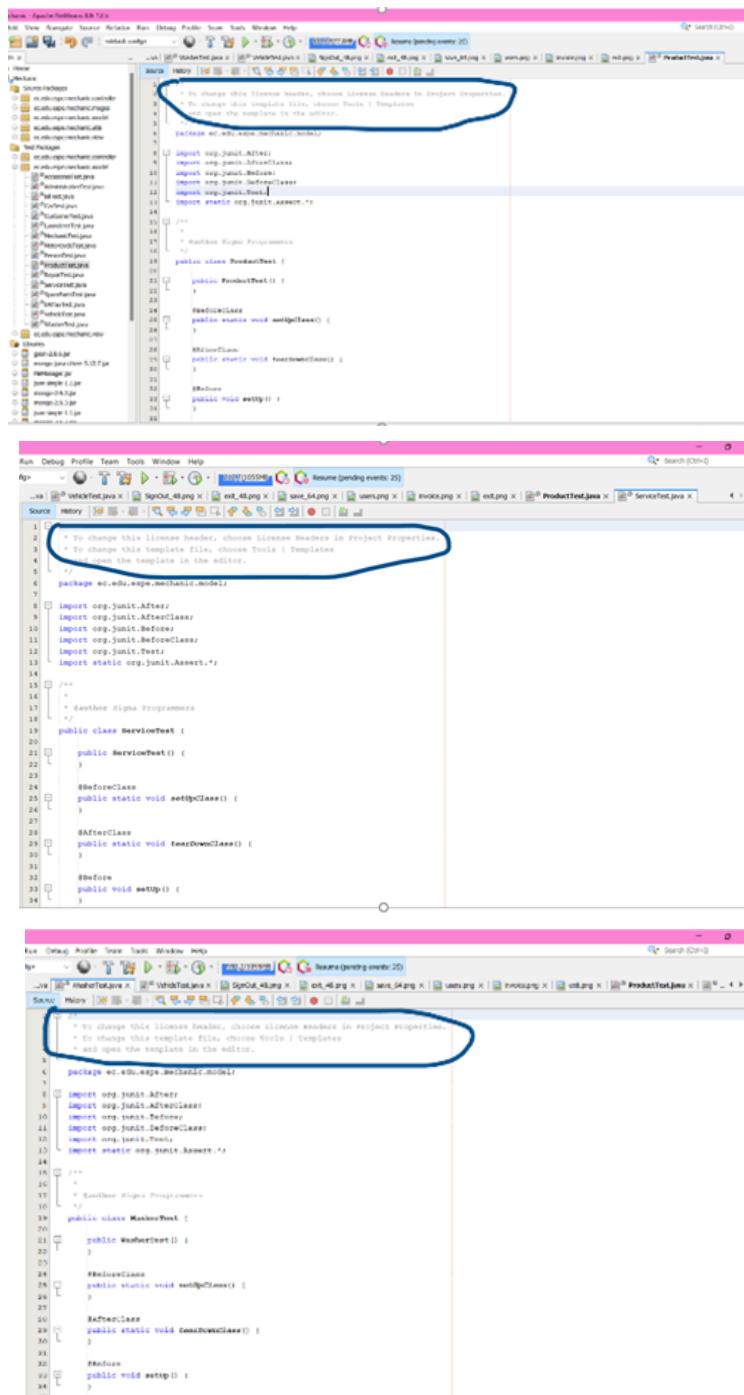


Test Packages

Package: ec.edu.espe.mechanic.model

Unnecessary comments, unclean code,

Used Principles: Single Responsibility Principle (This package is being used for testing only, so each class has a unique reason)



```
/*
 * 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.model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 * Author: Sigma Programmers
 */
public class ProductTest {

    /**
     * @BeforeClass
     */
    public static void setUpClass() {
        ...
    }

    /**
     * @AfterClass
     */
    public static void tearDownClass() {
        ...
    }

    /**
     * @Before
     */
    public void setUp() {
        ...
    }

    /**
     * @After
     */
    public void tearDown() {
        ...
    }
}
```



```
/*
 * 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.model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 * Author: Sigma Programmers
 */
public class ServiceTest {

    /**
     * @BeforeClass
     */
    public static void setUpClass() {
        ...
    }

    /**
     * @AfterClass
     */
    public static void tearDownClass() {
        ...
    }

    /**
     * @Before
     */
    public void setUp() {
        ...
    }

    /**
     * @After
     */
    public void tearDown() {
        ...
    }
}
```



```
/*
 * 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.model;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 * Author: Sigma Programmers
 */
public class WasherTest {

    /**
     * @BeforeClass
     */
    public static void setUpClass() {
        ...
    }

    /**
     * @AfterClass
     */
    public static void tearDownClass() {
        ...
    }

    /**
     * @Before
     */
    public void setUp() {
        ...
    }

    /**
     * @After
     */
    public void tearDown() {
        ...
    }
}
```

```
public void testHabilitacion() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.habilitacion();
    assertEquals("Medico habilitado", resultado);
    System.out.println("Medico habilitado");
}

/* Test of deshabilitacion method, of class Medico. */
@Test
public void testDeshabilitacion() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.deshabilitacion();
    assertEquals("Medico inhabilitado", resultado);
    System.out.println("Medico inhabilitado");
}

/* Test of getNombre method, of class Medico. */
@Test
public void testGetNombre() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.getNombre();
    assertEquals("Medico", resultado);
    System.out.println("Medico");
}
```

```
public void testHabilitacion() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.habilitacion();
    assertEquals("Medico habilitado", resultado);
    System.out.println("Medico habilitado");
}

/* Test of deshabilitacion method, of class Medico. */
@Test
public void testDeshabilitacion() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.deshabilitacion();
    assertEquals("Medico inhabilitado", resultado);
    System.out.println("Medico inhabilitado");
}

/* Test of getNombre method, of class Medico. */
@Test
public void testGetNombre() {
    System.out.println("Testing");
    Medico instance = null;
    String resultado = instance.getNombre();
    assertEquals("Medico", resultado);
    System.out.println("Medico");
}
```

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 quanthy;
63     >>>>> e4b3f82c2e54cd426697e6d1e44a137df95afedb
64     /**
65 }
66

```

- Comments not deleted

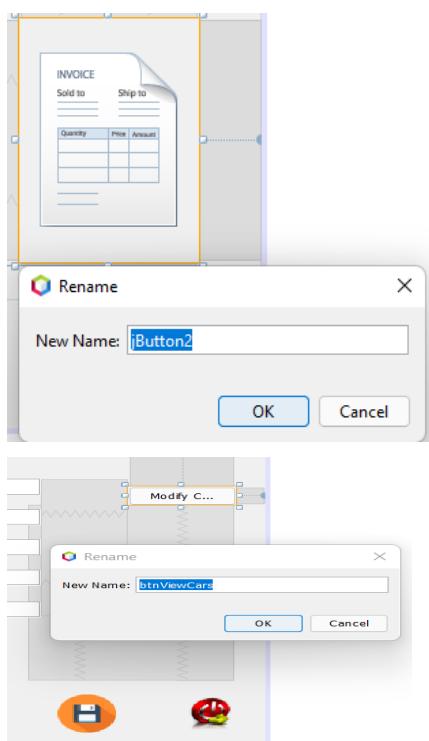
```

/*
 * 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

The screenshot shows a Java code editor with a file tree on the left. The file tree under 'ec.edu.espe.mechanic.model' includes: Accesories.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, and SQLDBManager.java. The code editor has several lines of code highlighted in yellow, indicating potential issues or warnings.

```

181     DB db = mongo.getDB(dataBase);
182     DBCollection dbCollection = db.getCollection(collection);
183     BasicDBObject searchedName = new BasicDBObject();
184     BasicDBObject updateData = new BasicDBObject();
try {
    ...
    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 file tree under 'ec.edu.espe.mechanic.utils' containing: Connection.java, DBManager.java, FileManager.java, MySQLDBManager.java, OperationMongoDB.java, Oracle.java, Persistence.java, and SQLDBManager.java.

-Open / Closed Principe

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

```

private void btnDeleteActionPerfomed(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="aaaaaaaa.com"/>		
ID:	<input type="text" value="001"/>		

<input type="button" value="Save"/>	<input type="button" value="Delete"/>	<input type="button" value="Delete All"/>	<input type="button" value="Modify"/>
-------------------------------------	---------------------------------------	---	---------------------------------------

Name	Last Name	Telephone N...	Email	ID

<input type="button" value="View Cars"/>	<input type="button" value="Return"/>	<input type="button" value="Exit"/>
--	---------------------------------------	-------------------------------------

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

The screenshot shows a Java code editor with the file 'Accesories.java' open. The code defines a class 'Accesories' that extends 'Product'. It includes a constructor that takes 'name', 'quanity', and 'price' as parameters and calls the super constructor.

```
1 package ec.edu.espe.mechanic.model;
2
3 /**
4 * 
5 * @author Sigma Programmers
6 */
7 public class Accesories extends Product {
8
9     public Accesories(String name, int quanity, int price) {
10        super(name, quanity, price);
11    }
12
13 }
```

- Classes must be singular

```
55     /**
56      * 
57      * @author Sigma Programmers
58      */
59
60      /**
61      * String name ;
62      * int quanity;
63      * >>>> e4b3f82c2e54cd426697e6dle44a137df95afedb
64      */
65
66 }
```

- uncommented code

The screenshot shows a list of Java variables representing UI components. Each variable is preceded by a lock icon, indicating they are final or constant.

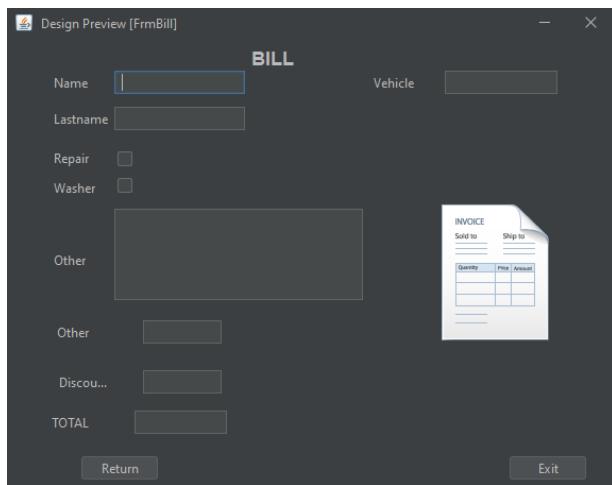
- btnReturn : JButton
- discount : JTextField
- jButton1 : JButton
- jButton2 : JButton
- jLabel1 : JLabel
- jLabel10 : JLabel
- jLabel11 : JLabel
- jLabel12 : JLabel
- jLabel13 : JLabel
- jLabel14 : JLabel
- jLabel2 : JLabel

-Must have specific names

Single responsibility: The user or client must know the data to be added



Interface segregation:



TEAM 4:

Angel Guaman
 Sebastián Caisatoa
 Sebastián Tayo
 Antony Morales
 José Sánchez

Open closed principle

Declaration of variable

```
private void btnDeleteActionPerformed(java
    JSONArray jrr = new JSONArray();
    JSON
    Object ob = null;
    JSONParser Jp = new JSONParser();
    JSON //fetch file--
```

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

```
Para unir los datos de la clase JSON con el modelo de la clase Vehicle escribe este P
private int year;
private String Registration;
private String Plate;
private String trademark;
private String model;
private float mileage;
```

Unnecessary comments

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"/>										
<input type="button" value="Insert"/>		<input type="button" value="Delete"/>	<input type="button" value="Delete All"/>								
<table border="1"> <thead> <tr> <th>Name</th> <th>Last Name</th> <th>Phone</th> <th>code</th> </tr> </thead> <tbody> <tr><td colspan="4"> </td></tr> </tbody> </table>				Name	Last Name	Phone	code				
Name	Last Name	Phone	code								

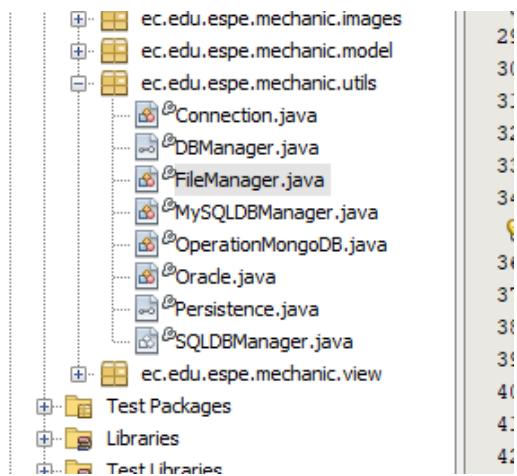
BILL

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	<input type="text"/>		
Discount	<input type="text"/>		
TOTAL	<input type="text"/>		
<input type="button" value="Return"/> <input type="button" value="Exit"/>			

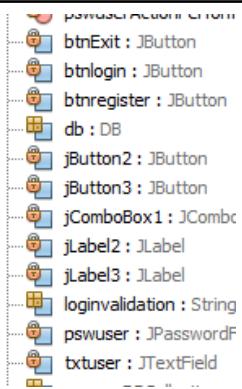
INVOICE
Sold to Ship to
Quantity Price Amount

Single Responsibility

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

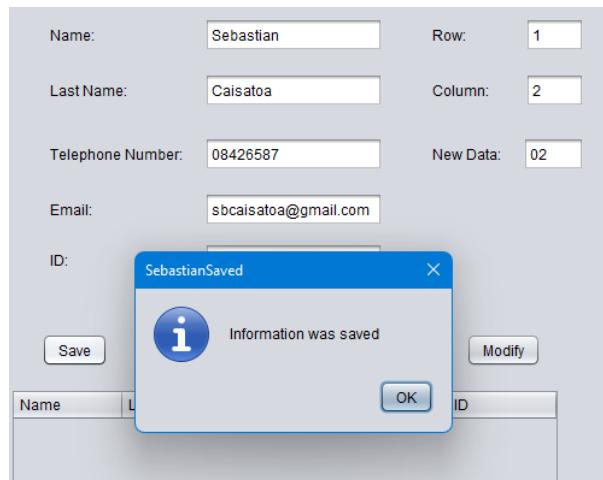


Undeclared Buttons



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 DBObject 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
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services Start Page frmRegistro.java | Product.java
Source History <default config> Dejar de compartir Ocultar
1 package ec.edu.espe.mechanic.controller;
2
3 import com.mongodb.BasicDBObject;
4 import com.mongodb.DBObject;
5
6 /**
7 * @author Sigma Programmers
8 */
9
10 public class ValueProduct {
11
12     BasicDBObject document = new BasicDBObject();
13
14     public DBObject addProduct(String name, int quanty, int price) {
15         document.put("Name", name);
16         document.put("Quanty", quanty);
17         document.put("Price", price);
18         return null;
19     }
20
21 }

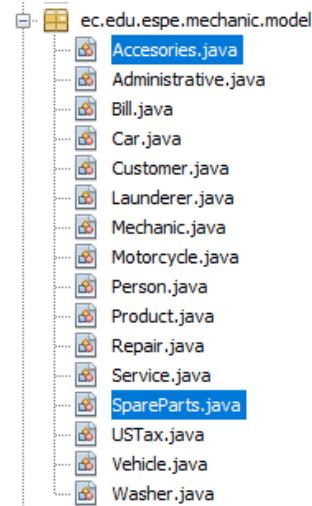
```

```

Mechanic - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services Start Page IvaCalculator.java | ValueVehicle.java
Source History <default config> Dejar de compartir Ocultar
1 package ec.edu.espe.mechanic.controller;
2
3 import com.mongodb.BasicDBObject;
4 import com.mongodb.DBObject;
5
6 /**
7 * @author Sigma Programmers
8 */
9
10 public class ValueVehicle {
11
12     BasicDBObject document = new BasicDBObject();
13
14     public DBObject addVehicle(int year, String Registration, String Plate, String trademark, String model, float mileage) {
15
16         document.put("Year", year);
17         document.put("Registration", Registration);
18         document.put("Plate", Plate);
19         document.put("Trademark", trademark);
20         document.put("Model", model);
21         document.put("Mileage", mileage);
22         return null;
23     }
24
25 }
26

```

Incorrect class names. Classes are written in the singular:



They are not names for the buttons:

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
    // TODO add your handling code here:
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    FrmEmployers frmEmployers = new FrmEmployers();
    frmEmployers.setVisible(true);
    dispose();

// TODO add your handling code here:
}

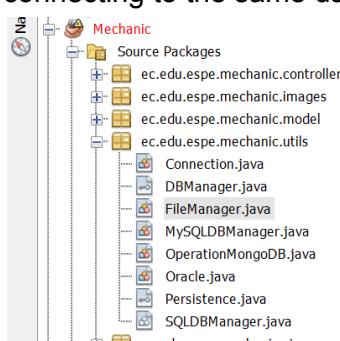
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    FrmEmployers frmEmployers = new FrmEmployers();
    frmEmployers.setVisible(true);
    dispose();
    // TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
    // TODO add your handling code here:
}

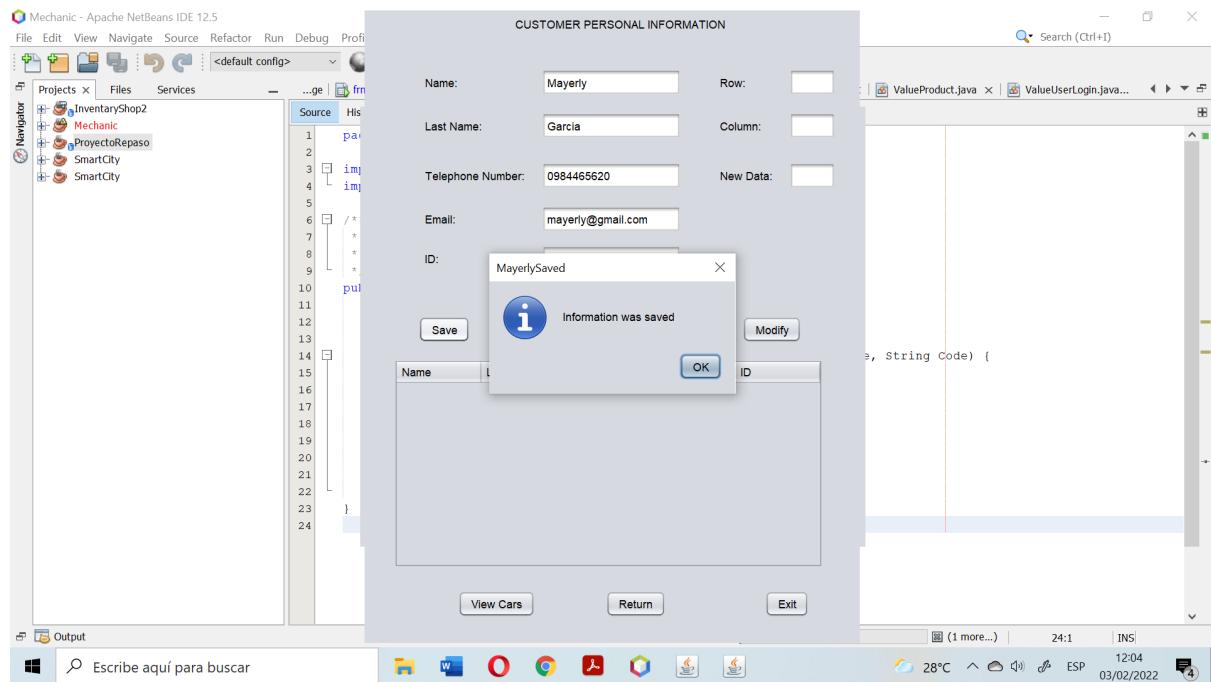
private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

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:

Name	Last Name	Phone	code

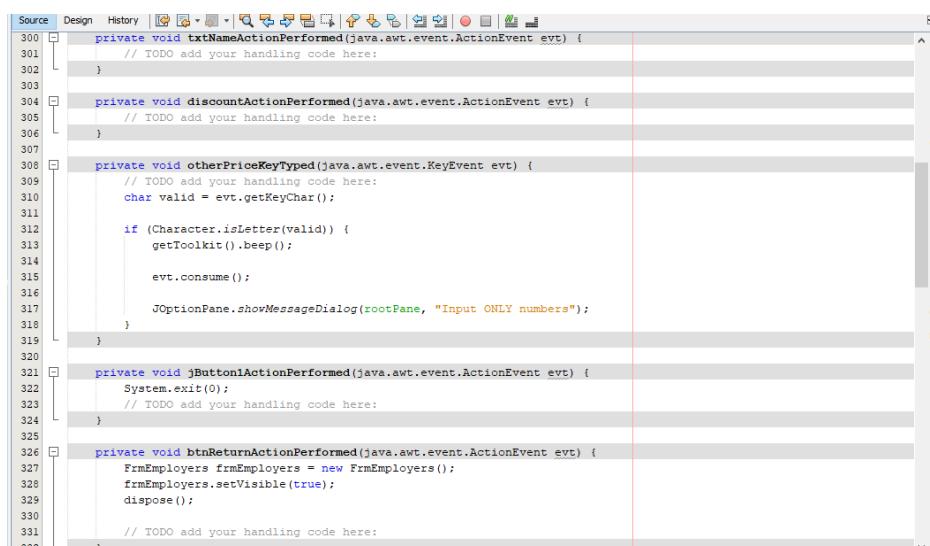
Single Responsibility Principle broken because the "add vehicle" object is not related to the "EmployeeController" class.

```
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  
45     Person person = new Person(name,lastName,plate);  
46  
47     String saveData = person.toString();  
48  
49  
50
```

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



The screenshot shows a Java IDE interface with five tabs at the top: 'FrmCar.java X', 'FrmCarManage.java X', 'FrmCustomerRecord.java X', 'FrmCustumerManage.java X', and 'Frm...'. Below the tabs is a toolbar with various icons. The main area contains two code snippets:

```
/*
 * 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.view;
```



```
/*
 * 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.view;
```

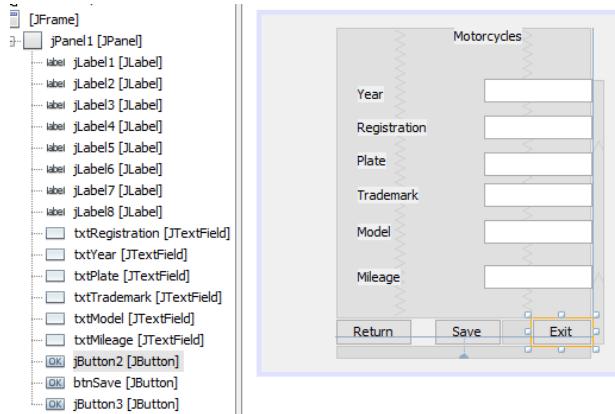
Below the code snippets is a section titled 'Unnecessary comments (FrmCarManage and FrmWorkManage)'.

To the left, there is a hierarchical tree view of the GUI components:

- [JFrame]
 - jPanel1 [JPanel]
 - label jLabel1 [JLabel]
 - label jLabel2 [JLabel]
 - label jLabel3 [JLabel]
 - label jLabel4 [JLabel]
 - label jLabel5 [JLabel]
 - label jLabel6 [JLabel]
 - label jLabel7 [JLabel]
 - label jLabel8 [JLabel]
 - txtRegistration [JTextField]
 - txtYear [JTextField]
 - txtPlate [JTextField]
 - txtTrademark [JTextField]
 - txtModel [JTextField]
 - txtMileage [JTextField]
 - OK jButton2 [JButton]
 - OK btnSave [JButton]
 - OK jButton3 [JButton]

On the right, a visual representation of the 'Motorcycles' form is shown. It has a title bar 'Motorcycles'. Inside, there are five text input fields labeled 'Year', 'Registration', 'Plate', 'Trademark', and 'Model'. Below these is another text input field labeled 'Mileage'. At the bottom are three buttons: 'Return', 'Save', and 'Exit', with 'Exit' being highlighted by a yellow box.

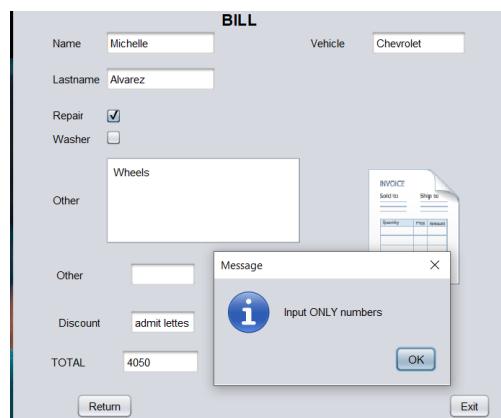
Unnecessary comments (FrmCarManage and FrmWorkManage)



Bad names for Return and Exit buttons (FrmMotorcycles)

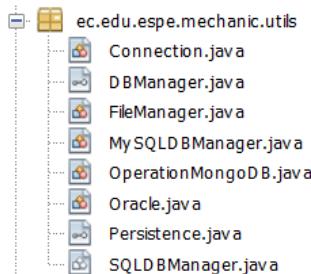
```
    DefaultTableModel modelo;
```

Table name in spanish (FrmWorkerRecord)



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 called "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.

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);
```

Cammel Case

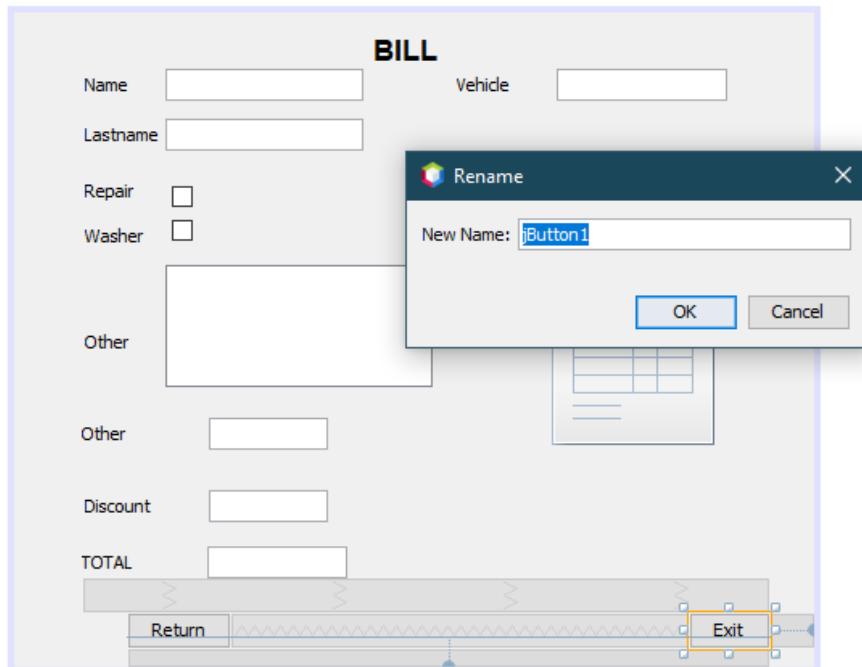
```

public class Launderer extends Washer {

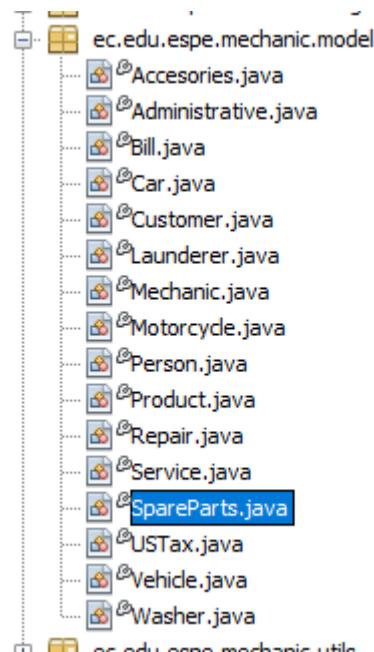
    public Launderer(String specialits, float waitTime) {
        super(specialits, waitTime);
    }

}

```



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() || txtOldData.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(), txtOldData.getText(), cmbField.getSelectedItem().toString());
                txtDataToUpdate.setText("");
                txtOldData.setText("");
                cmbField.setSelectedIndex(0);
                break;
            case 1:
                JOptionPane.showMessageDialog(null, "Information was NOT saved", txtDataToDelete.getText() + "NOT deleted",
                        JOptionPane.INFORMATION_MESSAGE);
                txtDataToUpdate.setText("");
                txtOldData.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) {

```

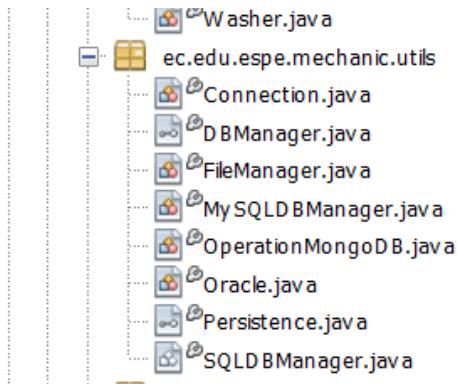
the button contains code

```

/*
 * 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.
 */

```

- The connection to the database is made by 2 classes when it should only have one



- Comments that are not necessary

```
public static void main(String args[]) {  
    /* Set the Nimbus look and feel */  
    Look and feel setting code (optional)  
    //</editor-fold>  
    /* Create and display the form */  
    java.awt.EventQueue.invokeLater(new Runnable() {
```

We must take into account the programming of the packages and know where each one is programmed.

The "getters" and "setters" must be programmed in the "model" package, not in the "controller" package, that's a big mistake.

```
16
17     /**
18      * @return the user
19      */
20     public String getUser() {
21         return user;
22     }
23
24     /**
25      * @param user the user to set
26      */
27     public void setUser(String user) {
28         this.user = user;
29     }
30
31     /**
32      * @return the password
33      */
34     public String getPassword() {
35         return password;
36     }
37
```

We must take into account the use of lowercase letters for the variables, since if we use uppercase letters we make a mistake.

```
document.put("Year", year);
document.put("Registration", Registration);
document.put("Plate", Plate);
```

Open/Closed

The names of the methods and variables are very general and are not specific.

```
boolean logged = false;
String readLine;
Persistence persistence;
```

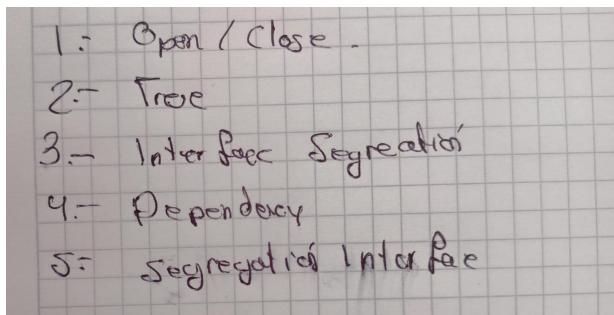
Team 8

Jonathan Insuasti

QUIZZES about SOLID Principles

ANSWERS:

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



UNIVERSIDAD

ESPE

Apellido

Ricardo

- 1) Open closed
- 2) Review
- 3) Interface Segregation
- 4) Dependency Inversion

DIA MES AÑO

NOTA

Mayerly García

OPP 7490

- Open principle
- Close principle.
- $\left. \begin{array}{l} \text{Data members} \\ \text{Method declarations} \\ \text{Method definitions} \end{array} \right\}$ True.
- 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, Responsibility principle.

Día MES Año
Alejandro De la Cruz 2490

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

- 1) Close / Open Jthink
- 2) Review
- 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 Caisatoca

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 principal.

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: Leorel 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 Merchán
ODP 7490

1. Open / Clase Principal

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: Surface 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

Jaimie Eixor

OOP 3490

- 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

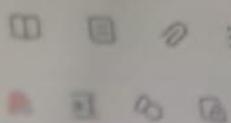
2. True

3. Interface segregation

4. Dependency Inversion principle

5. Interface Segregation

Quiz



Quiz

Name: José Francisco Schüller M. Shigae

1 Open Close

2 Review

3 Abstraction Core Idea

4 Derendent ↴

5



Name: Daniel Lincango

NRC: 7490

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

DD MM AA

N. Benjamin Cadena

7490 - 00P,

Thursday, February 3rd 2022

1.-

Open, closed, principle

2.-

Reusability

3.-

Interface segregation

4.-

Dependency Inversion

5.-

Single Responsibility principle

Norma

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

IRC: 7490

Answers:

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

Mateo Maldonado

- What are 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 Asumanwa

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

NQC: 7490

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

Payo Sebastián

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

Intencion / Open / closed Principle

2. -

3. -

Interface Segregation Principle

4. The client classes should not depend

Dependency inversion Principle

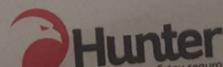
5. A component should only do what it has to.

Interface Segregation

Scanned by TapScanner

- 1. Open / closed Principle
- 2. Single responsibility Principle
- 3. Interface Segregation Principle
- 4. Dependence inversion Principle
- 5. Liskov's Substitution

Camilio Tecco



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

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

Cristian Arroba Nrc: 7490

1. Open class
2. Tress
3. Interface segregation
4. Dependency
5. Segregation interface