Taller 1 Patron
Link:
https://onlinegdb.com/dt7Nbq3E8

RUBRICA PARA EVALUACIÒN	

Preguntas	***Puntos***	***Calificación***	***Observación***
1. La clase main, llama a la vista (View), y al controlador (Controler)	1		La clase main crea studentview y sudentcontroller y ejecuta start
2. Elaborar el modelo en base de datos (BD) al dado, únicamente se adaptando, añadiendo tres nuevos Constructores	1		Se creó la clase Student con dos atributos: name y rollNo. Además, se implementaron tres constructores: uno vacío, uno con parámetros y uno copia. Esto permite flexibilidad al crear objetos Student en diferentes contextos.
3. Para la Vista view crear un método de inserción, el cual simula cómo sería la inserción tradicional	1		InputStudent en StudentView sumula la insercion tradicional
4. El controlador se cambió en su mayoría, haciendo uso únicamente de los métodos que se requieren para hacer un intermediario entre el modelo, y la vista.	1		StudentController conecta StudentView con Student Database
5. finalmente, crear una clase que simula una base de datos, la cual brinda apoyo en la administración de los	1		StudentDatabase tiene 5 estudiantes predefinidos y métodos CRUD

datos quemados. (05 estudiantes).			
EJECUCION			
TOTAL	**5**	**/5**	

REQUISITOS:
Para cada RF realizar la revisión de código y explicar a través de la ejecución el funcionamiento de MVC

1. La clase main, llama al View, y al controlador

La clase Main crea una instancia de la vista (StudentView) y del controlador (StudentController), y luego llama al método start() del controlador. Esto inicia la ejecución del flujo MVC.

2. Se hizo el modelo en base al dado, únicamente se adaptando, añadiendo tres nuevos Constructores

Se creó la clase Student con dos atributos: name y rollNo. Además, se implementaron tres constructores: uno vacío, uno con parámetros y uno copia. Esto permite flexibilidad al crear objetos Student en diferentes contextos.

3. Para el view se creó un método de inserción, el cual simula cómo sería la inserción tradicional

La clase StudentView contiene el método inputStudent() que simula la entrada de datos de un estudiante con valores quemados (David, 1), representando una inserción tradicional. También tiene el método printStudentDetails() para mostrar los datos de cada estudiante.

4. El controlador se cambió en su mayoría, haciendo uso únicamente de los métodos que se requieren para hacer un intermediario entre el modelo, y la vista

El controlador actúa como intermediario entre la vista y la base de datos. Tiene métodos para mostrar estudiantes (fetchStudents), insertar un nuevo estudiante (createStudent) y actualizar un estudiante existente (updateStudent).

5. se creó una clase que simula una base de datos, la cual brinda apoyo en la administración de los datos quemados.

Se creó una clase que simula una base de datos con 5 estudiantes quemados. Esta clase permite agregar, actualizar y eliminar estudiantes mediante métodos como postStudent(), putStudent() y deleteStudent().

Ejecucion del codigo/ Pantallas

```
public class Main {

    public static void main(String[] args) {
        StudentView view = new StudentView();
        StudentController controller = new StudentController(view);
        controller.start();
    }
}
```

input

```
Student:
Name: Juan
Roll No: 11

Student:
Name: David
Roll No: 1


...Program finished with exit code 0
Press ENTER to exit console.
```

1. Run | Debug | Stop | Share | Save | { } Beautify

java | Student.java | StudentView.java | StudentController.j... | StudentDatabase.j...

```
/*****************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, R
C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLi
Code, Compile, Run and Debug online from anywhere in world.

*****************************************************************************/

public class Main {

    public static void main(String[] args) {
        StudentView view = new StudentView();
        StudentController controller = new StudentController(view);
        controller.start();
    }
}
```

2.
```
    public Student(String name, String rollNo) {
        esto. rollNo = rollNo;
        esto. nombre = nombre;
    }
```
3.
```
Student:
Name: Juan
Roll No: 11

Student:
Name: David
Roll No: 1


...Program finished with exit code 0
Press ENTER to exit console.
```
4.
```
**********Updating Data**********
Student:
Name: Jon
Roll No: 10

Student:
Name: Miguel
Roll No: 11

Student:
Name: Ana
Roll No: 12
```
5.
```
public class Student {

    private String rollNo;
    private String name;

    public Student() {
        this("", "");
    }

    public Student(String name, String rollNo) {
        this.rollNo = rollNo;
        this.name = name;
    }

    public Student(Student student) {
        this.rollNo = student.getRollNo();
        this.name = student.getRollNo();
    }
```