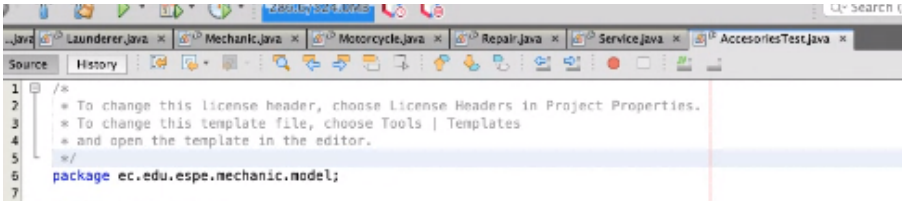
WS32 - SOFTWARE PRINCIPLES GROUPS

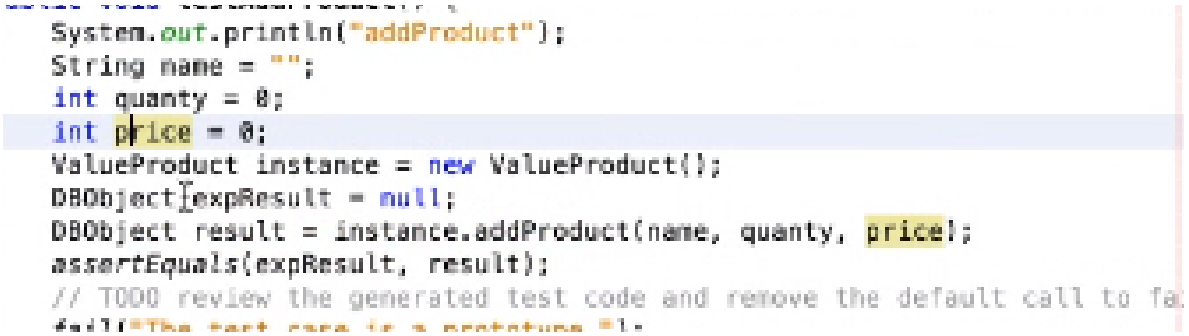
DATE: 03th february 2021

TEAM 1:
Salma Villegas
Alexander Ruano
Daniel Lincango
Mateo Maldonado
Leandro Quinga
Joel Zeas



Innecesary comments, unclean code

Unclean Code



```
*/
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

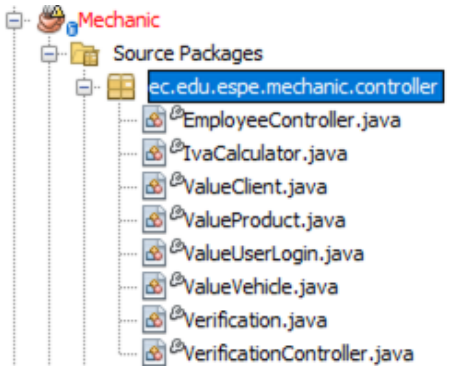
```
    */
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
        // TODO add your handling code here:
    }
```

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 ResponsibilityPrinciples:** 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;
}
```

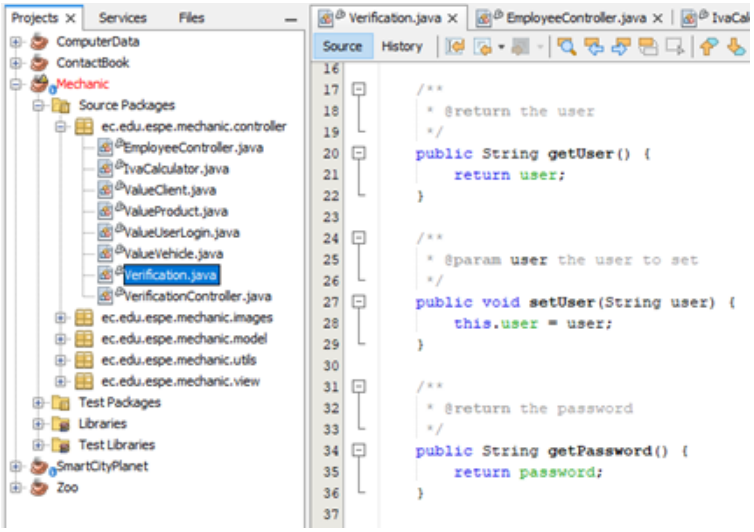
- Use an else variable that replaces the return and returns an actual data

```java
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

```java
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

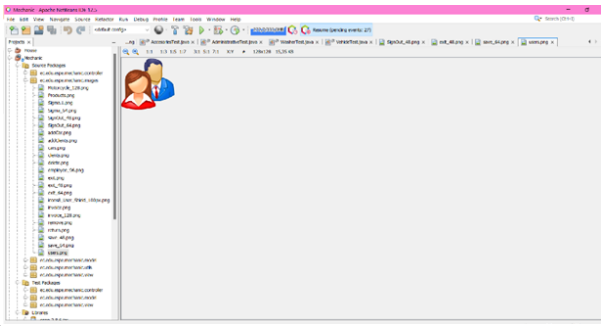
```java
public boolean login(Verification verification, String user, String password) {

    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) {
```

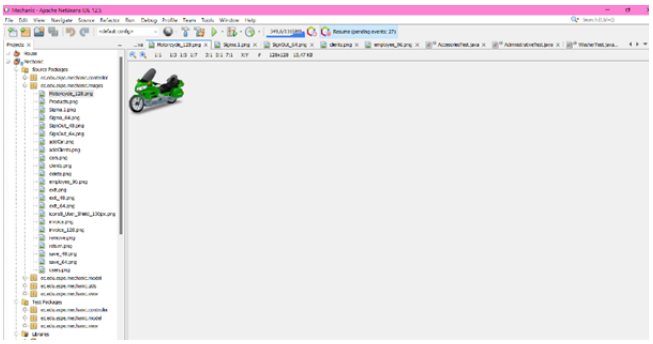
Source Packages

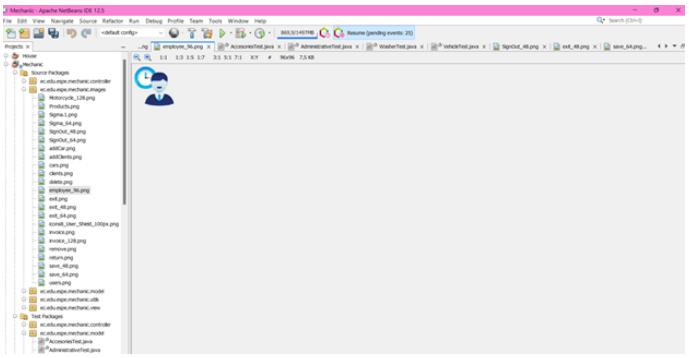
Package: ec.edu.espe.mechanic.images

Nothing to mention, each class has only one image.

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





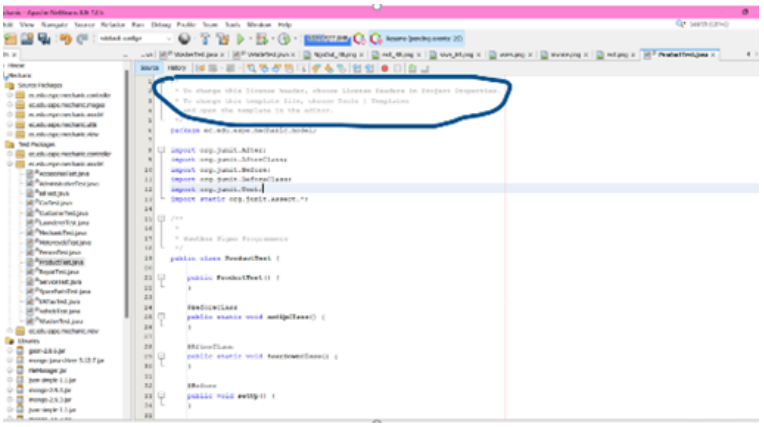


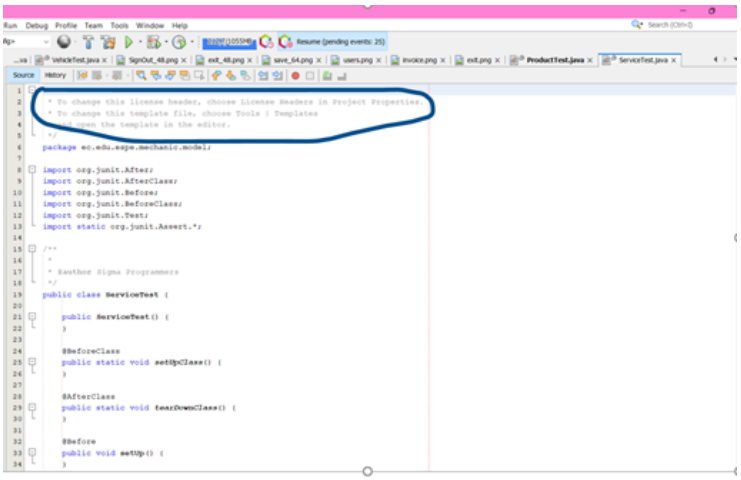
Test Packages

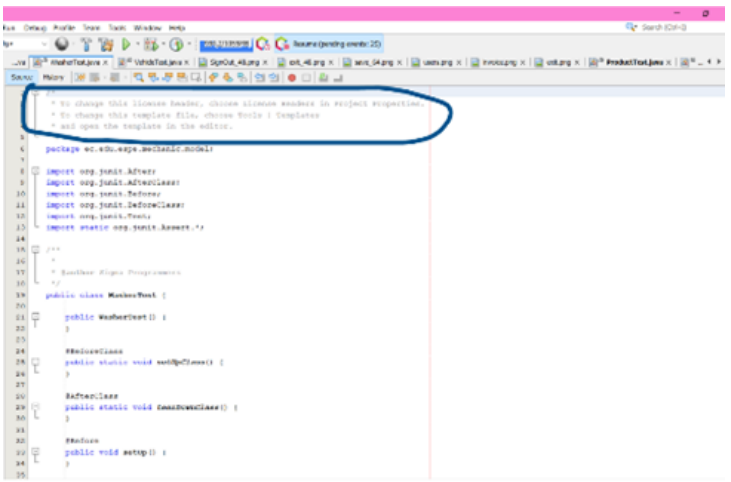
Package: ec.edu.espe.mechanic.model

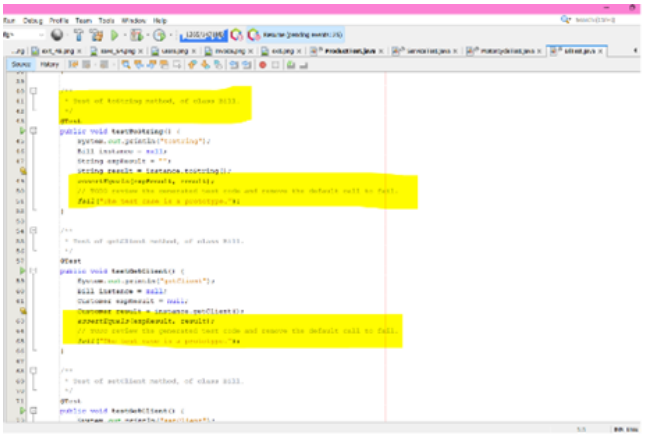
Unnecessary comments, unclean code,

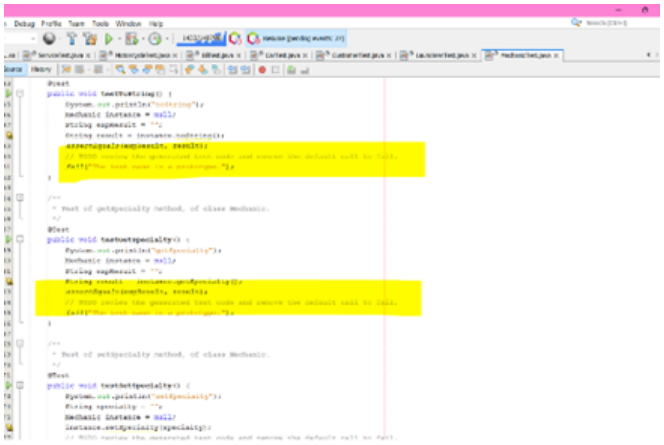
Used Principles: Single Responsibility Principle (This package is being used for testing only, so each class has a unique reason)











TEAM 2:
Mateo Landazuri
Andy Calderon
Leonel Mantuano
Pablo Bustillos
Richar Maisincho
Sebastian Palacios

- Attributes in different sizes are written with camelcases

```java
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

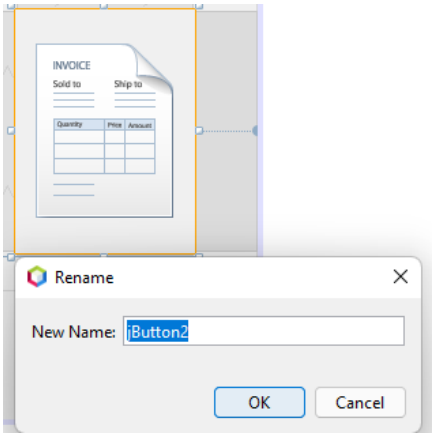
```
/**
 * @return the Email
 */
public String getEmail() {
    return Email;
}

/**
 * @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;
}
```

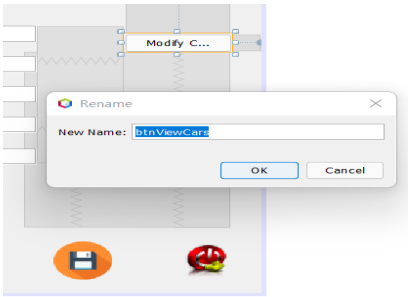
```
61          String name ;
62          int quanty;
63      >>>>>>> e4b3f82c2e54cd426697e6d1e44a137df95afedb
64          */
65      }
66
```

- 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



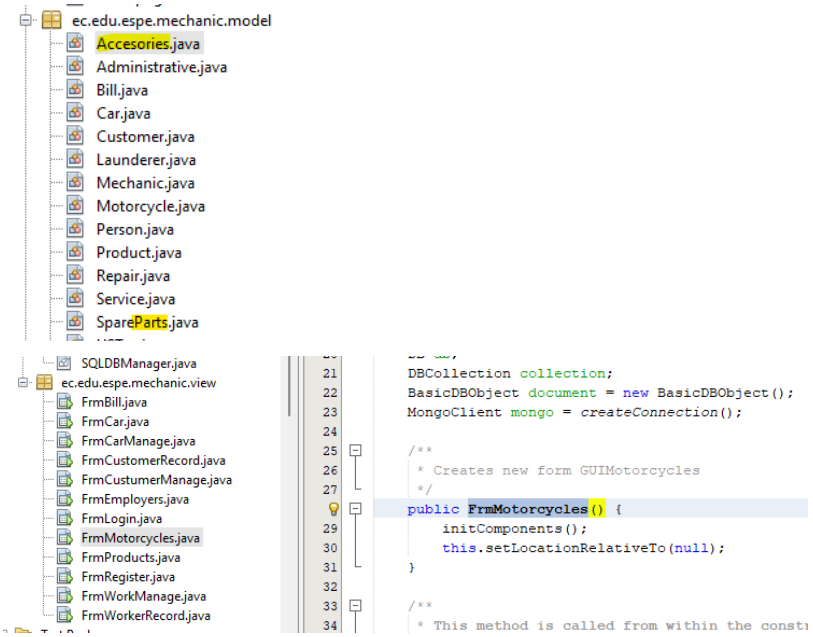


- Capital letters are not used

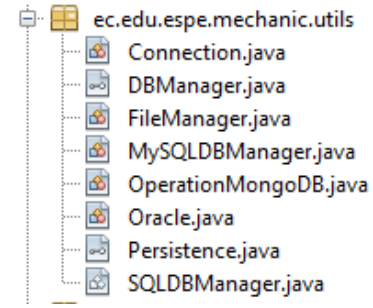
```java
public class ValueProduct {

    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;
    }
}
```

- Class names must be singular nouns

```java
        DB db = mongo.getDB(dataBase);
        DBCollection dbCollection = db.getCollection(collection);
        BasicDBObject searchedName = new BasicDBObject();
        BasicDBObject updateData = new BasicDBObject();
        try {
```



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

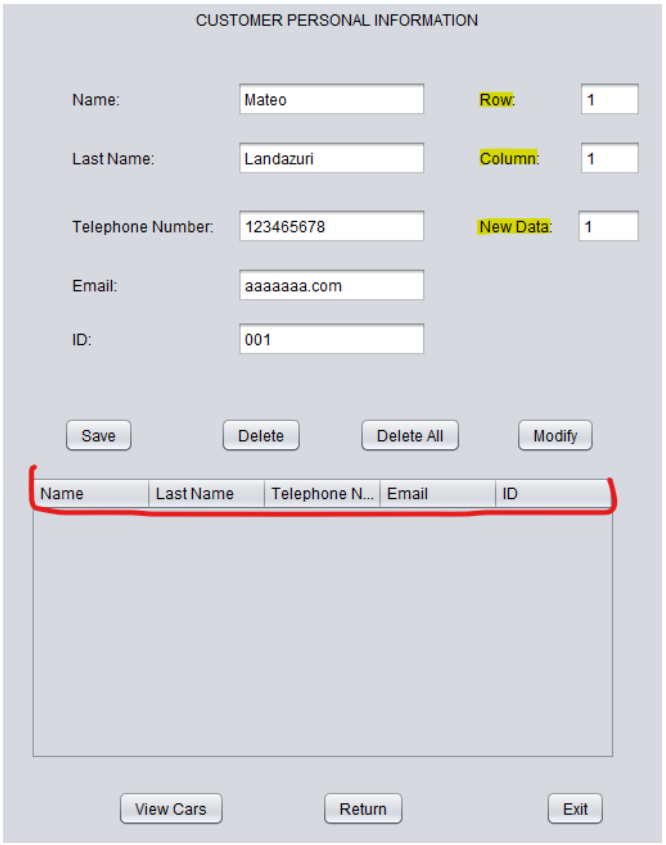


-Open / Closed Principe

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

```java
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).



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 | Accesories.java

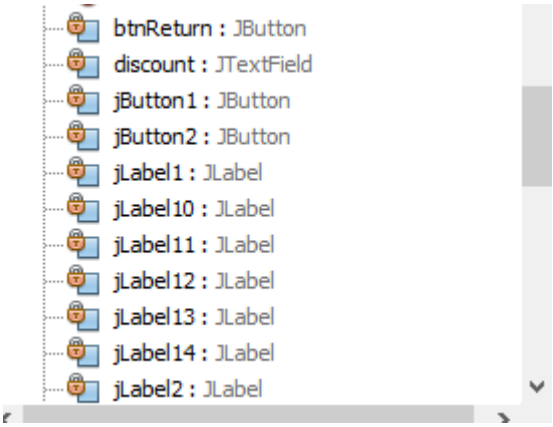
Source | History

```java
package ec.edu.espe.mechanic.model;

/**
 *
 * @author Sigma Programmers
 */
public class Accesories extends Product {

    public Accesories(String name, int quanty, int price) {
        super(name, quanty, price);
    }

}
```

- Classes must be singular

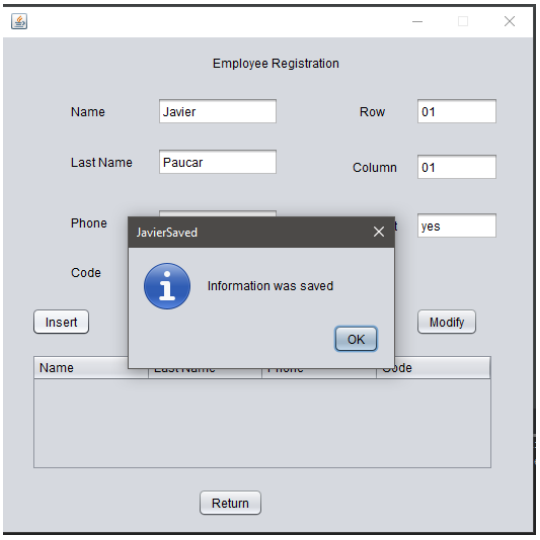
```java
        */
    public void setPrice(int price) {
        this.price = price;
    }

    /*=======
    String name ;
    int quanty;
>>>>>>> e4b3f82c2e54cd426697e6d1e44a137df95afedb
        */
}
```

- uncommented code

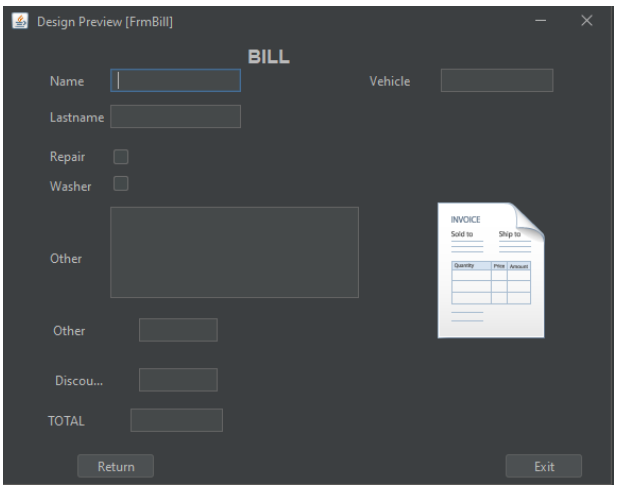


-Must have specific names

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

<u>Open closed principle</u>

Declaration of variable

```
ivate void btnDeleteActionPerformed(java
    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.

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

    public Vehicle(int year, String Regi
```

Unnecessary comments

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

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Ru
```

Unprogrammed code

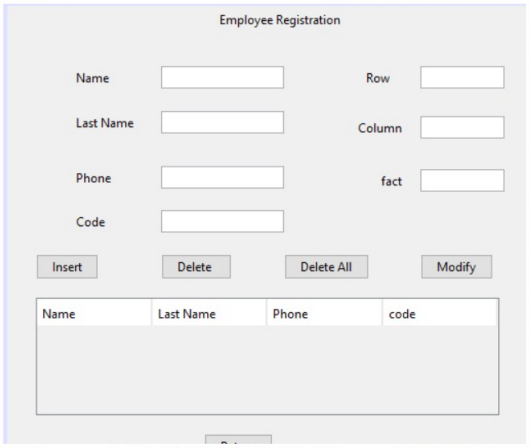
```
    private void txtLastNameActionPerformed(java.aw
        // TODO add your handling code here:
    }

    private void txtIDActionPerformed(java.awt.even
        // TODO add your handling code here:
    }

    private void txtNameActionPerformed(java.awt.ev
        // TODO add your handling code here:
    }
```

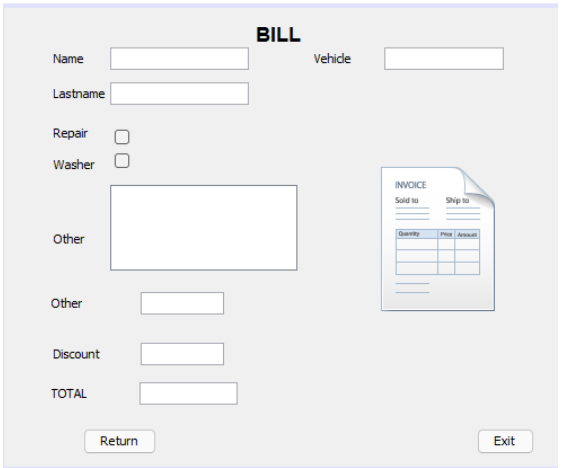
It calls the mongo db Connection in a button.

```
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

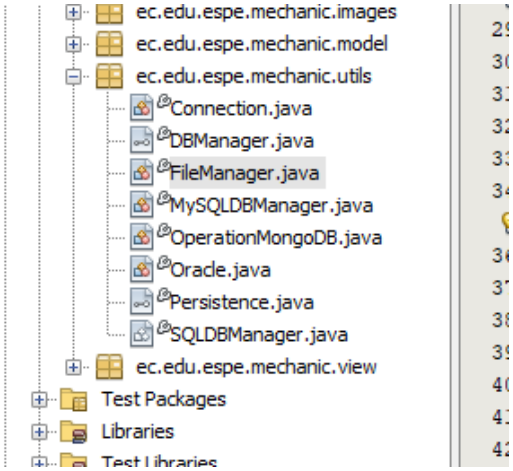
table don't have information of form



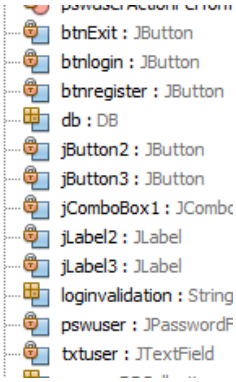


Single Responsibility

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

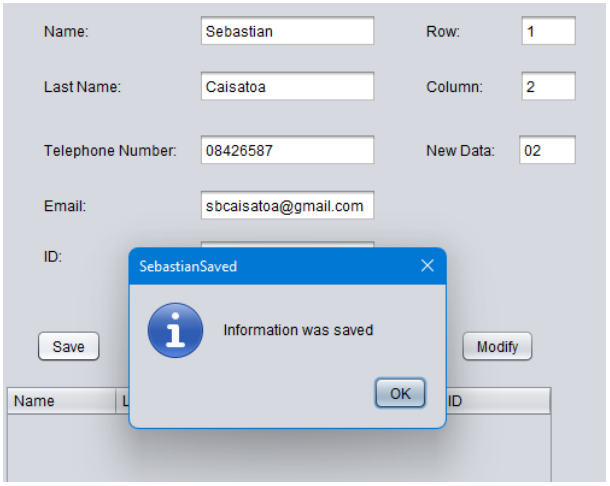


Undeclared Buttons



Segregation principle

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

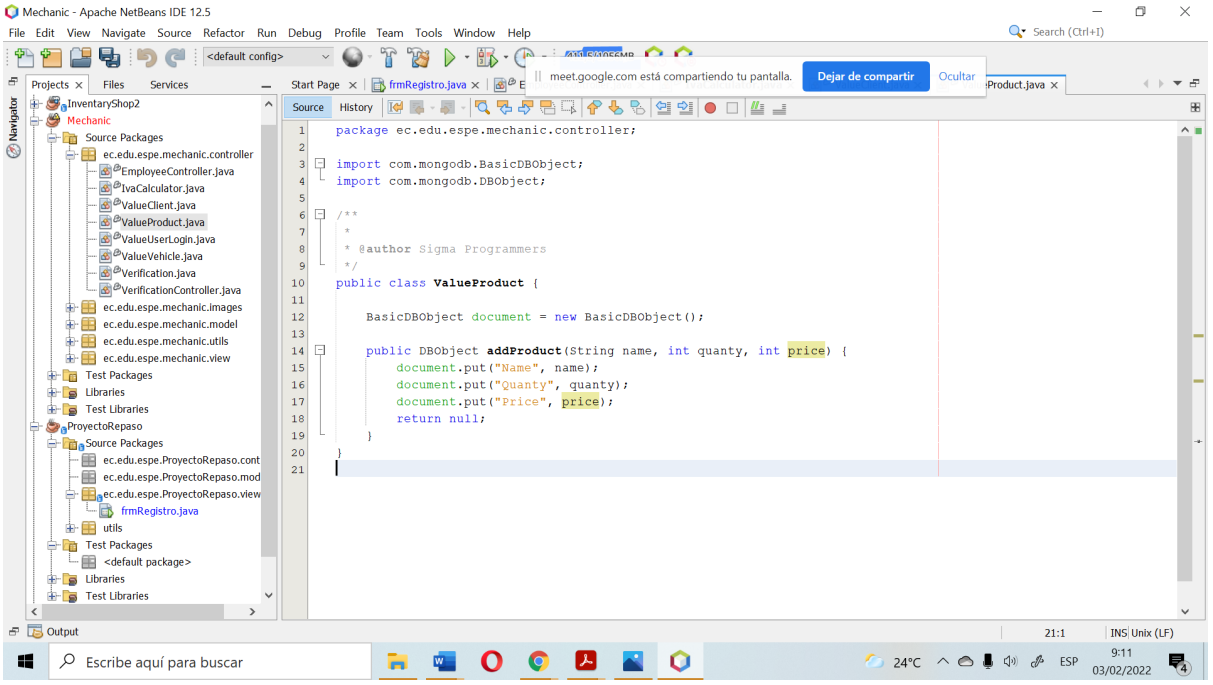


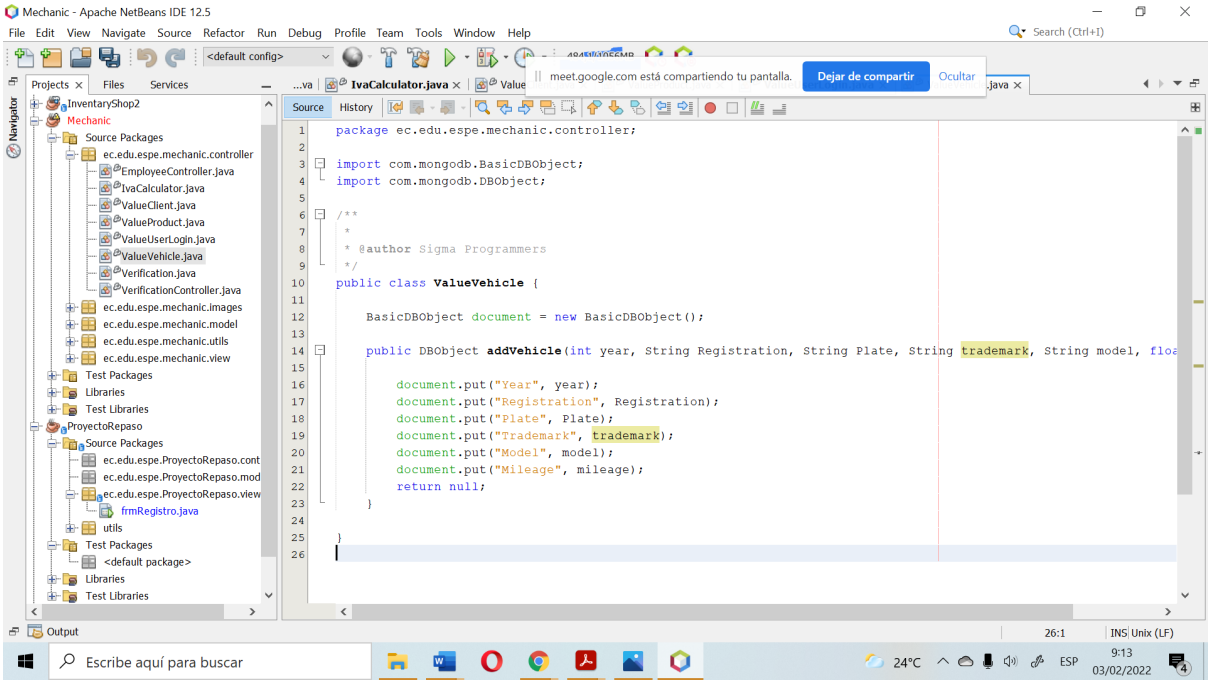
TEAM 5:

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

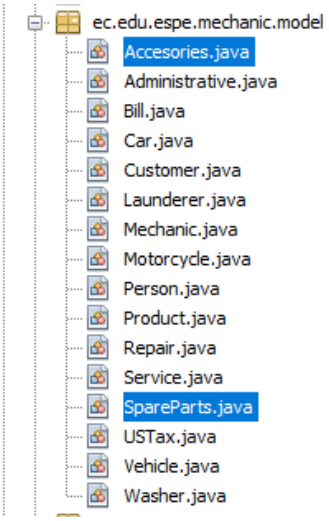
```
BasicDBObject document = new BasicDBObject();

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





Incorrect class names. Classes are written in the singular:

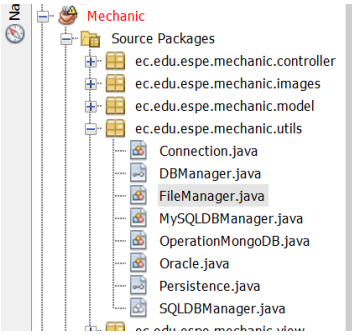


They are not names for the buttons:

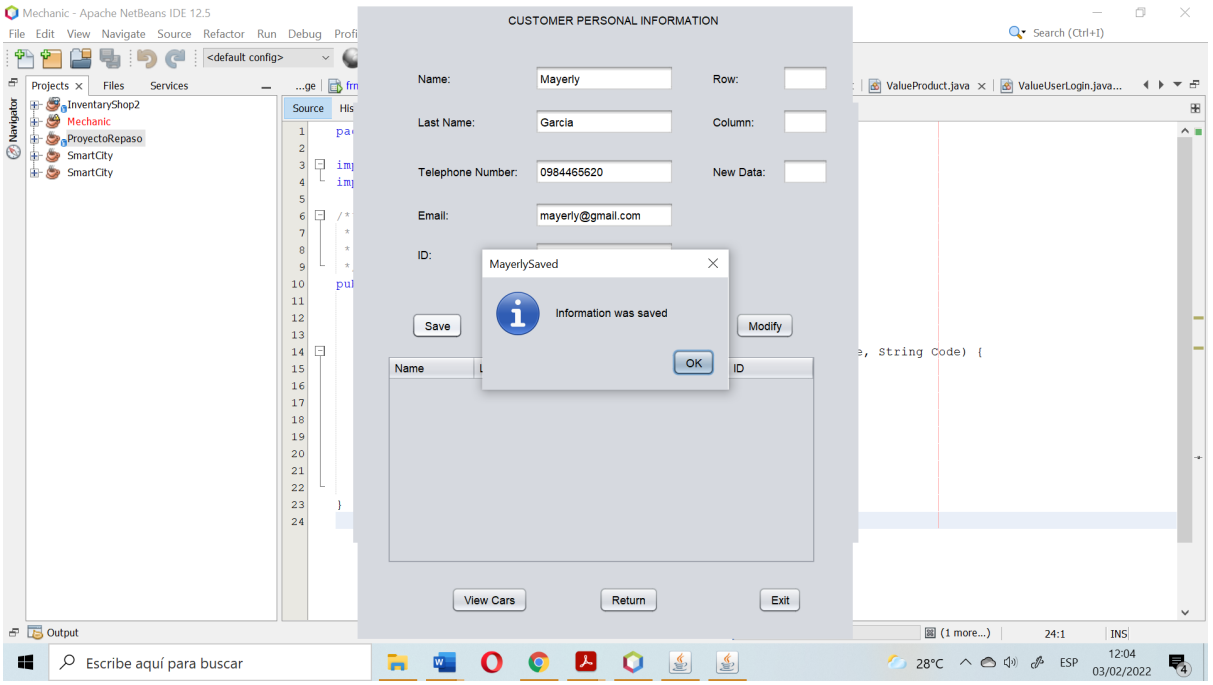
```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
        // TODO add your handling code here:
    }

    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:
    }
```

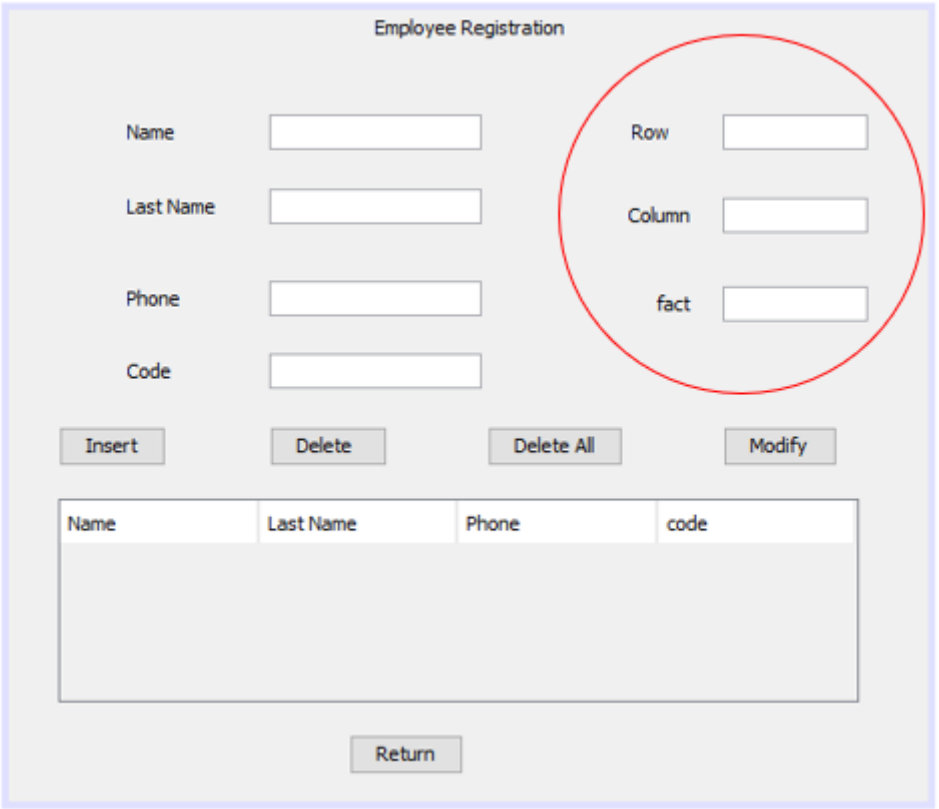
In the utils package there should be one class to the database, but no more than two classes connecting to the same database. (**Single Responsibility**)



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:



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

```java
package ec.edu.espe.mechanic.controller;

import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;

/**
 *
 * @author Sigma Programmers
 */
public class EmployeeController {

    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;
    }
}
```

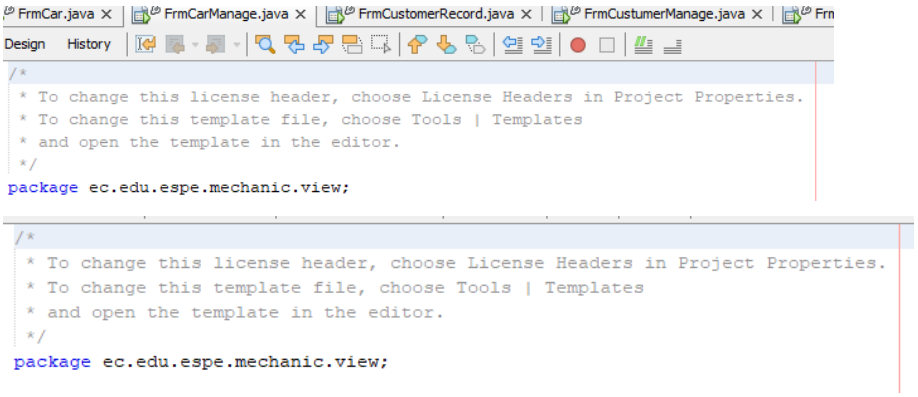
Bad declaration of the person Method of the Person class, missing to declare id

```java
    public Person(int id, String name, String lastName, String plate) {
        this.id = id;
        this.name = name;
        this.lastName = lastName;
        this.plate = plate;
    }
```

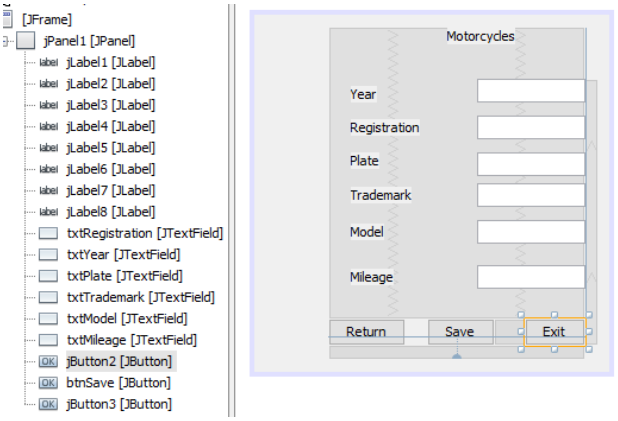
```java
        Person person = new Person(name,lastName,plate);

        String saveData = person.toString();
```

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

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

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

    private void otherPriceKeyTyped(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        char valid = evt.getKeyChar();

        if (Character.isLetter(valid)) {
            getToolkit().beep();

            evt.consume();

            JOptionPane.showMessageDialog(rootPane, "Input ONLY numbers");
        }
    }

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

    private void btnReturnActionPerformed(java.awt.event.ActionEvent evt) {
        FrmEmployers frmEmployers = new FrmEmployers();
        frmEmployers.setVisible(true);
        dispose();

        // TODO add your handling code here:
    }
```



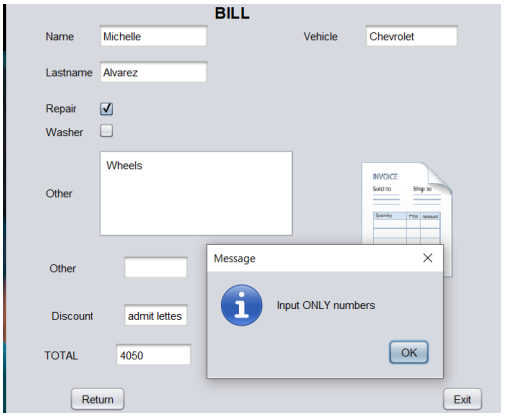
Unnecessary comments (FrmCarManage and FrmWorkManage)



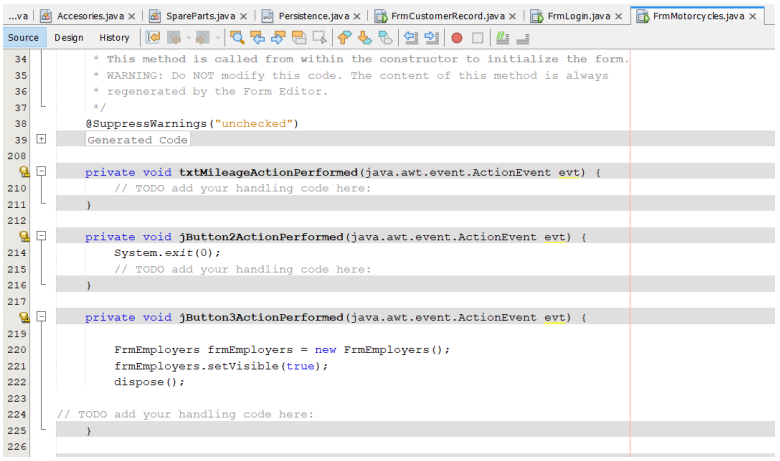
Bad names for Return and Exit buttons (FrmMotorcycles)

```
public class FrmWorkerRecord extends javax.swing.JFrame {

    DefaultTableModel modelo;
```

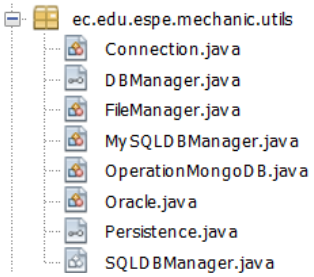
Table name in spanish (FrmWorkerRecord)



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



In the FrmMotorcycle the 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

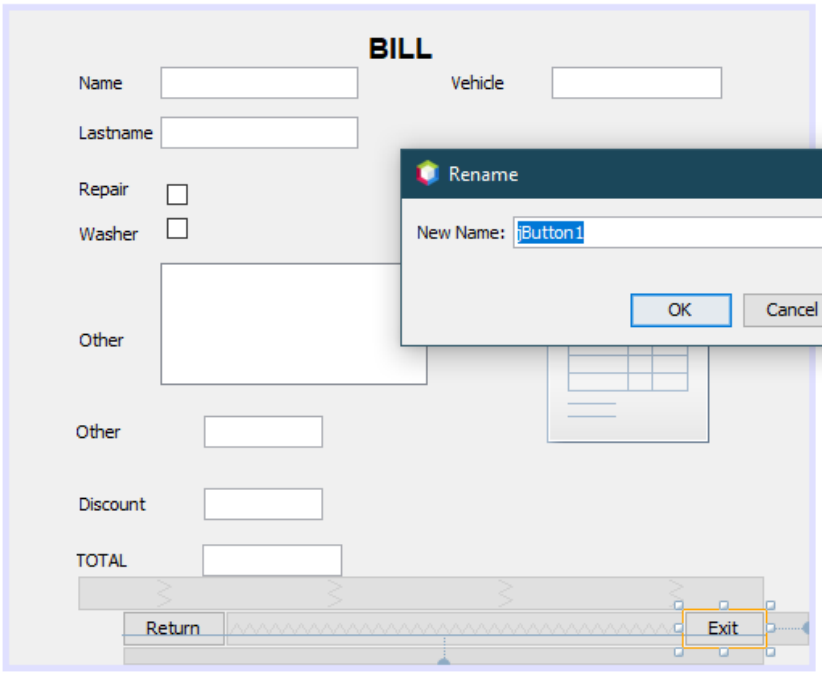
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
    System.exit(0);
    // TODO add your handling code here:
```

Change button variable name Exit

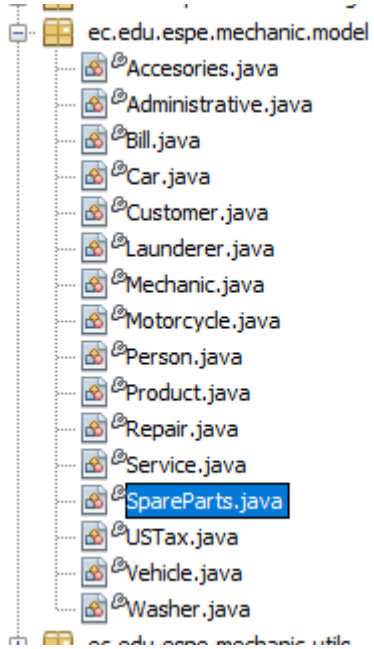
```
public class Customer extends Person {

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

}
```

Cammel Case

```java
public class Launderer extends Washer {

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

}
```



Name of the buttons

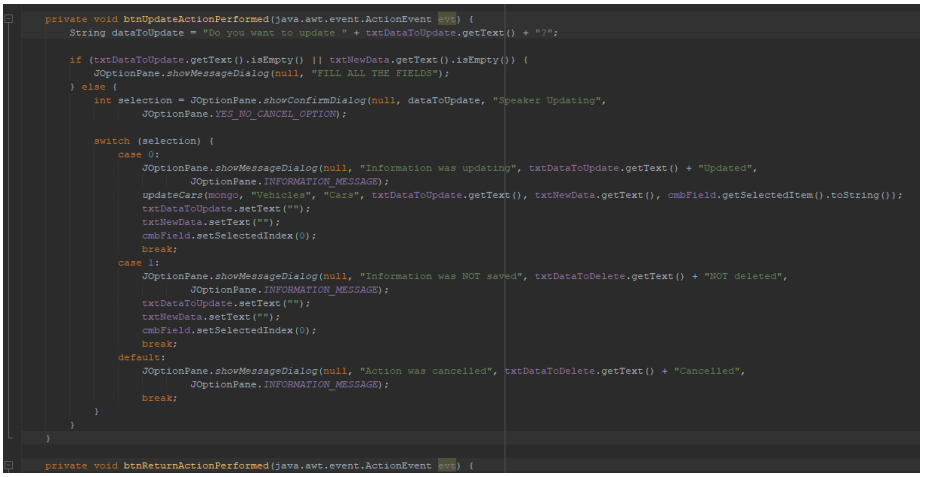


some classes are written in plural

Default comments are not deleted

```java
/*
 * 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

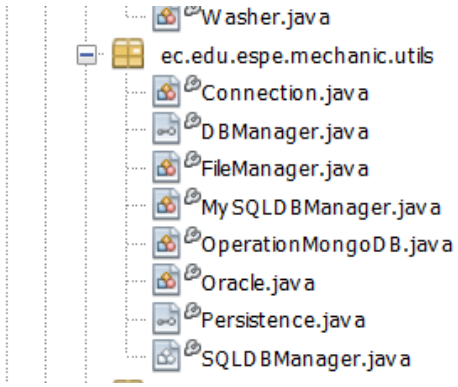


```java
private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    String dataToUpdate = "Do you want to update " + txtDataToUpdate.getText() + "?";

    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);
                txtDataToUpdate.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

```java
/*
 * 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>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</editor-fold>
    //</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;
```

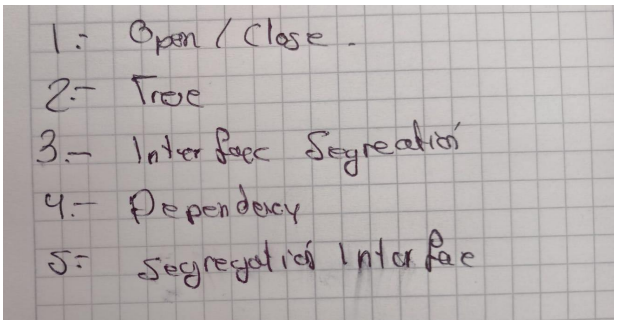
Team 8

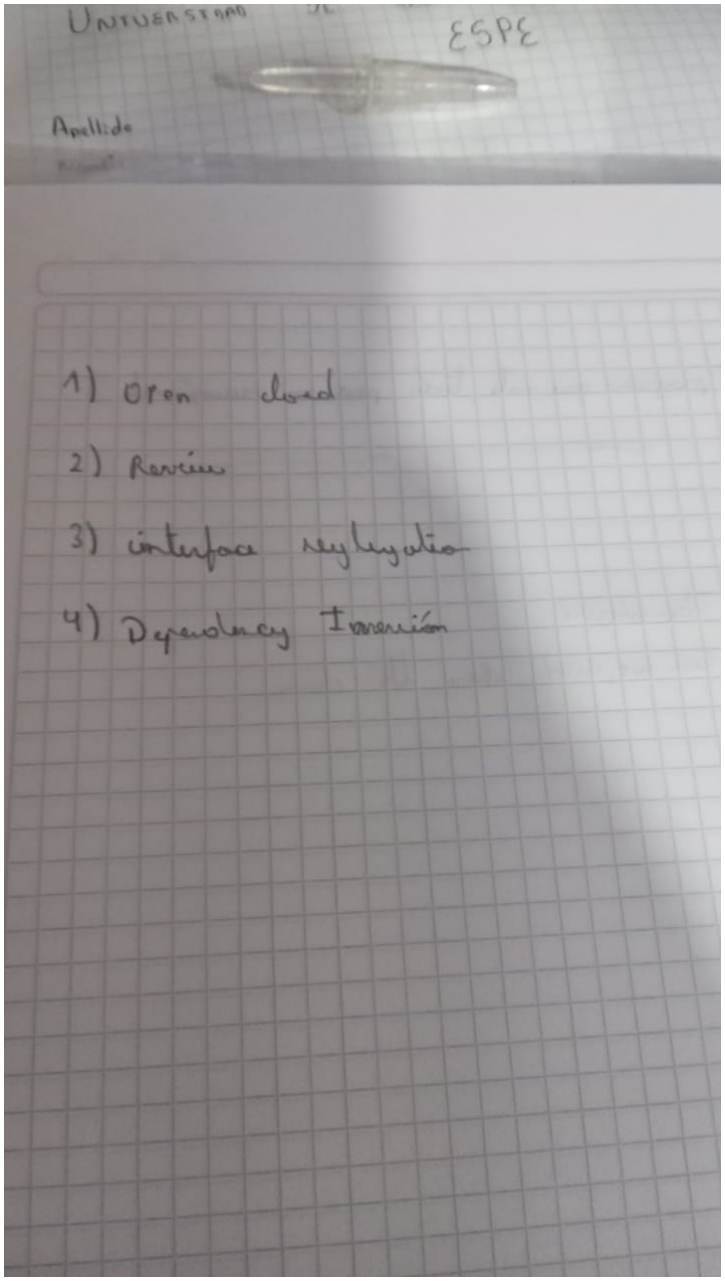
Jonathan Insuasti

QUIZZES about SOLID Principles

ANSWERS:

1. **O**pen /closed principle (Open/close)
2. **L**iskov Substitution
3. **I**nterface Segregation
4. **D**ependency Inversion
5. **S**ingle Responsibility




UNIVERSIDAD
ESPE
Apellido
1) Open closed
2) Review
3) interface segregation
4) Dependency Inversión

DIA MES AÑO NOTA /

Mayerly Garcia OPP 7490

1) · Open principle
· Close principle

- Data members
  Method declarations } True
  Method definitions

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

- Dependency Inversion

- Interface Segregation

Name Altamirano Cristhian

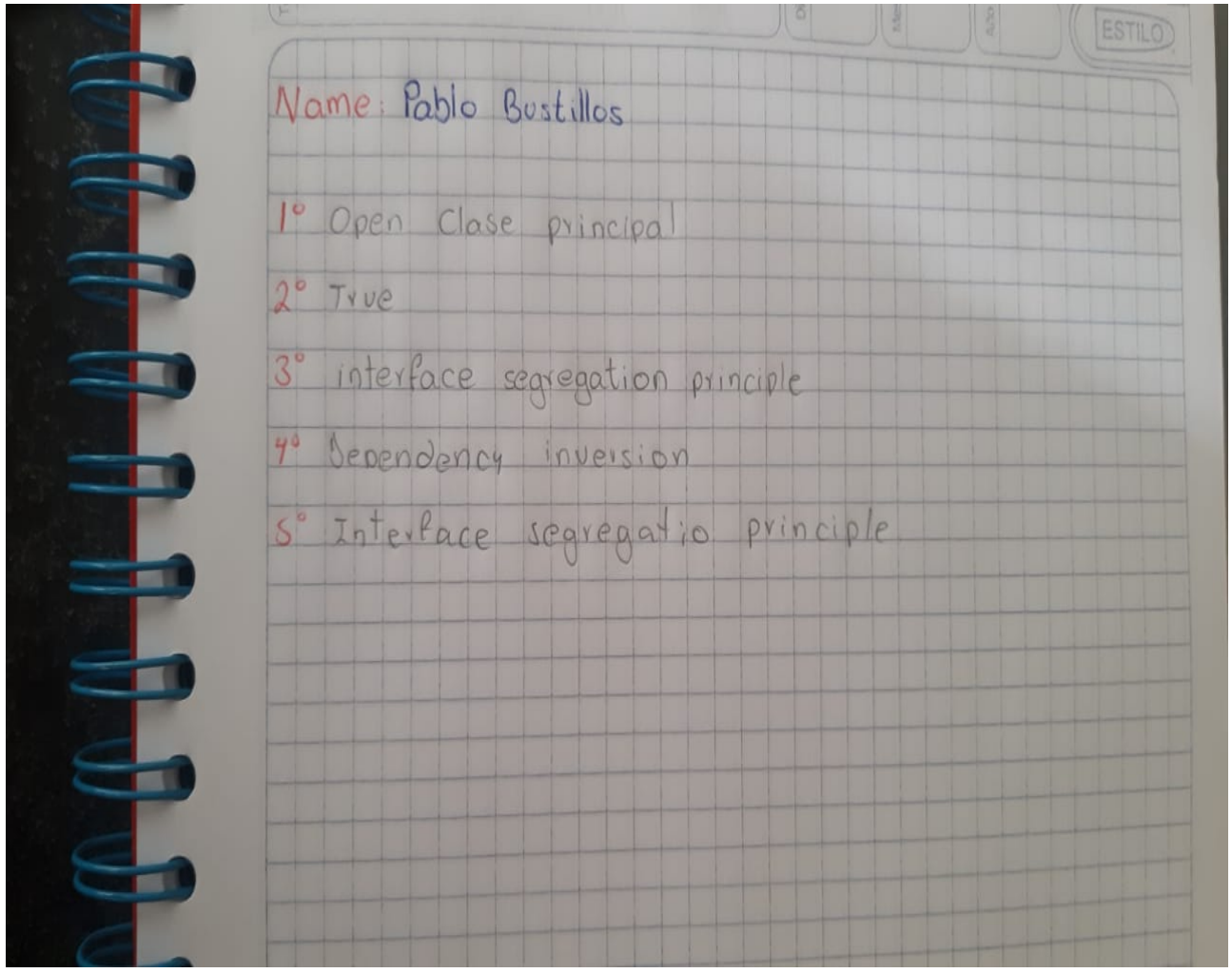
NRC: 7490

What is

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

Alejandro De La Cruz 7490

1) Close / Open) think
2) Review
3) Interface Segregation principle
4) Dependency Inversion
5) Single Responsability principal


ESTILO
Name: Pablo Bustillos
1° Open Clase principal
2° Tvue
3° interface segregation principle
4° Dependency inversion
5° Interface segregatio principle

Sebastián Caisatoa

1.- Open / Close Principle

2.- True

3.- Interface Segregation

4.- Dependency Inversión

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 responsability

Name: Salma Villegas

1. Open - Close Principle

2. True

3 Linkov Substitution Principle.

4. Dependency Inversion Principle

5. Single Responsability Principle.

MELISSA GÓMEZ

1.- OPEN - CLOSE

2.- LISKOV

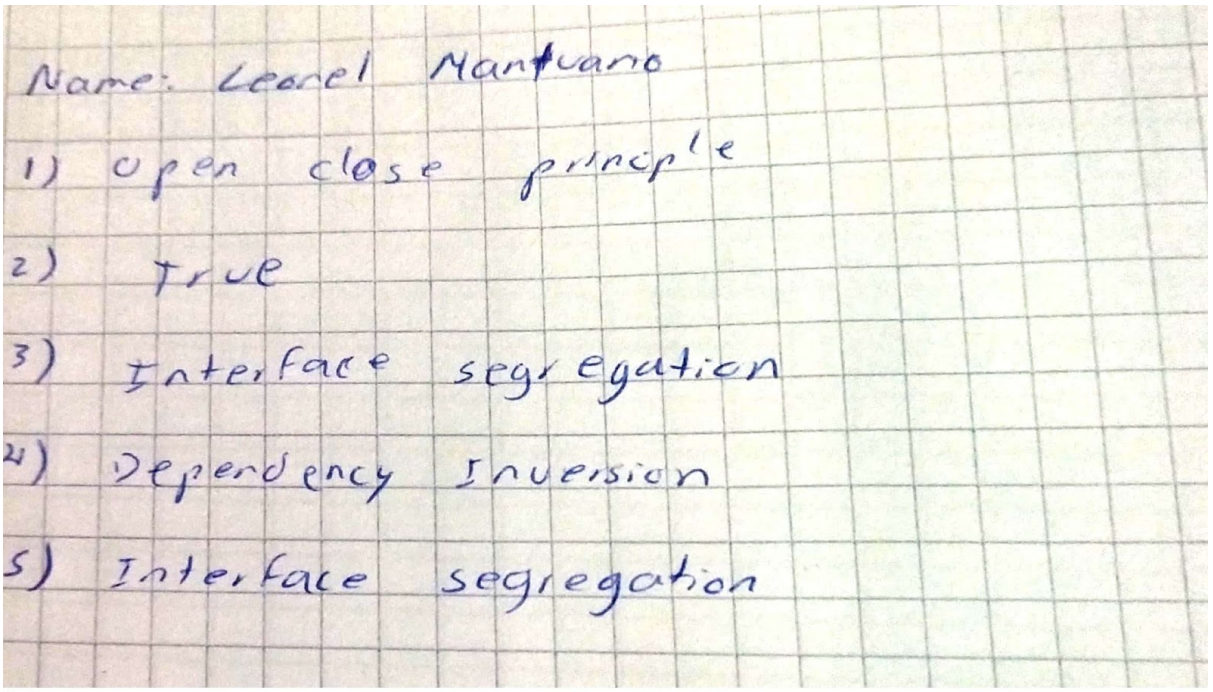
3.- DEPENDENCY INVERSION

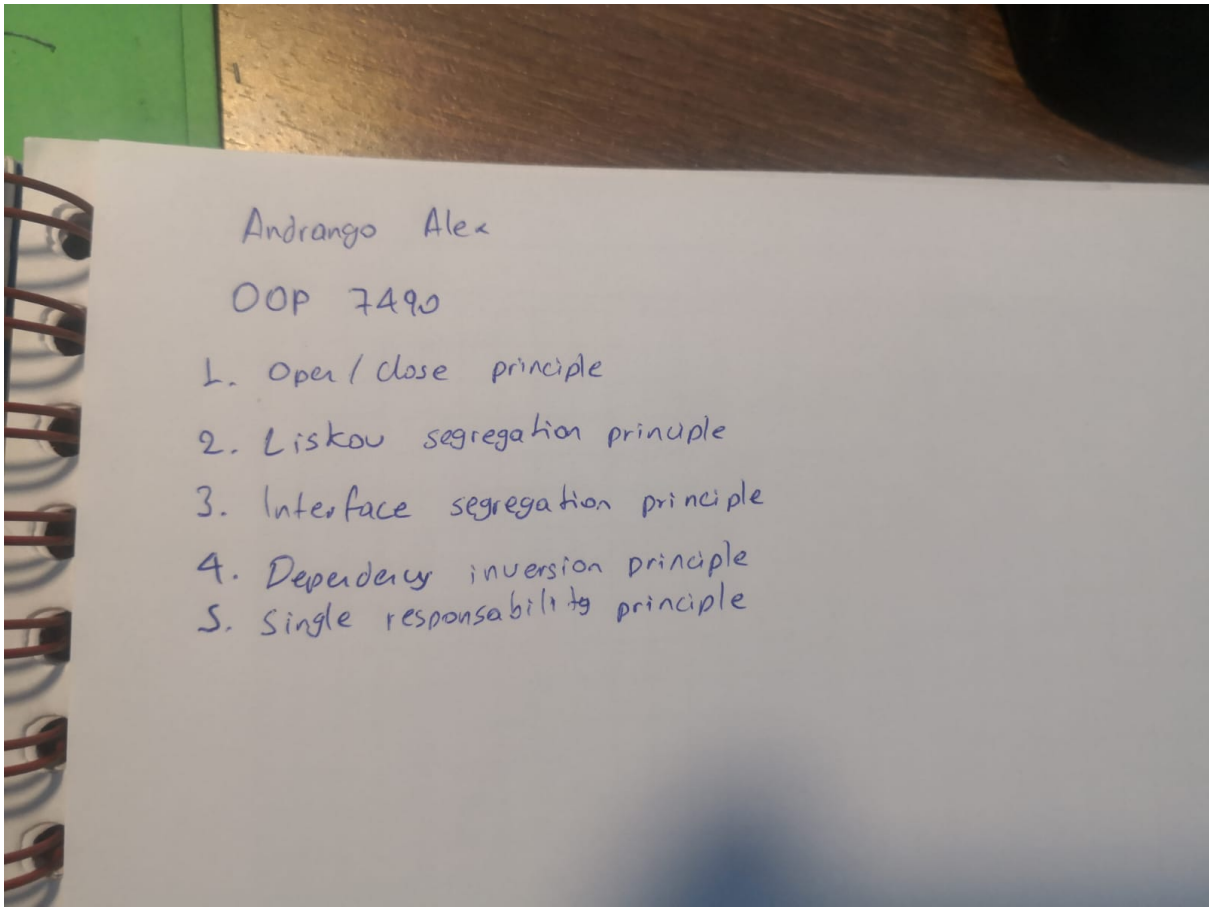
4.- INTERFACE SEGREGATION

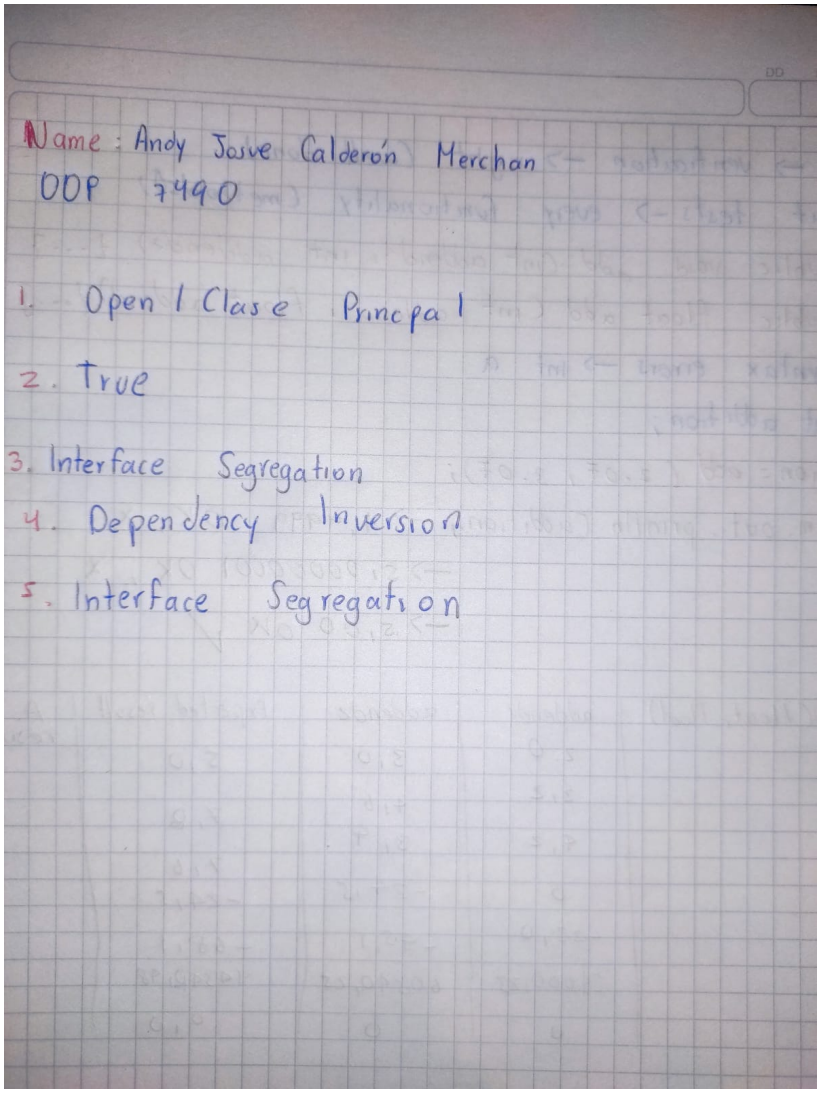
5.- SINGLE RESPONSABILITY

Michelle Alvarez OOP 7490

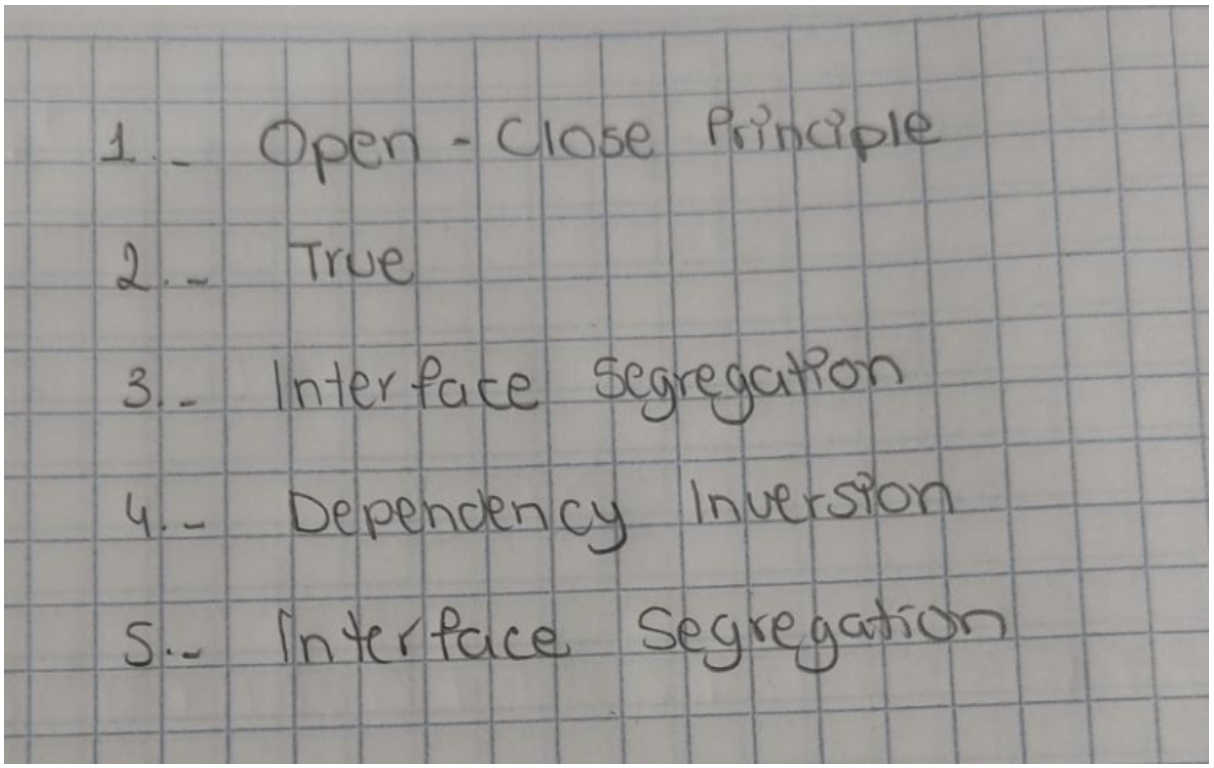
1. Open / Close Principle
2. True
3. Interface Segregation
4. Dependency Inversion
5. Interface Segregation

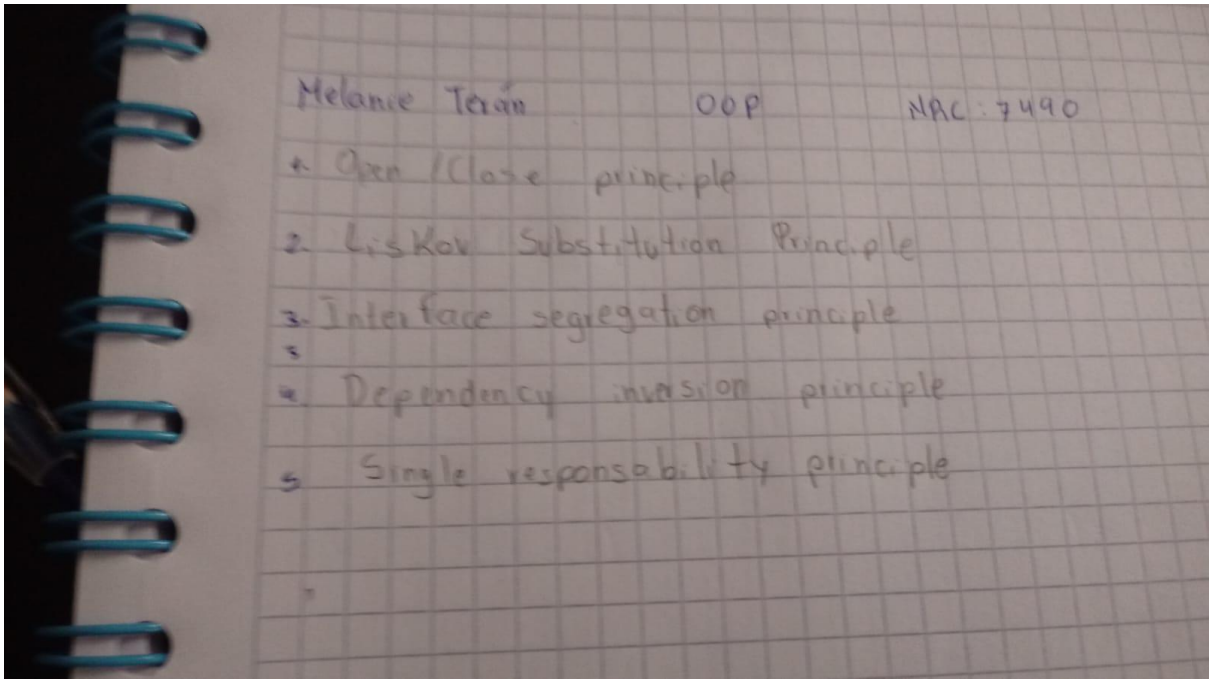



Andrango Alex
OOP 7490
1. Open / close principle
2. Liskov segregation prinuple
3. Interface segregation principle
4. Dependency inversion principle
5. Single responsability principle



Katherin Bravo




Melanie Terán OOP NRC: 7490

1. Open /Close principle
2. Liskov Substitution Principle
3. Interface segregation principle
3
4. Dependency inversion principle
5. Single responsability principle

Andrea Tapia 7490.

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

Quizizz about SOLID Principles

Name: Andrade Alan NRC: 7490

1: Close - Open - Principle

2: Review

3: Sinterface segregation principle.

4: Dependency Abstract principle

5: Sinterface Segregation

Darling Cruz

1 Open - Close Principle

2 True

3 You have segregation

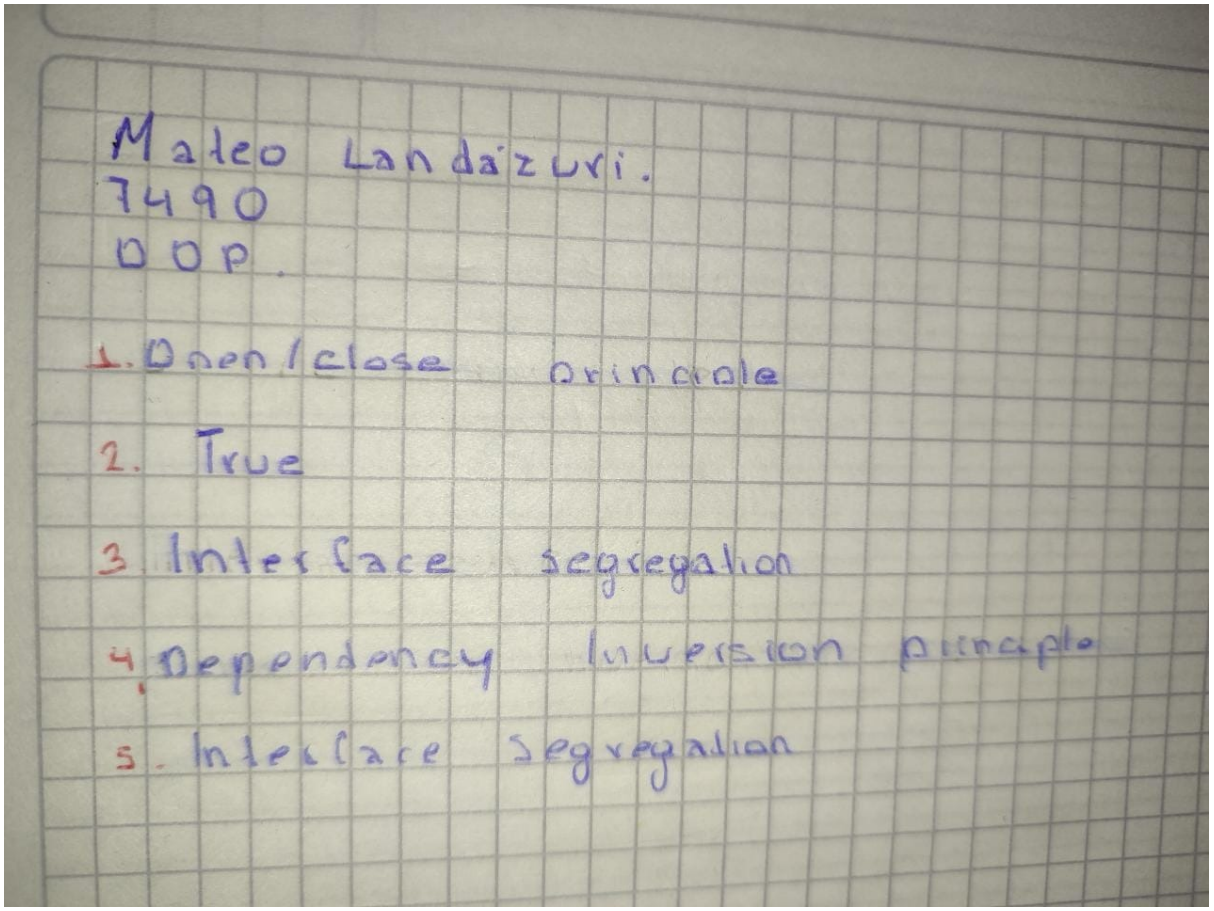
4 Dependency

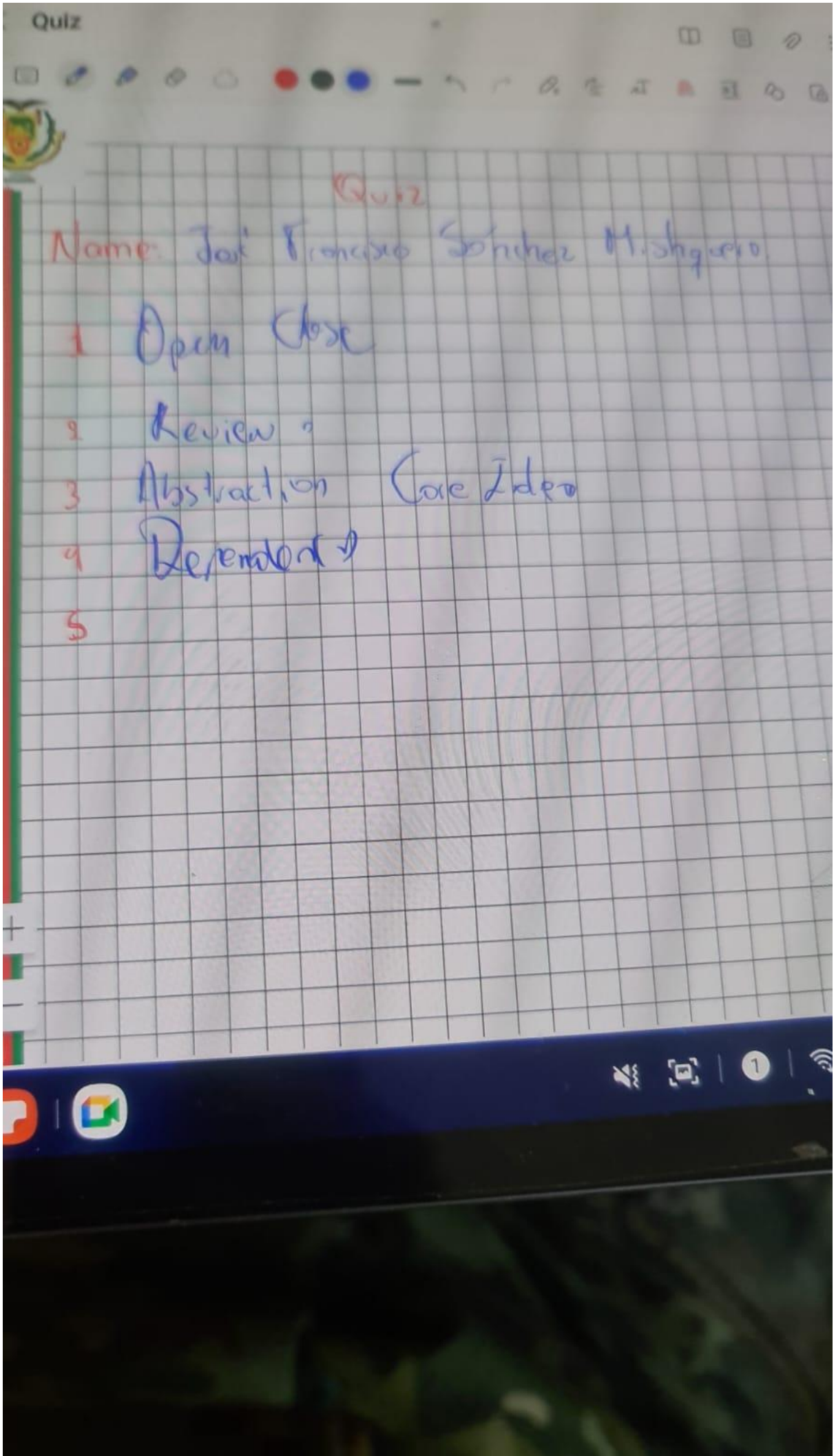
5 ~~agregation~~ interface

Jaime Eicor OOP 7490

1.) Open close principal
2) True.
3) interpuce Segregation principle
4) Dependency inversion
5) Interpuce Segregation principle
6)

RICHAR MAISINCHO
Open - close pricipal
True
Interface segregation
Dependencia inversion
Interface segregation




Quiz
Quiz
Name: José Francisco Sánchez Manzano
1 Open Close
2 Review
3 Abstraction Core Idea
4 Recursion
5

Name: Daniel Lincango

NRC: 7490

(1) Open/Close

(2) Liskon

(3) Interface

(4) Dependence, Abstract principle

(5) Single Responsability

DD MM AA

N. Benjamin Cadena

7490 - OOP

Thursday, February 3th 2022

1.-

Open, closed, principle

2.-

Review

3.-

Inteface segregation

4.-

Dependency Inversion

5.-

Single, Responsability, principle

Norma

NAME: JAVIER PAUCAR

1. OPEN / CLOSE PRINCIPLE
2. REVIEW
3. INTERFACE SEGREGATION
4. DEPENDENCY, ABSTRACT PRINCIPLE
5.

Name: Diego Sebastian Palacios Condo

1 Open / Close principal
2. Liskov segregation
3 Interface segregation
4. Dependency Inversión
5. Interface segregation

Name: Anderson Almache

NRC: 7490

Answers:

1. Open...

2. Yes

3. Interface Segregation

4. Dependency, Abstract

5. Single

Mateo Maldonado

- dWhat is the principles should be ready for funcionality

Single responsibility principle

- When we have an interillance the ~~father~~ children with ever the children need.
parent classes needit.

Dependency inversion

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

Interface segregation principle

- I have to organize the sistem in two

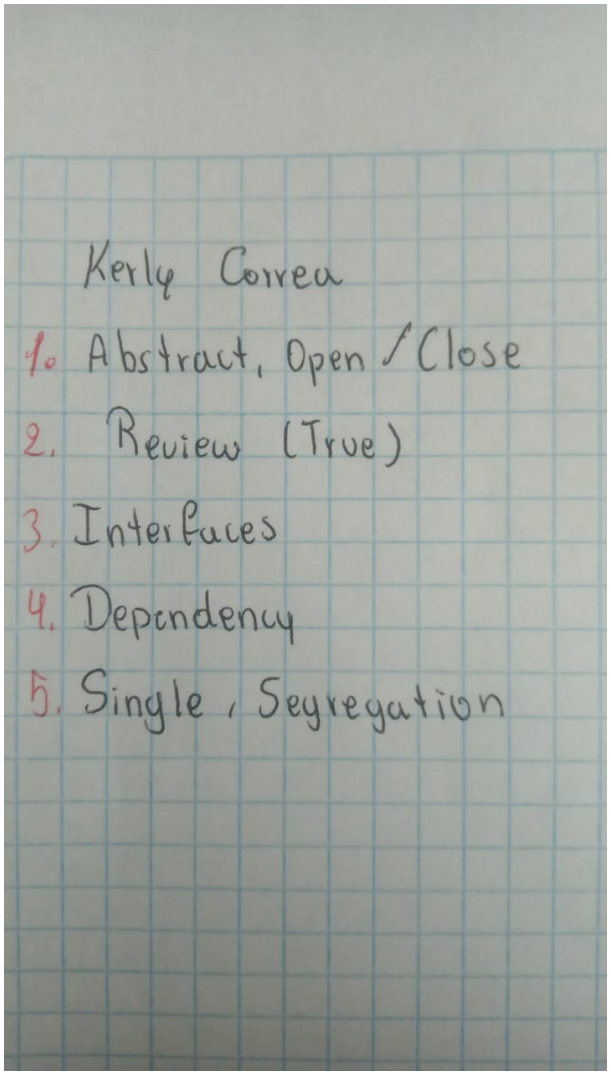
Dependency inversion principle

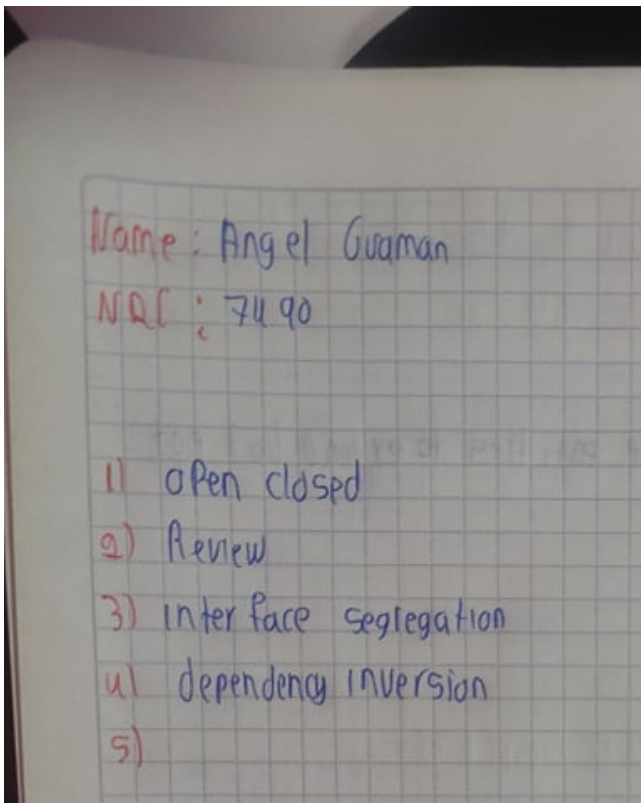
- This principle says that component do the muscle and no other action

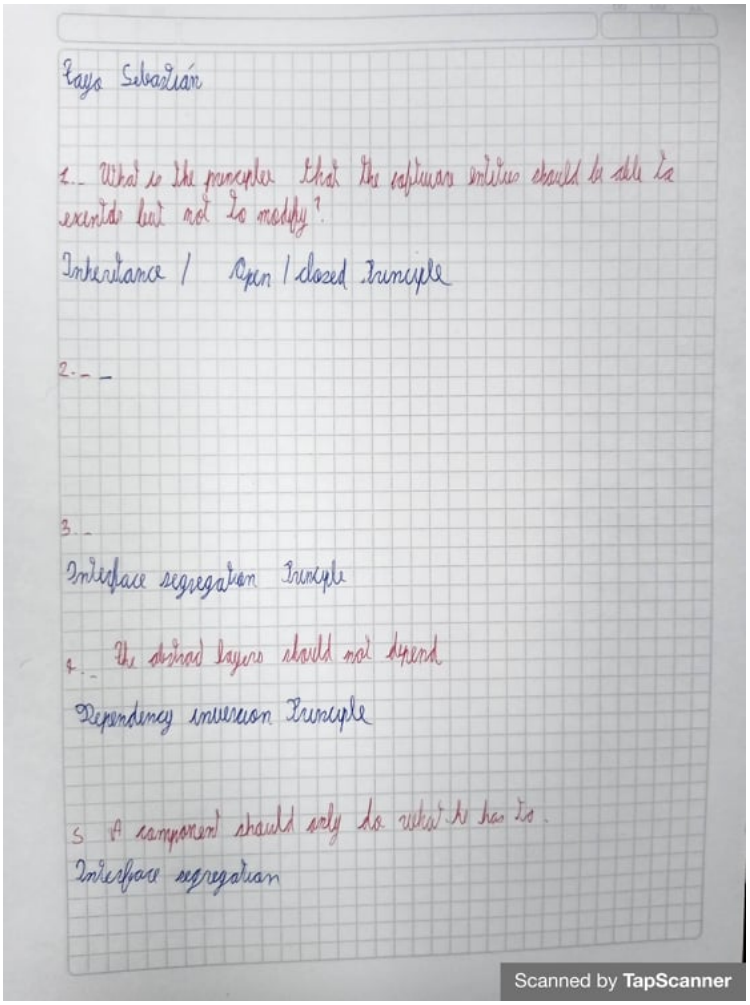
Interface segregation principle

Dylan Asumaza 7490 OPP

1. Open / Close principle
2. Review
3. Interface segregation principle
4. Dependency Abstract principle
5. Single responsability principle


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


Name: Angel Guaman
NRC: 7490
1) open closed
2) Review
3) interface segregation
4) dependency inversion
5)



Rayo Sebastián

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

Inheritance / Open / closed Principle

2.- -

3.-

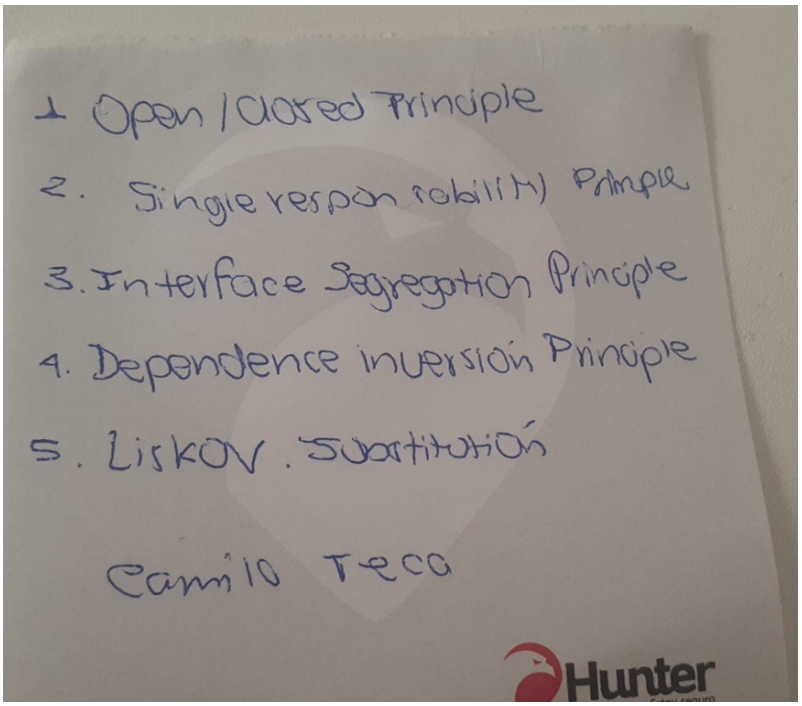
Interface segregation Principle

4.- The abstract layers should not depend

Dependency inversion Principle

5 A component should only do what it has to.

Interface segregation



1 Open / Closed Principle

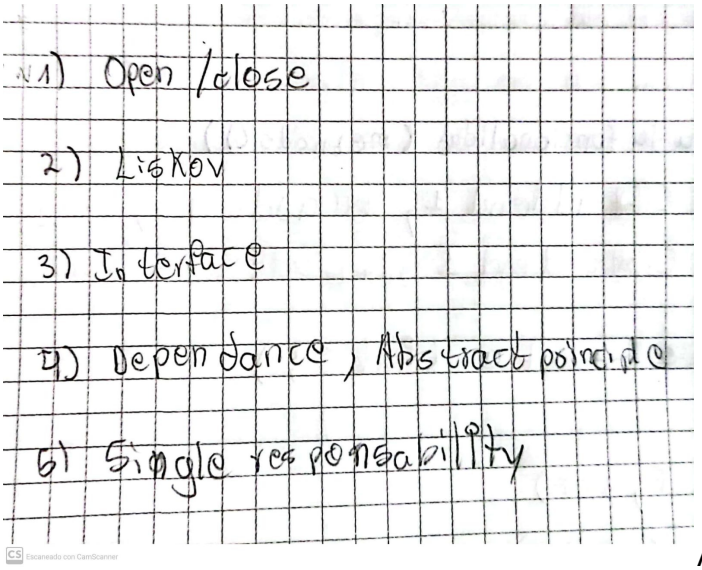
2. Single responsibility Principle

3. Interface Segregation Principle

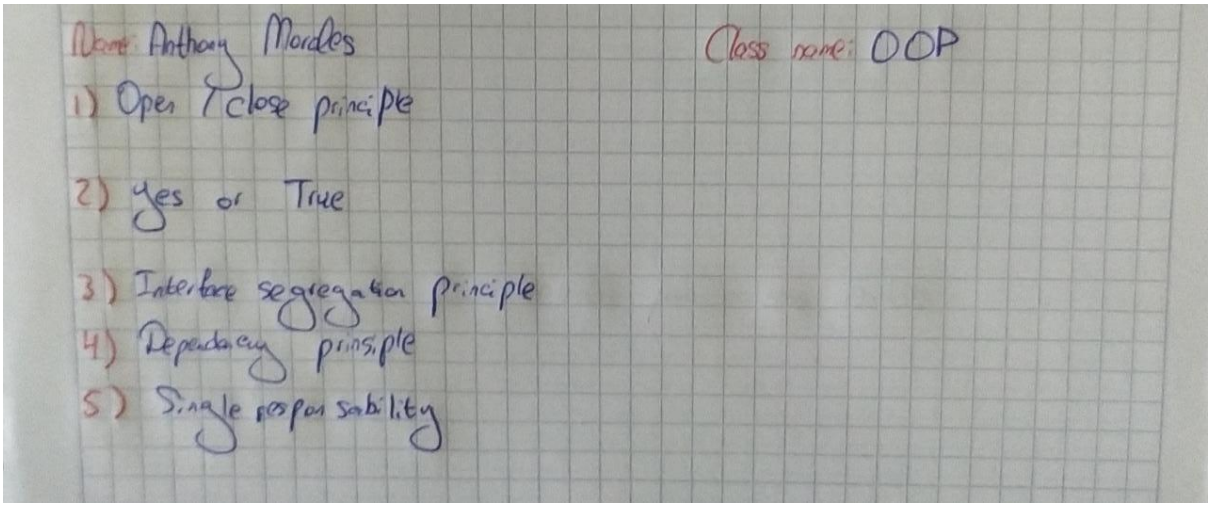
4. Dependence inversion Principle

5. Liskov. Substitution

Camilo Teco



Alexander Ruano



Cristian Arroba Nrc: 7490

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