

UNIVERSIDAD DE LAS FUERZAS ARMADAS

ESPE



Object-oriented programming

NRC: 9642

U2 Workshop N° 02

Topic: Code Inspection

ING. Jorge Edison Lascano.

Rubric:

- 1) class modeling 10 pts.
 - 2) Running program 10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10 pts.
 - 4) Saving JSON data 10pts.
 - 5) Code Quality 10 pts.
- TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram)
- 2) Running program (cmd line executing the program)
- 3) Printing/counting/deleting objects from JSON File (according to the roster number)
- 4) Saving JSON data (screenshot of the json file)
- 5) Code Quality (lines of code with stinky code)

1. RUBEN DARIO BENAVIDES MACIAS (Vehicles, print) /50

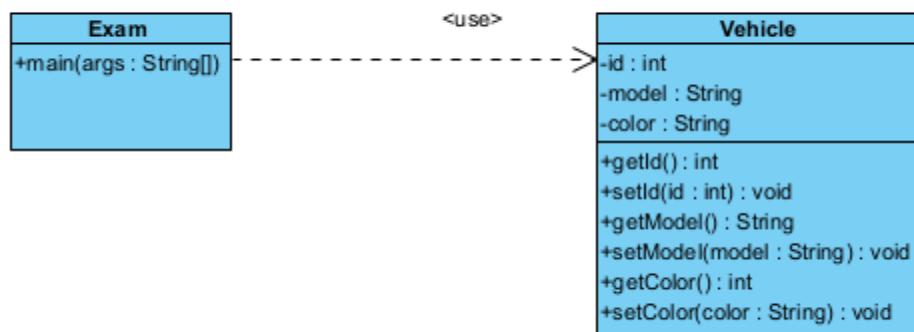
Inspector: Edison Verdesoto

Rubric:

- 1) class modeling 9/10 pts.
 - 2) Running program 10/10 pts.
 - 3) Printing/counting/deleting objects from JSON File 9/10 pts.
 - 4) Saving JSON data 10/10pts.
 - 5) Code Quality 6/10 pts.
- TOTAL: 44/50 pts.

Evidence

- 1) class modeling (Class diagram)



Main Class: 4/4

Model Class: 3/4 Missing the attribute

Dependency: 2/2

Total: 9/10

2) Running program 10/10 (cmd line executing the program)

```
Introduce the id for the vehicle 4:  
789  
Introduce the model for the vehicle 4:  
Audi  
Introduce the color for the vehicle 4:  
brown  
Introduce the id for the vehicle 5:  
7894565  
Introduce the model for the vehicle 5:  
Jeep  
Introduce the color for the vehicle 5:  
Yellow  
{
```

Showing option: 5/5

The screenshot shows an IDE interface with two main panes. The left pane displays a file tree with files: Vehicle.java, ec.edu.espe.exam1.view.Exam.java, Libraries, and gson-2.10.1.jar. Below the file tree is a terminal window titled 'Output - OOP-SW-ESPE-2023-9642-Exam1 (run)' showing the execution of the program. The right pane shows the generated JSON output.

```
[  
  {  
    "id": 256,  
    "model": "BMW",  
    "color": "Green"  
  },  
  {  
    "id": 5896,  
    "model": "Volkswagen",  
    "color": "brown"  
  },  
  {  
    "id": 123,  
    "model": "Chevrolet",  
    "color": "black"  
  },  
  {  
    "id": 789,  
    "model": "Audi",  
    "color": "brown"  
  },  
  {  
    "id": 7894565,  
    "model": "Jeep",  
    "color": "Yellow"  
  }]
```

Json Print: 5/5

Total:10/10

3) Printing/counting/deleting objects from JSON File
(according to the roster number)

```

1 * {
2     "id": 256,
3     "model": "BMW",
4     "color": "Green"
5 }
6
7 *
8     "id": 5896,
9     "model": "Volkswagen",
10    "color": "brown"
11 }
12
13 *
14     "id": 123,
15     "model": "Chevrolet",
16     "color": "black"
17 }
18
19 *
20     "id": 789,
21     "model": "Audi",
22     "color": "brown"
23 }
24

```

Support JSONLint for \$2/Month

Results

```

Error: Parse error on line 5:
... "color": "Green"}{ "id": 5896,      "mod
-----^
Expecting 'EOF', ']', ',', ']', got '{'

```

Validate Json: 9/10, error in line 5.

4) Saving JSON data

(screenshot of the json file)



```

*vehicles: Bloc de notas
Archivo Edición Formato Ver Ayuda
{
    "id": 256,
    "model": "BMW",
    "color": "Green"
}

{
    "id": 5896,
    "model": "Volkswagen",
    "color": "brown"
}

{
    "id": 123,
    "model": "Chevrolet",
    "color": "black"
}

{
    "id": 789,
    "model": "Audi",
    "color": "brown"
}

{
    "id": 7894565,
    "model": "Jeep",
    "color": "Yellow"
}
<

```

Json File: 10/10

5) Code Quality (lines of code with stinky code)

```
public class Exam {
    public static void main(String[] args) {

        ArrayList<Vehicle> vehicles = new ArrayList<Vehicle>();
        int id;
        String model;
        String color;

        for(int i = 0; i < 5; i++){

            Vehicle vehicle = new Vehicle();

            System.out.println("Introduce the id for the vehicle " + (i+1)+ ": " );
            while(true){
                try{
                    Scanner scanner = new Scanner(source: System.in);
                    id = scanner.nextInt();
                    break;
                }catch(Exception e){
                    System.out.println("Error. You just introduce wrong data. Try again: ");
                }
            }

            System.out.println("Introduce the model for the vehicle " + (i+1)+ ": " );
            while(true){
                try{
                    Scanner scanner = new Scanner(source: System.in);
                    model = scanner.nextLine();
                    break;
                }catch(Exception e){
                    System.out.println("Error. You just introduce wrong data. Try again: ");
                }
            }
        }
    }
}
```

Code Quality: 6/10. Everything is in the main class, no method called.

2. ADONNY MATEO CALERO ARGUELLO (Count computers)

0/50 Form wasn't answered.

Inspector: Ruben Benavides

3. PABLO ESTEBAN CARRERA QUINDE

Inspector: Adonny Calero

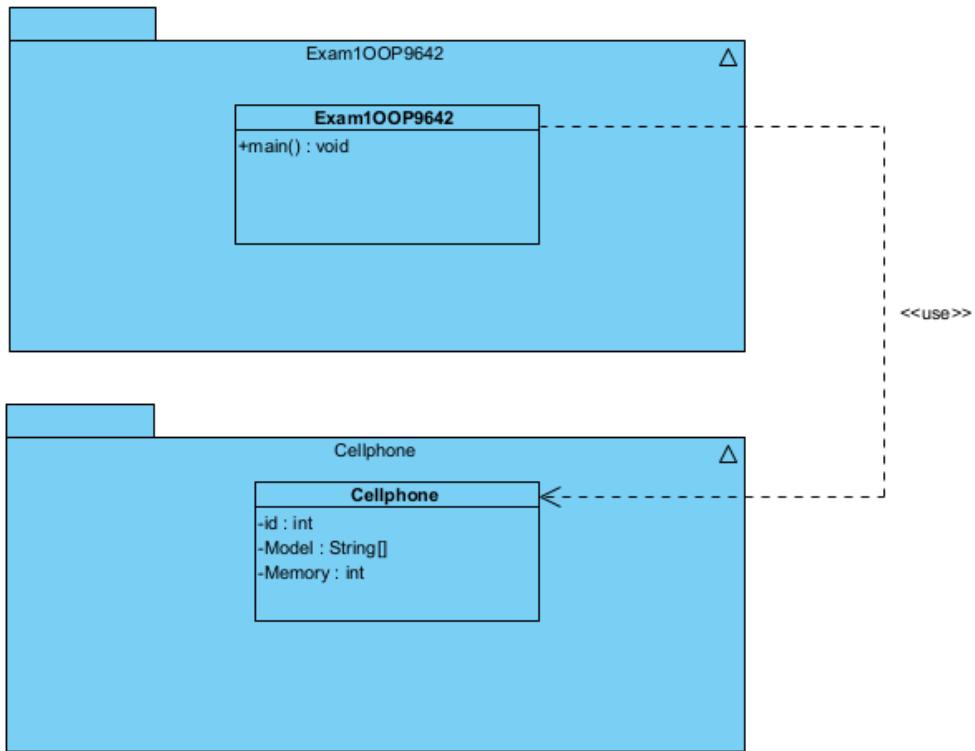
Action: Delete Cellphones

Rubric:

- 1) class modeling 10 pts.
 - 2) Running program 10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10 pts.
 - 4) Saving JSON data 10pts.
 - 5) Code Quality 10 pts.
- TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram)



Grade(main): 4/4

Grade(class): 3/4

Grade(Dependency) : 2/2

Total: 9/10

2) Running program (cmd line executing the program)

```

1 package e.edu.espe.model;
2
3 import java.io.FileWriter;
4 import java.io.IOException;
5 import com.google.gson.Gson;
6 import com.google.gson.GsonBuilder;
7 import java.util.ArrayList;
8
9 /**
10 * @author Pablo Carrera, Jsons, DCCO-ESPE
11 */
12
13 public class FileManager {
14     Gson gson = new GsonBuilder().setPrettyPrinting().create();
15     String json;
16
17     public void saveFile(ArrayList<Cellphone> cellphones, String fileName) {
18         Gson gson = new Gson();
19
20         String json = gson.toJson(cellphones);
21
22         try (FileWriter fileWriter = new FileWriter(fileName)) {
23             fileWriter.write(json);
24         } catch (IOException e) {
25             e.printStackTrace();
26         }
27     }
28 }

```

Output - CarreraPabloCellphoneExam1 (run)

```

run:
Unrecognized option: --module-path
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
C:\Users\Labes-DCCO\AppData\Local\NetBeans\Cache\17\executor-snippets\run.xml:11: The following error occurred while executing this line:
C:\Users\Labes-DCCO\AppData\Local\NetBeans\Cache\17\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 1 second)

```

Total: 0/10

3) Printing/counting/deleting objects from JSON File (according to the roster number)

```

[{"id": 10, "model": "A", "memory": 8}]

```

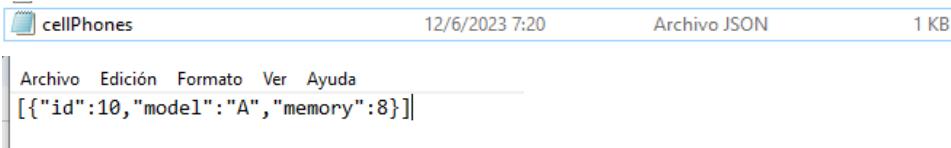
Validate JSON Clear

Results

Valid JSON

Validate Json: 10/10

4) Saving JSON data (Screenshot of the JSON file)



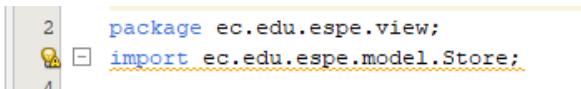
cellPhones 12/6/2023 7:20 Archivo JSON 1 KB

Archivo Edición Formato Ver Ayuda

```
[{"id":10,"model":"A","memory":8}]
```

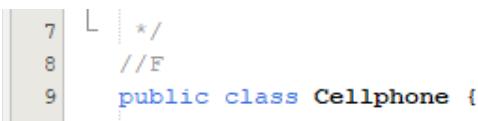
Total: 5/10 The program does not run.

5) Code Quality (Lines of code with stinky code)



```
2 package ec.edu.espe.view;
3 import ec.edu.espe.model.Store;
4
```

unnecessary headers



```
7 */
8 //F
9 public class Cellphone {
```

Unnecessary comments

Total: 8/10

Total Exam: 32/50

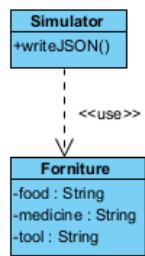
4. YESHUA AMADOR CHILIQUINGA AMAYA /10 - Print

Rubric:

- 1) class modeling /10 pts.
 - 2) Running program /10 pts.
 - 3) Printing/counting/deleting objects from JSON File /10 pts.
 - 4) Saving JSON data /10pts.
 - 5) Code Quality /10 pts.
- TOTAL: /50 pts.

Evidence

1) class modeling (Class diagram)



Main Class : 4/4

Second Class: 3/4 - no methods

Dependency: 2/2

Total: 9/10

2) Running program

(cmd line executing the program)

The screenshot shows the Android Studio interface with the following details:

- Projects** tab: Shows the project structure with packages like `ec.edu.espe.examboo.model` and `ec.edu.espe.examboo.view`, and dependencies on `gson-2.10.1.jar` and `JDK 20 (Default)`.
- Files** tab: Selected file is `Simulator.java`.
- Services** tab: Not currently active.
- Code Editor**: Displays Java code for a `Simulator` class. The code handles user input for selecting a JSON file and printing its content. A tooltip "Toggle Rectangular Selection (Ctrl+Shift+R)" is visible over the code editor.
- Output** tab: Shows the run log output for the application.

```
int selection;
System.out.println("Select the options\n");
System.out.println("1\n2\n3\n4\n5\n6\n7\n8\n9\n0");
selection=sc.nextInt();
if(selection==1){
    writeJSON();
}
else{
    if(selection==2){
        String jsonFilePath = "furniture.json";
        try (BufferedReader reader = new BufferedReader(new FileReader(new File(jsonFilePath)))) {
            StringBuilder jsonContent = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                jsonContent.append(line);
            }
            // Mostrar el contenido del archivo JSON
            System.out.println(jsonContent.toString());
        }
    }
}
```

```
run:
Welcome
Select the options
1
2
```

2.1 showing option 5/5 pts.

2.2 action with external modify 5/5

Total: 10/10

3) Printing/counting/deleting objects from JSON File
number)

JSONLint - The JSON Validator

Try the New Pro More Developer Tools

```
1 * {  
2     "food": "patata",  
3     "medicine": "paracetamol",  
4     "tool": "matillo"  
5 }
```

Validate JSON Clear Support JSONLint for \$2/Month

Results

Valid JSON

Json: 10/10

Total: 10

4) Saving JSON data (screenshot of the json file)

```

C: > Users > DELL > Desktop > EXAM1PracticalExerciseYeshuaChiliqinga > ExamPOO > {} forniture.json > ...
1  {
2    "food": "patata",
3    "medicine": "paracetamol",
4    "tool": "A"
5  };
6  {
7    "food": "papa",
8    "medicine": "sabutamol",
9    "tool": "B"
10 };
11 {
12   "food": "zanahoria",
13   "medicine": "ibuprofeno",
14   "tool": "C"
15 };
16 {
17   "food": "carne",
18   "medicine": "Icterix",
19   "tool": "D"
20 }

```

Printing Json File

5) Code Quality (lines of code with stinky code)

```

/**
 *
 * @author Yeshua Chiliqinga, JSons; DCCO-ESPE
 */

public class Forniture {
    private String food;
    private String medicine;
    private String tool;

    public Forniture(String food, String medicine, String tool) {
        this.food = food;
        this.medicine = medicine;
        this.tool = tool;
    }

    public String getFood() {
        return food;
    }

    public void setFood(String food) {
        this.food = food;
    }
}

```

Class Furniture: Everything is OK

Class Simulator: 5 pt

```

/*
public class Simulator {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(source: System.in);
        int selection;
        System.out.println("Welcom");
        System.out.println("Select the options\n"
            + "1\n"
            + "2");
        selection=sc.nextInt();
        if(selection==1){
            writeJSON();
        }else{
            if(selection==2){
                String jsonFilePath = "furniture.json";

                try (BufferedReader reader = new BufferedReader(new FileReader(fileName: jsonFilePath))) {
                    StringBuilder jsonContent = new StringBuilder();
                    String line;
                    while ((line = reader.readLine()) != null) {
                        jsonContent.append(str:line);
                    }

                    // Mostrar el contenido del archivo JSON
                    System.out.println(jsonContent.toString());
                } catch (IOException e) {
                    System.out.println("Error al leer el archivo JSON: " + e.getMessage());
                }
            }
        }
    }
}

```

The class contains: Everything the Jsonwrite and everything in the same class

Total:

Inspector: Pablo Carrera

5. PAMELA NAOMI CHIPE ZAMBRANO (Apartmets) 46/50

Inspector: Yeshua Chiliquinga

Rubric:

1) class modeling 10 pts.

Class name: 4/4

Class topic: 4/4

Relations: 2/2

2) Running program 10 pts.

Run program: 5/5

Run option: 5/5

3) Printing/counting/deleting objects from JSON File 10 pts.

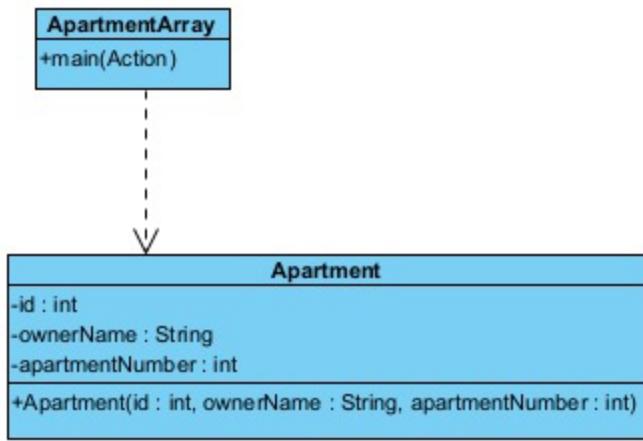
4) Saving JSON data 10pts.

5) Code Quality 10 pts.

TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram) No use void print



Main class : 4

Name class 1

Id 1

Variables 2

Use 2

Total: 10

2) Running program (cmd line executing the program) 10/10

3) Printing/counting/deleting objects from JSON File (according to the roster number) 8/5

4) Saving JSON data (screenshot of the json file) 10/10

Today			
	apartments	6/12/2023 8:08 AM	JSON Source File 1 KB
	manifest.mf	6/12/2023 1:24 AM	MF File 1 KB
	build.xml	6/12/2023 1:24 AM	xmlfile 4 KB
	test	6/12/2023 7:37 AM	File folder
	build	6/12/2023 1:26 AM	File folder
	src	6/12/2023 1:25 AM	File folder
	nbproject	6/12/2023 1:24 AM	File folder

yes, it has the json structure

5) Code Quality (lines of code with stinky code) 8/10

many spaces between code

6. JOAN OSWALDO COBEÑA ZAMBRANO (Sports) /50
Inspector: Pamela Chipe

Rubric:

1) class modeling 10/10 pts.

- Main class 4/4
- Object class 4 /4
- Dependency 2 /2

2) Running program 9/10 pts. run the code

- showing option 5/5 pts.
- action with external modify 4/5

3) Printing/counting/deleting objects from JSON File 10/ 10 pts.

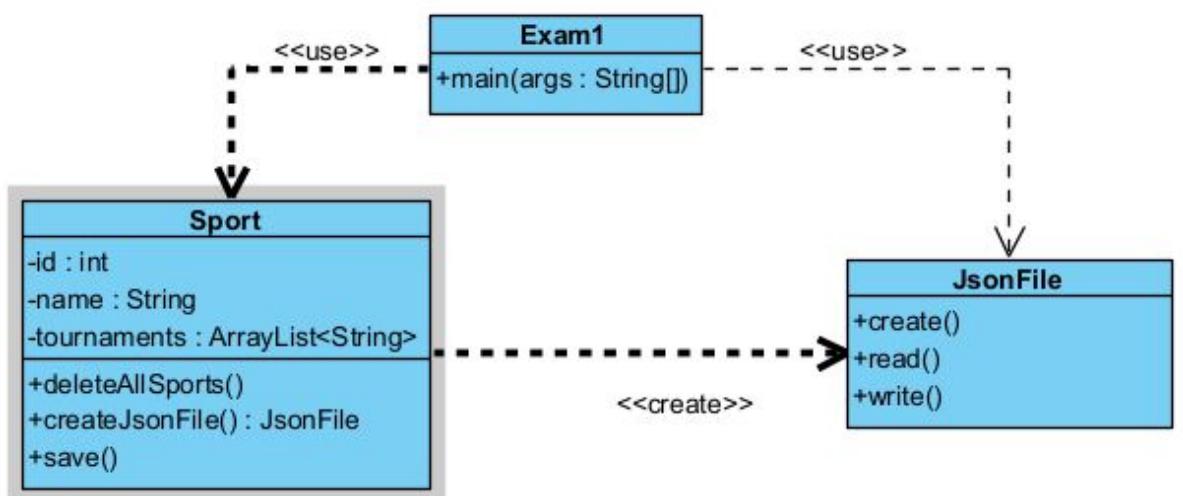
4) Saving JSON data 10 /10pts.

5) Code Quality 9/10 pts. in delete it doesn't work quite correctly

TOTAL:48/ 50 pts.

Evidence

1) class modeling (Class diagram)



2) Running program (cmd line executing the program)

```
run:
1. Data Input
2. Delete all
3. Exit
1
Input the sport id
```

3) Printing/counting/deleting objects from JSON File (according to the roster number)

```
1. Data Input
2. Delete all
3. Exit
1
Input the sport id
1
Input the sport name
Football
Input the sport tournament
Champions
1. Data Input
2. Delete all
3. Exit
```

4) Saving JSON data (Screenshot of the json file)

Nombre	Fecha de modificación	Tipo	Tamaño
build	10/06/2023 06:36 p. m.	Carpeta de archivos	
nbproject	10/06/2023 06:36 p. m.	Carpeta de archivos	
src	10/06/2023 06:36 p. m.	Carpeta de archivos	
build.xml	10/06/2023 06:36 p. m.	Archivo XML	4 KB
manifest.mf	10/06/2023 06:36 p. m.	Archivo MF	1 KB
Sports	10/06/2023 06:36 p. m.	Archivo de origen ...	1 KB

The screenshot shows a JSON editor interface. On the left, there is a code editor window displaying the following JSON array:

```
1 * [{"id": 132,
2   "name": "12321",
3   "tournament": "321"
4 }]
5 ]]
```

Below the code editor are two buttons: "Validate JSON" and "Clear".

On the right, there is a preview window titled "Sports" showing the JSON data:

```
[{"id":132,"name":"12321","tournament":"321"}]
```

At the bottom of the editor, there is a status bar with the following information: "Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8".

5) Code Quality (lines of code with stinky code)

7. ANABEL DAVILA MOLINA

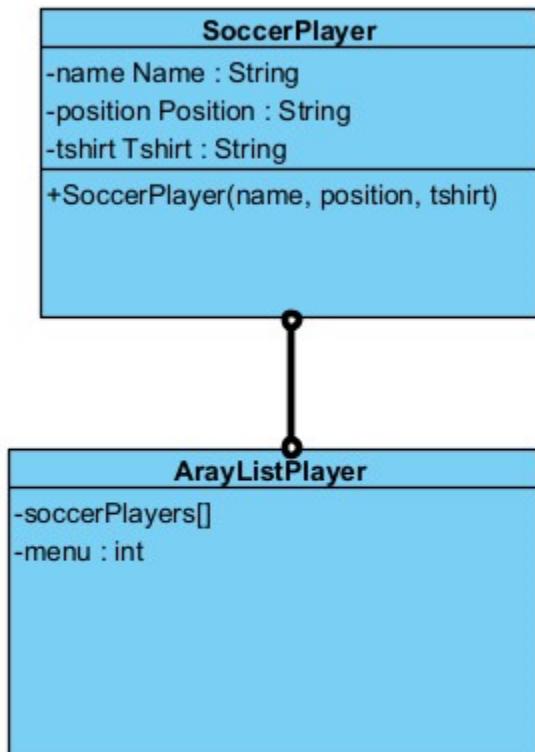
Print Soccer players

Rubric:

- 1) class modeling 7/10pts.
 - 1.1 main class $\frac{3}{4}$ pts.
 - 1.2 object class $\frac{3}{4}$ pts.
 - 1.3 dependency $\frac{1}{2}$ pts.
 - 2) Running program 6/10 pts.
 - 2.1 showing option 5/5 pts.
 - 2.2 action with external modify 1/5
 - 3) Printing/counting/deleting objects from JSON File 10/10 pts.
 - 4) Saving JSON data 9/10 pts. Diagram no-concordance.
 - 5) Code Quality 6/10 pts.
- TOTAL: 38 pts.

Evidence

- 1) class modeling (Class diagram)



- 2) Running program (cmd line executing the program)

```
Select an option:
1. Add a player
2. Print the data base of the players
3. Print all the players
0. Out
```

The screenshot shows two windows side-by-side. On the left is the Apache NetBeans IDE 17 interface, with the title bar "ExamDavilaAnabel - Apache NetBeans IDE 17". The main area displays a Java file named "SoccerPlayer.java" with the following code:

```
players.add(player);
```

The output window below shows the result of the run:

```
layerData{id=3, playerName=Carlos, shirtNumber=3}, SoccerPlayerData{id=4, playerName=Roberto, shirtNumber=4}]
```

On the right is a Visual Studio Code window titled "players.json". The JSON file contains the following data:

```
17 [
18   {
19     "id": 4,
20     "playerName": "Roberto",
21     "shirtNumber": 4
22   },
23   {
24     "id": 6,
25     "playerName": "External test",
26     "shirtNumber": 7
27 }
```

Default creations and no modify.

3) Printing/counting/deleting objects from JSON File

(according to the roster number)

```
4
Add the name of the player
Test4
Add the shirt number of the player
4
Select an option:
1. Add a player
2. Print the data base of the players
3. Print all the players
0. Out
1
Add the id of the player
5
Add the name of the player
Test5
Add the shirt number of the player
5
Select an option:
1. Add a player
2. Print the data base of the players
3. Print all the players
0. Out
2
Data base file created
Select an option:
1. Add a player
2. Print the data base of the players
3. Print all the players
0. Out
3
```

```
SoccerPlayerData{id=2, playerName=Test2, shirtNumber=2}, SoccerPlayerData{id=3, playerName=Test3, shirtNumber=3}, SoccerPlayerData{id=4, playerName=Test4, shirtNumber=4}, SoccerPlayerData{id=5, playerName=Test5, shirtNumber=5}]
```

4) Saving JSON data (Screenshot of the json file)

```

1 [{ 
2   "id": 1,
3   "playerName": "Ramiro",
4   "shirtNumber": 1
5 },
6 {
7   "id": 2,
8   "playerName": "Juan",
9   "shirtNumber": 2
10 },
11 {
12   "id": 3,
13   "playerName": "Carlos",
14   "shirtNumber": 3
15 },
16 {
17   "id": 4,
18   "playerName": "Roberto",

```

Validate JSON Clear

Results

Valid JSON

JSONLint Partners

Check out their products!

players.json - Visual Studio Code

El modo restringido está destinado a la navegación segura por el código. Confíe en esta vent...

Administrar Más información

players.json x

```

1 [
2   {
3     "id": 1,
4     "playerName": "Ramiro",
5     "shirtNumber": 1
6   },
7   {
8     "id": 2,
9     "playerName": "Juan",
10    "shirtNumber": 2
11 },
12   {
13     "id": 3,
14     "playerName": "Carlos",
15     "shirtNumber": 3
16 },
17   {
18     "id": 4,
19     "playerName": "Roberto",
20     "shirtNumber": 4
21 }
22 ]

```

Modo restringido 0 △ 0 Lin. 22, col. 2 Espacios: 2 UTF-8 LF JSON

5) Code Quality

(lines of code with stinky code)

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this li
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes.java to edit this template
 */

```

Unnecessary headers

```

/*
 * @author Anabel Davila,Kill Chain, DCOO-ESPE
 */
public class SoccerPlayer {

    //creamos un método para hacer el menú
    public void SoccerPlayerData(){

```

Unnecessary comments

```
public class SoccerPlayer {  
    //Creamos un método para hacer el menú  
    public void SoccerPlayerData(){  
  
        SoccerPlayerData player;  
        ArrayList<SoccerPlayerData> players = new ArrayList<>();  
  
        Scanner collectData = new Scanner(source: System.in);  
  
        player = new SoccerPlayerData (id: 1, playerName: "Ramiro", shirtNumber: 1);  
        players.add(e: player);  
        player = new SoccerPlayerData (id: 2, playerName: "Juan", shirtNumber: 2);  
        players.add(e: player);  
        player = new SoccerPlayerData (id: 3, playerName: "Carlos", shirtNumber: 3);  
        players.add(e: player);  
        player = new SoccerPlayerData (id: 4, playerName: "Roberto", shirtNumber: 4);  
        players.add(e: player);  
  
        int menu;  
  
        do{  
  
            System.out.println(x: " Select an option: ");  
            System.out.println(x: "1. Add a player ");  
            System.out.println(x: "2. Print the data base of the players");  
            System.out.println(x: "3. Print all the players");  
        }  
    }  
}
```

Menu options in the SoccerPlayer class, instead of main.

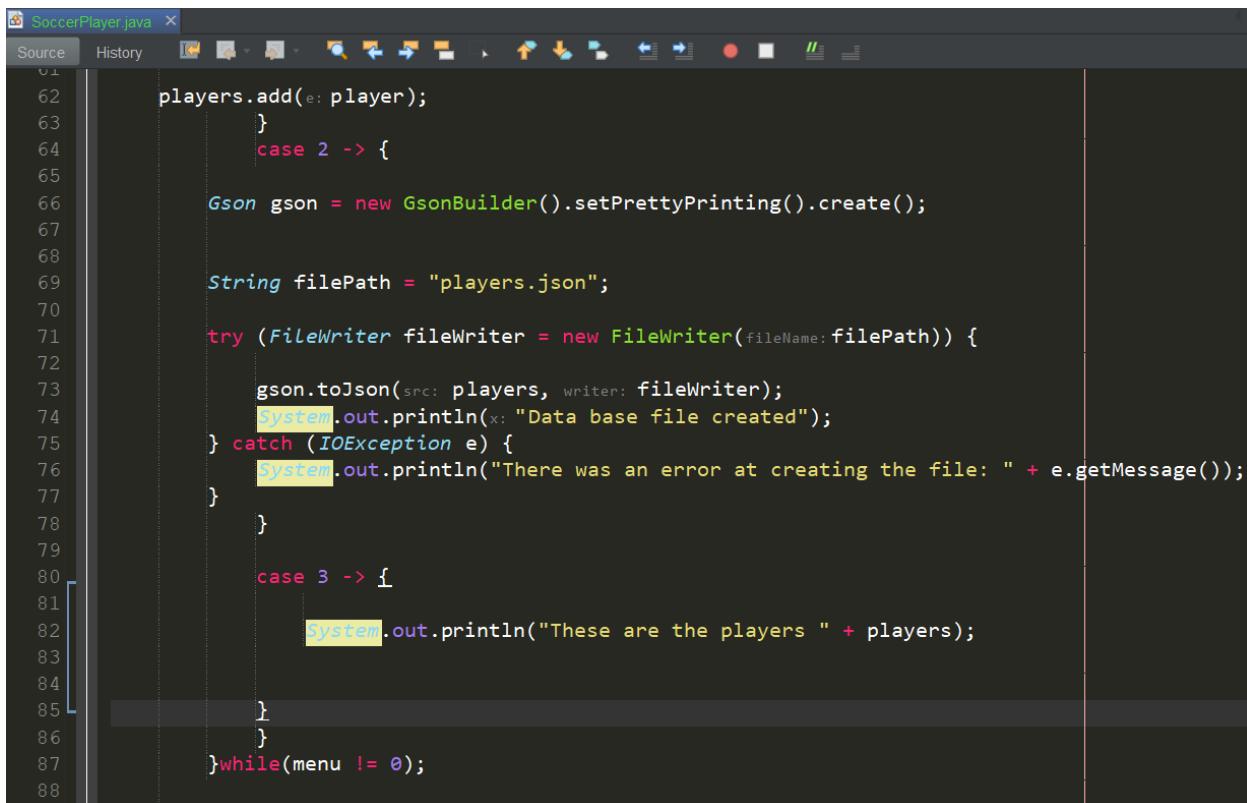
```
do{

    System.out.println(" Select an option: ");
    System.out.println("1. Add a player ");
    System.out.println("2. Print the data base of the players");
    System.out.println("3. Print all the players");
    System.out.println("0. Out");

    menu = collectData.nextInt();
    switch (menu){
        case 1 -> {
            System.out.println("Add the id of the player");
            int id = collectData.nextInt();
            System.out.println("Add the name of the player");
            String playerName = collectData.next();
            System.out.println("Add the shirt number of the player");
            int shirtNumber = collectData.nextInt();
            player = new SoccerPlayerData (id, playerName, shirtNumber);

            players.add(e: player);
        }
    }
}
```

Bad Indentation.



The screenshot shows the same Java code as the previous one, but with improved indentation. The code is now properly aligned, making it easier to read. The IDE interface includes tabs for 'Source' and 'History', and various toolbars and status bars.

```
62     players.add(e: player);
63
64     }
65     case 2 -> {
66
67         Gson gson = new GsonBuilder().setPrettyPrinting().create();
68
69         String filePath = "players.json";
70
71         try (FileWriter fileWriter = new FileWriter(fileName: filePath)) {
72
73             gson.toJson(src: players, writer: fileWriter);
74             System.out.println("Data base file created");
75         } catch (IOException e) {
76             System.out.println("There was an error at creating the file: " + e.getMessage());
77         }
78
79     case 3 -> {
80
81         System.out.println("These are the players " + players);
82
83     }
84
85     }
86
87 }while(menu != 0);
```

JSON File CRUD in the same SoccerPlayer class.

Inspector: Joan Cobeña

8. ANDRES ALEXANDER ESPIN ANDRADE

0/50 Forms wasn't answered

Rubric:

- 1) class modeling 10 pts.
 - 2) Running program 10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10 pts.
 - 4) Saving JSON data 10pts.
 - 5) Code Quality 10 pts.
- TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram)
- 2) Running program (cmd line executing the program)
- 3) Printing/counting/deleting objects from JSON File (according to the roster number)
- 4) Saving JSON data (screenshot of the json file)
- 5) Code Quality (lines of code with stinky code)

Inspector: Anabel Davila Molina

9. ALEJANDRO CAETANO FLORES MAYA University (delete)

Inspector: Espin Andres

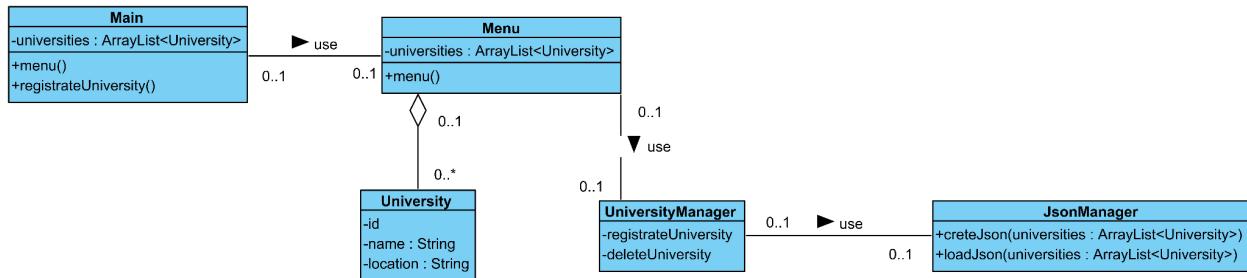
Rubric:

- 1) class modeling 10 pts.
 Class name: 4/4.
 Class topic: 4/4
 Relations: 2/2
- 2) Running program 10 pts.
 Run program: 5/5
 Run option: 5/5
- 3) Printing/counting/deleting objects from JSON File 10 pts.
 Json Validate: 10/10
- 4) Saving JSON data 10pts.
 Saving data 10/10
- 5) Code Quality 10 pts.

TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram)



2) Running program (cmd line executing the program)

```

run:
1.Enter university
2.Delete all universities
3.Close
1
Enter university id:
12
Enter university name:
saasdad
Enter university location:
qwdqweqw
University registered correctly
1.Enter university
2.Delete all universities
3.Close
2
All universities deleted correctly
1.Enter university
2.Delete all universities
3.Close

```

3) Printing/counting/deleting objects from JSON File (Delete)

The screenshot shows a JSONLint interface. On the left, there is a code editor containing the following JSON array:

```

1 * [
2   {
3     "id": 23423,
4     "name": "sadagfdgkdfjkvdf",
5     "location": "asdadadsada"
6   }
7 ]

```

Below the code editor are two buttons: "Validate JSON" and "Clear". To the right of the code editor is a blue button that says "Support JSONLint for \$2/Month". At the bottom, there is a "Results" section with a green bar indicating "Valid JSON".

4) Saving JSON data (screenshot of the json file)

```
[{"id":34354534,"name":"sdqwdqhosqoid","location":"ewfewfhyjyy"}, {"id":3453453,"name":"safsgfjukg","location":"dsadweerytut"}, {"id":322,"name":"asdafbfgjg","location":"adasda"}, {"id":2,"name":"asdsdfhf","location":"dsfrtyr"}, {"id":5,"name":"sdfdgjtjrt","location":"dgdfgd"}, {"id":9,"name":"fdsfdtry","location":"sdfre"}, {"id":454,"name":"sdfgtrt","location":"sdfdsf"}]
```

```
[]
```

5) Code Quality (lines of code with stinky code)

Clean code 9/10

TOTAL: 49/50 pts.

10. JORDAN ALEXANDER GUAMAN ALCIVAR - Count Buildings

Inspector: Caetano Flores

Rubric:

1) class modeling 5/10 pts.

Main class: 4/4.

Object Class: 0/4 pts.

Dependence: 1/2 pts. Association instead of dependency

- 2) Running program 10/10 pts.
 - 3) Printing/counting/deleting objects from JSON File 6/10 pts. List the items instead of cout it.
 - 4) Saving JSON data 9/10pts. Missing building name
 - 5) Code Quality 9/10 pts. Bad indentation on line 57 in “TenanPerson”
- TOTAL: 39/50 pts.

Evidence

- 1) class modeling (Class diagram)



- 2) Running program (cmd line executing the program)

```
Select an option:
1. Add a tenant:
2. Print the data base of the tenants in the build :
3. Print all the tenants
0. Out
2
Data base file created,OK
-----BUILDING SIMULATOR-----
Select an option:
1. Add a tenant:
2. Print the data base of the tenants in the build :
3. Print all the tenants
0. Out
3
These are the tenants: [TenantData(departmentNumber=23, tenantId=1, tenantName=null), TenantData(departmentNumber=76, tenantId=2, tenantName=null),
-----SIMULATOR-----
```

- 3) Printing/counting/deleting objects from JSON File (according to the roster number)

```
These are the tenants: [TenantData(departmentNumber=23, tenantId=1, tenantName=null), TenantData(departmentNumber=76, tenantId=2, tenantName=null),
```

4) Saving JSON data (screenshot of the json file)

The screenshot shows a JSON editor interface. At the top, there is a code editor window containing the following JSON array:

```
[{"departmentNumber": 23, "tenantId": 1}, {"departmentNumber": 76, "tenantId": 2}, {"departmentNumber": 45, "tenantId": 3}, {"departmentNumber": 56, "tenantId": 4}, {"departmentNumber": 10, "tenantId": 4}, {"departmentNumber": 10, "tenantId": 4}],
```

Below the code editor are two buttons: "Validate JSON" and "Clear". To the right of these buttons is a blue button that says "Support JSONLint for \$2/Month".

Underneath the buttons, the word "Results" is displayed. Below "Results" is a green bar containing the text "valid JSON".

5) Code Quality (lines of code with stinky code)

The screenshot shows a Java code editor with several sections of code highlighted in different colors (blue, green, yellow). The highlighted sections are labeled "stinky code". The code appears to be related to tenant management, specifically adding tenants to a list and handling a case statement.

```
tenant = new TenantData (tenantId: departmentNumber, tenantName: tenantName , departmentNumber: tenantId);  
tenants.add(e: tenant);  
  
tenants.add(e: tenant);  
}  
  
case 2 :{  
  
Gson gson = new GsonBuilder().create();  
String json = gson.toJson(tenants);  
System.out.println(json);  
}
```

11. CARLOS ANDRES JAYA HERRERA

0/50 Forms wasn't answered

Inspector: Jordan Guaman

Rubric:

- 1) class modeling 7/10 pts.
 - 2) Running program 5/10 pts.
 - 3) Printing/counting/deleting objects from JSON File /10 pts.
 - 4) Saving JSON data /10pts.
 - 5) Code Quality /10 pts.
- TOTAL: 50 pts.

12. BRYAN ALEXANDER JUMBO SALCEDO

18/50

Inspector: Carlos Jaya

Rubric:

- 1) class modeling 9 pts. Unused classes

```

1 package ec.edu.espe.Exercise;
2
3 /**
4 * Click phbs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
5 * Click phbs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
6 */
7
8 import java.util.ArrayList;
9 import java.util.Scanner;
10
11 /**
12 * @author Alexander J
13 */
14 public class Parks {
15     public void Save(){
16         }
17     }
18 }

```

Parks.java - Navigator x
Members <empty>
Parks
 Save()

2) Running program 0 pts. The program does not execute, nothing on the main class

```

1
2 /**
3 * @author Alexander J
4 */
5
6 public class Main {
7     public static void main(String[] args ) {
8
9     }
10 }

```

3) Printing/counting/deleting objects from JSON File 0 pts.

4) Saving JSON data 0 pts.

5) Code Quality 9 pts.Unused code

```

4
5     public void createBill(){
6
7 }
8
9 }

```

TOTAL: 0 pts.

13. JEFFREY ALEXANDER MANOBANDA CHABLA 0/50 Forms wasn't answered
Inspector: Bryan Jumbo

14. JHORDY PAUL MARCILLO MORA
Inspector: Jeffrey Alexander Manobanda Chabla
2) Count Banks

Rubric:

- 1) class modeling 8/10 pts.
- 2) Running program 7/10 pts. (No valid Json)

- 3) Printing/counting/deleting objects from JSON File 5/10 pts. (No valid Json)
 4) Saving JSON data 0/10pts. (No valid Json)
 5) Code Quality 9/10 pts. (The json file is in the main)
 TOTAL: 29 / 50 pts.

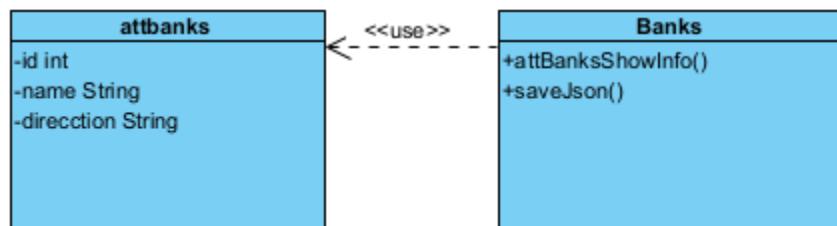
Evidence

- 1) class modeling (Class diagram)

Main class: 3/4.

Object Class: 3/4

Dependence: 2/2



- 2) Running program (cmd line executing the program)

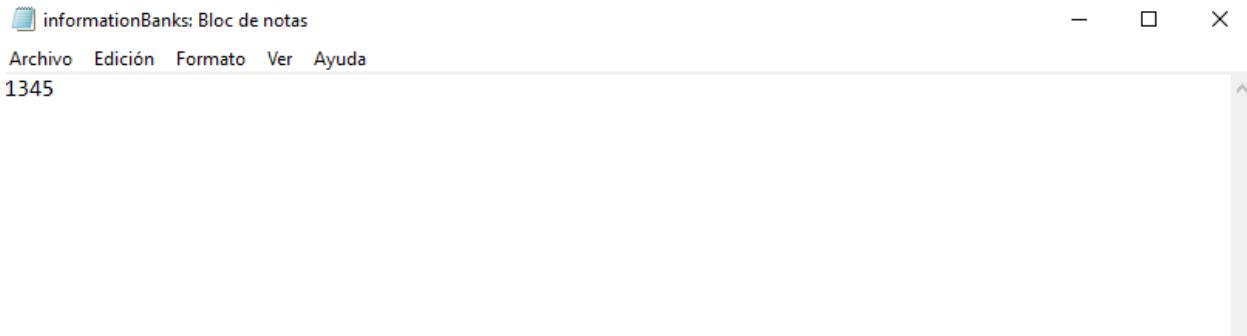
The screenshot shows a terminal window titled 'Output - Banks (run) #2'. The output of the program is displayed:

```

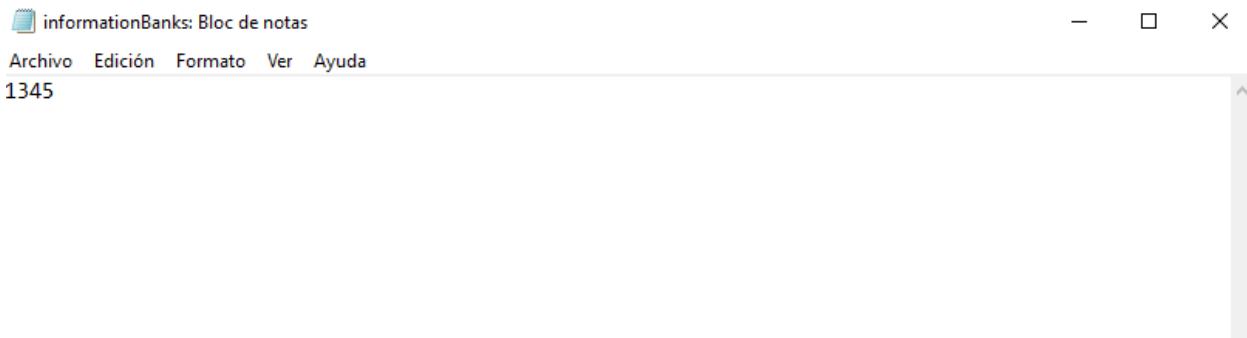
run:
enter ID
1345
enter name
enter dirección
san juab
File was saved
BUILD SUCCESSFUL (total time: 1 minute 41 seconds)
  
```

The terminal window has a standard OS X-style interface with a title bar, scroll bars, and a menu bar at the bottom with options like Archivo, Edición, Formato, Ver, Ayuda.

- 3) Printing/counting/deleting objects from JSON File (according to the roster number)



4) Saving JSON data (screenshot of the json file)



5) Code Quality (lines of code with stinky code)

```

private static void attbanksShowInfo(){
    attbanks Attbanks;
    Scanner scan;

    scan = new Scanner(source System.in);

    int id;
    String name;
    String direction;
    ArrayList<String> informationBanks;
    informationBanks = new ArrayList<>();

    System.out.println("enter ID");
    id = scan.nextInt();

    System.out.println("enter name");
    name = scan.nextLine();

    System.out.println("enter direction");
    direction = scan.nextLine();

    Gson gson = new Gson();
    String json = gson.toJson(id);

    try (FileWriter fw = new FileWriter(fileName: "informationBanks.json")) {
        fw.write(json);
        System.out.println("File was saved");
    } catch (Exception e) {
        System.out.println("error: file not open or found");
    }
}

```

15. JOSUE ISAAC MARIN ALQUINGA(Student) (Delete objects)

Inspector: Jhordy Marcillo

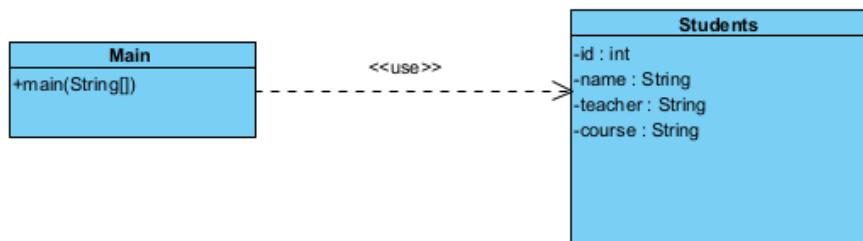
Rubric:

- 1) class modeling 7/10 pts.
- 2) Running program 10/10 pts.
- 3) Printing/counting/deleting objects from JSON File 10 /10 pts.
- 4) Saving JSON data 8/10pts.

5) Code Quality 7/10 pts.
TOTAL: 42/50 pts.

Evidence

1) Class modeling (Class diagram)



Wrong name, No methods

2) Running program

The screenshot shows the NetBeans IDE interface with several tabs open: Start Page, FarmSimulator.java, Banks.java, Main.java, and Student.java. The Main.java tab contains the following code:

```
private static void displayStudents() {
    System.out.println("-----LISTA DE ESTUDIANTES-----");
    if (students.isEmpty()) {
        System.out.println("No hay estudiantes registrados.");
    } else {
        for (Student student : students) {
            System.out.println(student);
        }
    }
}
```

The Output window shows the application's execution:

```
-----LISTA DE ESTUDIANTES-----
No hay estudiantes registrados.
```

The Navigator window shows the Main.java file with the 'loadStudentsFromFile()' method highlighted. The code for this method is:

```
public void loadStudentsFromFile() {
    Scanner scanner = new Scanner(new File(FILE_NAME));
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        String[] tokens = line.split(",");
        Student student = new Student(tokens[0], tokens[1], tokens[2], tokens[3]);
        students.add(student);
    }
    scanner.close();
}
```

The terminal window shows the application's execution again:

```
-----ESTUDIANTES-----
1. Agregar estudiante
2. Eliminar estudiante
3. Mostrar estudiantes
4. Salir
Ingresé su opción: 1

AGREGAR ESTUDIANTE
Ingresé el ID del estudiante: 212
Ingresé el nombre del estudiante: juan
Ingresé el profesor del estudiante: edison
Ingresé el curso del estudiante: 221
Estudiante agregado correctamente.

-----ESTUDIANTES-----
1. Agregar estudiante
2. Eliminar estudiante
3. Mostrar estudiantes
4. Salir
Ingresé su opción: 3

LISTA DE ESTUDIANTES
ec.edu.espe.student.model.Student@7d4793a8
ec.edu.espe.student.model.Student@445b2a27
```

In the program everything runs correctly

3) Printing/counting/deleting objects from JSON File

The screenshot shows a browser window with multiple tabs open. The active tab is "JSONLint - The JSON Validator" at <https://jsonlint.com>. The JSON code entered is:

```

1+ [{ 
2     "id": 322,
3     "name": "juan",
4     "teacher": "edison",
5     "course": "2"
6 }]

```

Below the code, there are "Validate JSON" and "Clear" buttons. The "Results" section shows a green bar indicating "valid JSON". A blue button at the bottom right says "Support JSONLint for \$2/Month".

All attributes in json :)

4) Saving JSON data

The screenshot shows the Apache NetBeans IDE 15 interface. The project "StudentsExam1" is selected. In the "Main.java" file, the following code is present:

```

private static void loadStudentsFromFile() {
    File file = new File("FILE_NAME");
    try (Scanner scanner = new Scanner(file)) {
        ...
    }
}

```

The "Output - StudentsExam1 (run)" window shows the application's interaction with the user:

```

Ingresue su opcion: 3

LISTA DE ESTUDIANTES
ec.edu.espe.student.model.Student@7d4793a0
ec.edu.espe.student.model.Student@445b2d27
ec.edu.espe.student.model.Student@5479e3f
ec.edu.espe.student.model.Student@27082746
ec.edu.espe.student.model.Student@66133adc

MENU:
1. Agregar estudiante
2. Eliminar estudiante
3. Mostrar estudiantes
4. Salir

Ingresue su opcion: 2

ELIMINAR ESTUDIANTE
Ingresue el ID del estudiante a eliminar: 212
Estudiante eliminado correctamente.

```

Display is not displayed correctly

5) Code Quality

```

    82     students.add( student );
    83     System.out.println( "Estudiante agregado correctamente." );
    84   }
    85
    86   private static void removeStudent(Scanner scanner) {
    87     System.out.println( "\nELIMINAR ESTUDIANTE" );
    88     System.out.print( "Ingrese el ID del estudiante a eliminar: " );
    89     int id = scanner.nextInt();
    90     scanner.nextLine();
    91
    92     boolean found = false;
    93     for (int i = 0; i < students.size(); i++) {
    94       if (students.get( index ).getId() == id) {
    95         students.remove( index );
    96         found = true;
    97         System.out.println( "Estudiante eliminado correctamente." );
    98         break;
    99       }
    100
    101     if (!found) {
    102       System.out.println( "No se encontró ningún estudiante con el ID especificado." );
    103     }
    104   }
    105
    106   private static void displayStudents() {
    107     System.out.println( "\nLISTA DE ESTUDIANTES" );
    108     if (students.isEmpty()) {
    109       System.out.println( "No hay estudiantes registrados." );
    110     } else {
    111       for (Student student : students) {
    112         System.out.println( student );
    113       }
    114     }
    115   }
    116
    117 }

    main - Navigator ×
    Members <empty>
    Main
      Main0
        addStudent[Scanner scanner]
        displayStudents()
        loadStudentsFromFile()
        main[String]args
        removeStudent[Scanner scanner]
        saveStudentToFile()
        FILE_NAME : String
        students : List<Student>

    Output - StudentsExam1 [run] ×
    Ingrese el curso del estudiante: 32
    Estudiante agregada correctamente.

    StudentsExam1 (run) running... 27/16 INIS Windows (CRLF)
    8:25 12/6/2023
  
```

Everything is in the same class

16. MESIAS ORLANDO MARISCAL OÑA (Print All objects)

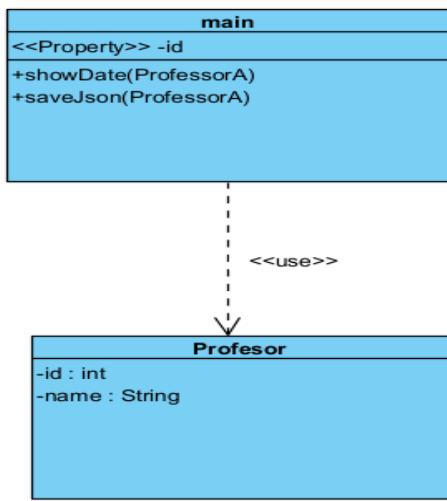
Inspector:Josue Isaac Marin Alquinga

Rubric:

- 1) class modeling **7/10 pts.**
 - 2) Running program **10/10 pts.**
 - 3) Printing/counting/deleting objects from JSON File **0/10 pts.**
 - 4) Saving JSON data **0/10pts.**
 - 5) Code Quality **8/10 pts.**
- TOTAL: **25/50 pts.**

Evidence

- 1) class modeling (Class diagram)



Name with lowercase. (3/4)

has no attributes (2/4)

Dependence(2/2)

Total:7/10

2) Running program (cmd line executing the program)

The screenshot shows a Java IDE interface with several tabs at the top: 'Output - Professor (run)', 'Start Page', 'info.java', 'ProfessorA.java', and 'createArray.java'. The 'Output - Professor (run)' tab contains the following command-line output:

```

run:
Enter the number of data entries:
3
Enter ID:
6
Enter name:
234t4rfds
Enter subject:
josue
Enter ID:
9
Enter name:
josue
Enter subject:
marin
Enter ID:
dcosjos
Enter name:

```

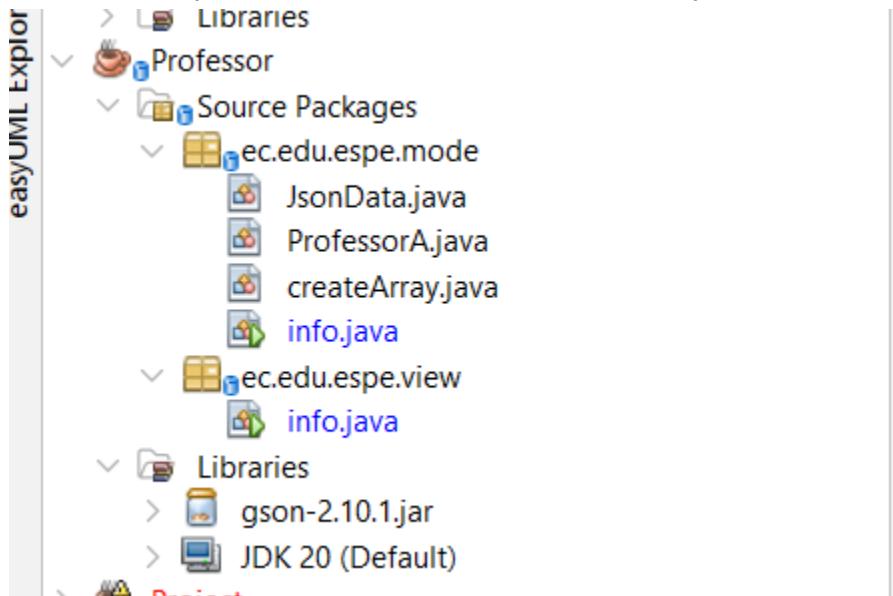
Run the program: 10/10

3) Printing/counting/deleting objects from JSON File (according to the roster number)
does not have JSON 0/10

4) Saving JSON data (screenshot of the json file)

0/10

5) Code Quality (lines of code with stinky code)



Unnecessary classes

Unnecessary comments

8/10

17. ANTHONY ALAIN MORALES CARVAJAL (add
Inspector: Mesias Mariscal

Rubric:

1) class modeling 7/10
2) Running program 5/10 pts.

3) Printing/counting/deleting objects from JSON File 0/10

No json

4) Saving JSON data 0 pts.

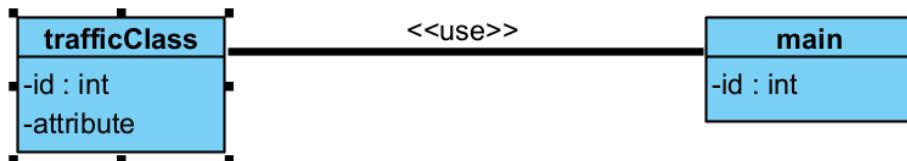
No file

5) Code Quality 10 /10

TOTAL: 22 pts.

Evidence

1) class modeling (Class diagram)



Name with lowercase. (3/4)

has no attributes (2/2)

Dependence(2/2)

Total: 7

2) Running program (cmd line executing the program)

TrafficExam - Apache NetBeans IDE 17

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Project... x Debugging Files Services

TrafficExam

Source Packages

- ec.edu.espe.traffic.controller
- ec.edu.espe.traffic.model
- TrafficCar.java
- TrafficMenu.java
- ec.edu.espe.traffic.view
- TrafficMain.java

Test Packages

Libraries

Test Libraries

TrafficCar.java x TrafficMain.java x TrafficMenu.java x

```

7  /*
8   *
9   * @author Anthony Morales, The_FAMSE
10  */
11 public class TrafficMenu {
12
13     public static void menu(){
14         Scanner scanner = new Scanner(source: System.in);
15
16
17         boolean out = false;
18         int option;
19
20
21         while(!out){
22             System.out.println("\tTraffic Choise 2");
23
24             option = scanner.nextInt();
25
26             switch(option){
27                 case 1:
28                     Enter Information
29                     break;
30
31                 case 2:
32                     Count the total number of stored objects
33                     break;
34
35                 case 3:
36                     Out
37                     break;
38
39                 default:
40                     Enter your choise:
41
42                     break;
43             }
44
45             if(option == 3)
46                 out = true;
47         }
48     }
49
50 }

```

Output - TrafficExam (run) x

```

run:
    Traffic Choise 2
    1. Enter Information
    2. Count the total number of stored objects
    3. Out
    Enter your choise:

```

<No View Available>

TrafficExam (run) running... x 1:1 INS

Option 1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Apache NetBeans IDE 17

Search (Ctrl+I)

Project... x Debugging Files Services

Start Page x TrafficMenu.java x TrafficMain.java x TrafficCar.java x

TrafficExam

Source Packages

Test Packages

Libraries

Test Libraries

TrafficCar.java x TrafficMain.java x TrafficMenu.java x

```

1 package ec.edu.espe.traffic.model;
2
3 import java.util.ArrayList;
4
5 /**
6  *
7  * @author Anthony Morales, The_FAMSE
8  */
9 public class TrafficCar {
10     private int id;
11     private String brand;
12     private String color;
13     private ArrayList<TrafficMenu> traffic;
14
15     public TrafficCar(int id, String brand, String color) {
16         this.id = id;
17     }
18
19     public int getId() {
20         return id;
21     }
22
23     public void setId(int id) {
24         this.id = id;
25     }
26
27     public String getBrand() {
28         return brand;
29     }
30
31     public void setBrand(String brand) {
32         this.brand = brand;
33     }
34
35     public String getColor() {
36         return color;
37     }
38
39     public void setColor(String color) {
40         this.color = color;
41     }
42
43     public ArrayList<TrafficMenu> getTraffic() {
44         return traffic;
45     }
46
47     public void setTraffic(ArrayList<TrafficMenu> traffic) {
48         this.traffic = traffic;
49     }
50
51     @Override
52     public String toString() {
53         return "TrafficCar{" +
54                 "id=" + id +
55                 ", brand='" + brand + '\'' +
56                 ", color='" + color + '\'' +
57                 '}';
58     }
59
60 }

```

TrafficCar.java - Navigator x

Members <empty>

TrafficCar

- id: int
- brand: String
- color: String
- traffic: ArrayList<TrafficMenu>
- TrafficCar(int id, String brand, String color)
- toString(): String ↗ Object
- getId(): int
- setId(int id)
- getBrand(): String

Output - TrafficExam (run) x

```

1. Enter Information
2. Count the total number of stored objects
3. Out
Enter your choise:
2
    Traffic Choise 2
    1. Enter Information
    2. Count the total number of stored objects
    3. Out
    Enter your choise:
    1
    Enter your id:
    12
    Enter the Car Brand:
    Mazda
    Enter the color:
    Black

```

TrafficExam (run) running... x 1:1 INS

Option 2

The screenshot shows the Apache NetBeans IDE 17 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Apache NetBeans IDE 17. The toolbar has icons for file operations like Open, Save, and Print. The main workspace has tabs for Start Page, TrafficMenu.java, TrafficMain.java, and TrafficCar.java. The TrafficCar.java tab is active, displaying the following code:

```

1 package ec.edu.espe.traffic.model;
2
3 import java.util.ArrayList;
4
5 /**
6  * @author Anthony Morales, The_FAMSE
7 */
8 public class TrafficCar {
9     private int id;
10    private String brand;
11    private String color;
12    private ArrayList<TrafficMenu> traffic;
13
14    public TrafficCar(int id, String brand, String color) {
15        this.id = id;
16    }

```

The Navigator panel on the left shows the members of the TrafficCar class, including id, brand, color, traffic, and several constructor and accessor methods. The Output panel at the bottom shows the program's run log:

```

Output - TrafficExam (run) ×
1. enter information
2. Count the total number of stored objects
3. Out
Enter your choise:
2
Traffic Choise 2
1. Enter Information
2. Count the total number of stored objects
3. Out
Enter your choise:
1
Enter your id:
12
Enter the Car Brand:
Mazda
Enter the color:
Black

```

3) Printing/counting/deleting objects from JSON File (according to the roster number)

No json

4) Saving JSON data (screenshot of the json file)

No file

5) Code Quality (lines of code with stinky code)

18. LENIN DAVID MORAN PALOMO

0/50 Form wasn't answered. (Trucks/Delete)

Inspector: Anthony Morales

19. LEONARDO VINICIO NARVAEZ CRIOLLO

Inspector: Moran David

Rubric:

1) class modeling **0/10 pts.**

- The name of the main is not valid because it could be confused
- no diagram

2) Running program **10/10 pts.**

Evidence

```

1 /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package ec.edu.espe.smartwatches.view;
6
7 import ec.edu.espe.smartwatches.model.Menu;
8 import java.util.Scanner;
9
10 /**
11  * @author Narvaez Leonardo, The FANSE, DCCO Espe
12  */
13 public class SmartWatches {
14     public static void main(String[] args) {
15         Menu.showMenu();
16     }
17 }

```

Salida - Narvaez.Leonardo.CodingSkills (run) x

```

red
Chicken added to the coop.
Menu options:
1. Add a Smart Watch that you want.
2. Show the smart watch features.
3. Generate the file.
4. Exit.
2.
Smart Watch data:

```

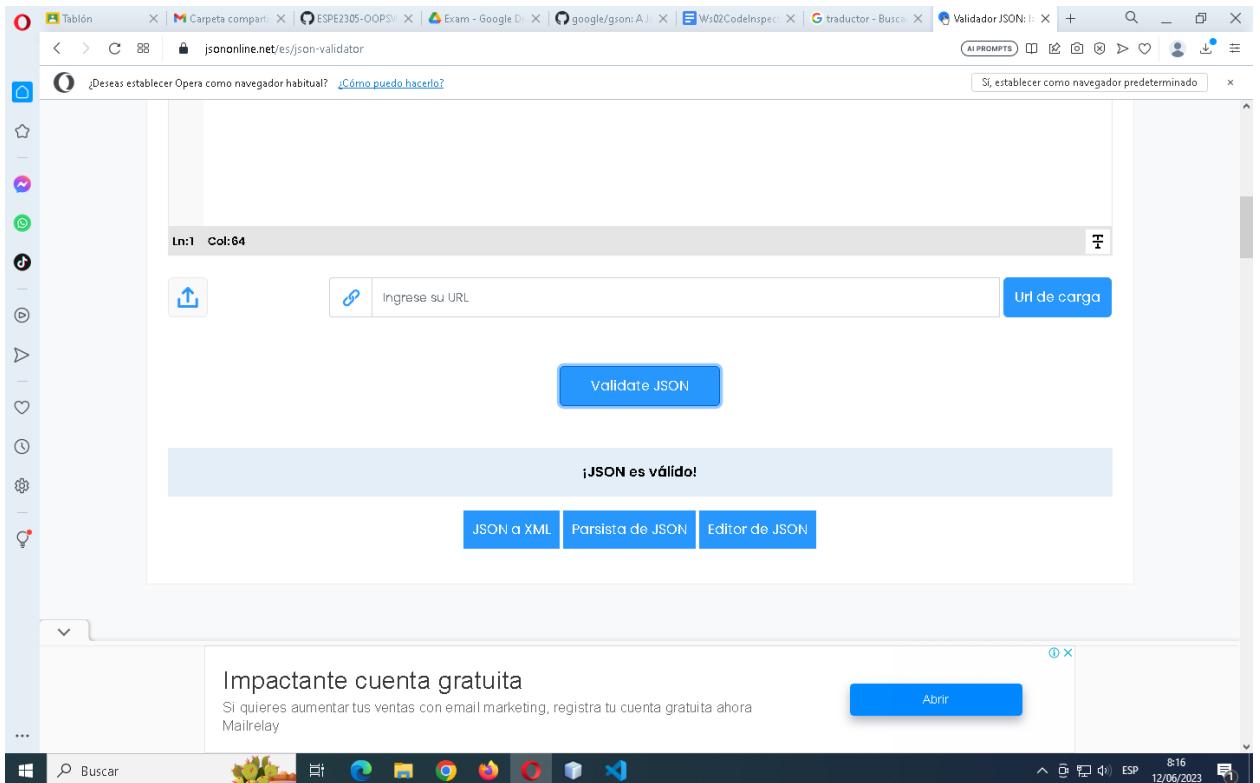
3) Printing/counting/deleting objects from JSON File **10/10 pts.**

Evidence

```

{
    "products": [
        {"id": 17, "color": "black", "mark": "xiaomi"}
    ]
}

```



4) Saving JSON data 4/10pts.

- generates data but not complete and with line break

5) Code Quality 8/10 pts.

Evidence

- unnecessary line of code
- unnecessary line of code and in revision of the rest is fine

```

public static void addSmartWatch(List list){
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter the ID:");
    int id = scanner.nextInt();
    System.out.println("Enter the mark that you want: ");
    String mark = scanner.next();
    System.out.println("Enter the Color: ");
    String color = scanner.next();
    Product smartWatch = new Product(id, color, name);
    list.getProducts().add(smartWatch);
    System.out.println("SmartWatch added to the coop.");
}

```

The code editor shows the Java code for the 'addSmartWatch' method. The code prompts the user for an ID, mark, and color, then creates a new 'Product' object and adds it to a list. The code is annotated with comments and includes imports for Scanner and Product.

TOTAL: 32/50 pts.

20. JOHN MATEO QUISHPE LLUMIQUINGA 0/10 Not sent to the form.

Inspector: Leonardo Narváez.

21. LEONEL ALEJANDRO TIPAN QUINTO (Singers, Delete objects)

Inspector: Leonardo Narváez.

Rubric:

- 1) class modeling 10 pts.
 - 2) Running program 10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10 pts.
 - 4) Saving JSON data 10pts.
 - 5) Code Quality 10 pts.
- TOTAL: 50 pts.

Evidence:

- 1) class modeling (Class diagram)
0/10. No class diagram uploaded.
- 2) Running program (cmd line executing the program)
The program runs very well: 10/10
Run program: 5/5
Run option: 5/5

```
Singer (run) * Singer (run) #2 *  
ID: 17  
Name: maria  
Age: 19  
-----  
1. List singers  
2. Delete all singers  
0. Exit  
Enter your choice: 2  
All singers deleted.  
1. List singers  
2. Delete all singers  
0. Exit  
Enter your choice:  
  
Enter your choice: 1  
ID: 15  
Name: leo  
Age: 18  
-----  
ID: 2  
Name: maria  
Age: 19  
-----  
1. List singers  
2. Delete all singers  
0. Exit  
Enter your choice:
```

```

Enter the number of singer objects to add: 2
Enter details for singer #1
ID: 15
Name: leo
Age: 18
Singer added successfully.
-----
Enter details for singer #2
ID: 2
Name: maria
Age: 19
Singer added successfully.

```

3) Printing/counting/deleting objects from JSON File (according to the roster number)

Json file is generated correctly: 10/10

The screenshot shows a JSON validation interface. On the left, there is a code editor containing the following JSON code:

```

1 var [
2   {
3     "id": 15,
4     "name": "leo",
5     "age": 18
6   },
7   {
8     "id": 2,
9     "name": "maria",
10    "age": 19
11  ]

```

Below the code editor are two buttons: "Validate JSON" and "Clear". To the right of these buttons is a blue button that says "Support JSONLint for \$2/Month".

Underneath the code editor, the word "Results" is displayed. A green bar indicates that the JSON is "Valid JSON".

At the bottom of the interface, there is a terminal window showing the path to the JSON file: "C: > Users > leona > Desktop > OOP Repository > ESPE2305-OOPSW9642 > exams > tipan > U1 > Question33 > singers.json". The terminal also shows the first line of the file: "1 [".

4) Saving JSON data (screenshot of the json file)

It updates correctly, but deletes everything, and it was only to delete one by one.
7/10.

The screenshot shows two tabs in Visual Studio Code. The top tab is titled 'singers.json' and contains the following JSON code:

```
[{"id": 1, "name": "Leone1", "age": 19}, {"id": 2, "name": "Maria1", "age": 20}, {"id": 3, "name": "Madonna", "age": 23}, {"id": 4, "name": "Robert", "age": 56}, {"id": 5, "name": "Selena", "age": 59}]
```

The bottom tab is also titled 'singers.json' and contains the following JSON code:

```
[]
```

5) Code Quality (lines of code with stinky code)

Unnecessary comments.

9/10.

```
public Singer(int id, String name, int age) {  
    this.id = id;  
    this.name = name;  
    this.age = age;  
}  
  
// Getters and Setters  
  
public int getId() {  
    return id;
```

Rubric:

- 1) class modeling 0/10 pts.
 - 2) Running program 10/10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10/10 pts.
 - 4) Saving JSON data 10/10pts.
 - 5) Code Quality 9/10 pts.
- TOTAL: 36/50 pts.**

22. EDISON DAMIAN VERDESOTO SEGOVIA 26 /50 (glasses, print)

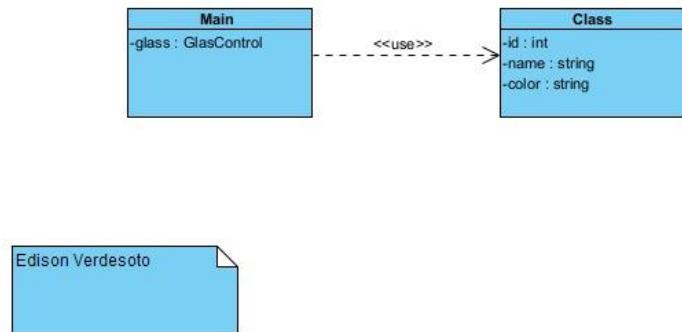
Inspector: Leonel Tipan

Rubric:

- 1) class modeling 10 pts.
 - 2) Running program 10 pts.
 - 3) Printing/counting/deleting objects from JSON File 10 pts.
 - 4) Saving JSON data 10pts.
 - 5) Code Quality 10 pts.
- TOTAL: 50 pts.

Evidence

- 1) class modeling (Class diagram) 7/10



Class name: 3/4

Model class: 2/4

Dependency: 2/2

- 2) Running program (cmd line executing the program) 5/10

```

run:
UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
Subject: Oriented Objects Programming
NRC: 9642
INSTRUCTOR: Edison Lascano
Practical Exam.
  
```

```

MENU
GLASSES INPUT
1. Add a glasses
2. Print the list
3. Exit
Choose an option
  
```

Showing Menu: 5/5

No json print: 0/5

3) Printing/counting/deleting objects from JSON File (according to the roster number) 0/10

No json file: 0/10

4) Saving JSON data (screenshot of the json file) 3/10

```
Scanner sc = new Scanner(source:System.in);
File file = new File (pathname: "./dataBase.json");
```

No Json file created, but exists in the program: 3/10

5) Code Quality (lines of code with stinky code) 10/10