***Taller 1 Patron***

***Link:***

[***https://onlinegdb.com/dt7Nbq3E8***](https://onlinegdb.com/dt7Nbq3E8)

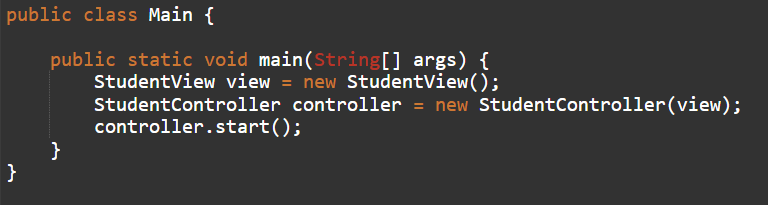
| ***RÚBRICA PARA EVALUACIÓN*** | | |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| ***Preguntas*** | ***Puntos*** | ***Calificación*** | ***Observación*** |
| 1. La clase main, llama a la vista (View), y al controlador (Controler) | 1 |  |  |
| 1. Elaborar el modelo en base de datos (BD) al dado, únicamente se adaptando, añadiendo tres nuevos Constructores | 1 |  |  |
| 1. Para la Vista view crear un método de inserción, el cual simula cómo sería la inserción tradicional | 1 |  |  |
| 1. 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 |  |  |
| 1. finalmente, crear una clase que simula una base de datos, la cual brinda apoyo en la administración de los datos quemados. (05 estudiantes). | 1 |  |  |
| EJECUCIÓN |  |  |  |
| **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**

En el código de la clase Main crea instancias tanto de la vista como del controlador, donde el controlador recibe la vista como argumento en su constructor, lo que permite interactuar con la vista y gestionar las operaciones entre el modelo y la vista, estos métodos se puede observar en la Figura 1, Figura 2 y Figura 3 de la ejecución.



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

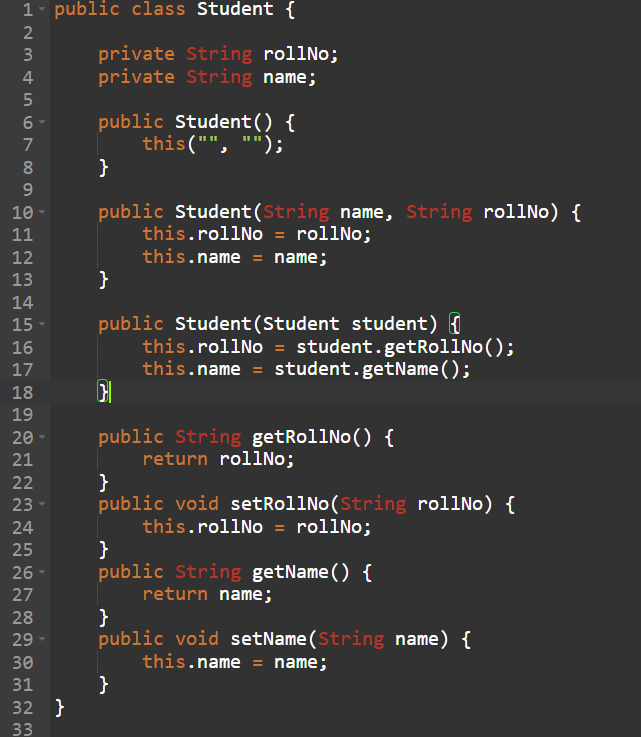
Si, el modelo incluye los tres constructores, sin embargo, en el constructor que requiere un Objeto tipo Student como parámetro (el tercer constructor), el campo del name se llena con el get del rollNo.



Por lo que se requeriría corregir el código para que use el getName del objeto parámetro



Este sería el resultado

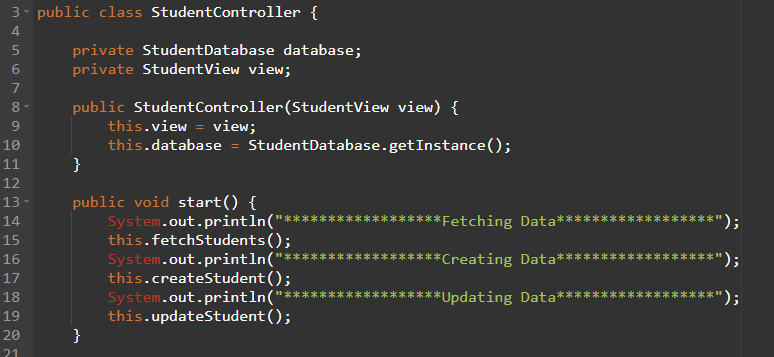


1. **Para el view se creó un método de inserción, el cual simula cómo sería la inserción tradicional**

Si, el método inputStudent() simula la inserción de un estudiante utilizando datos predeterminados. Este método emplea el segundo constructor de la clase Student.java y es llamado durante la creación del objeto en la clase StudentController.java, en la figura 2 se puede observar en la ejecución.

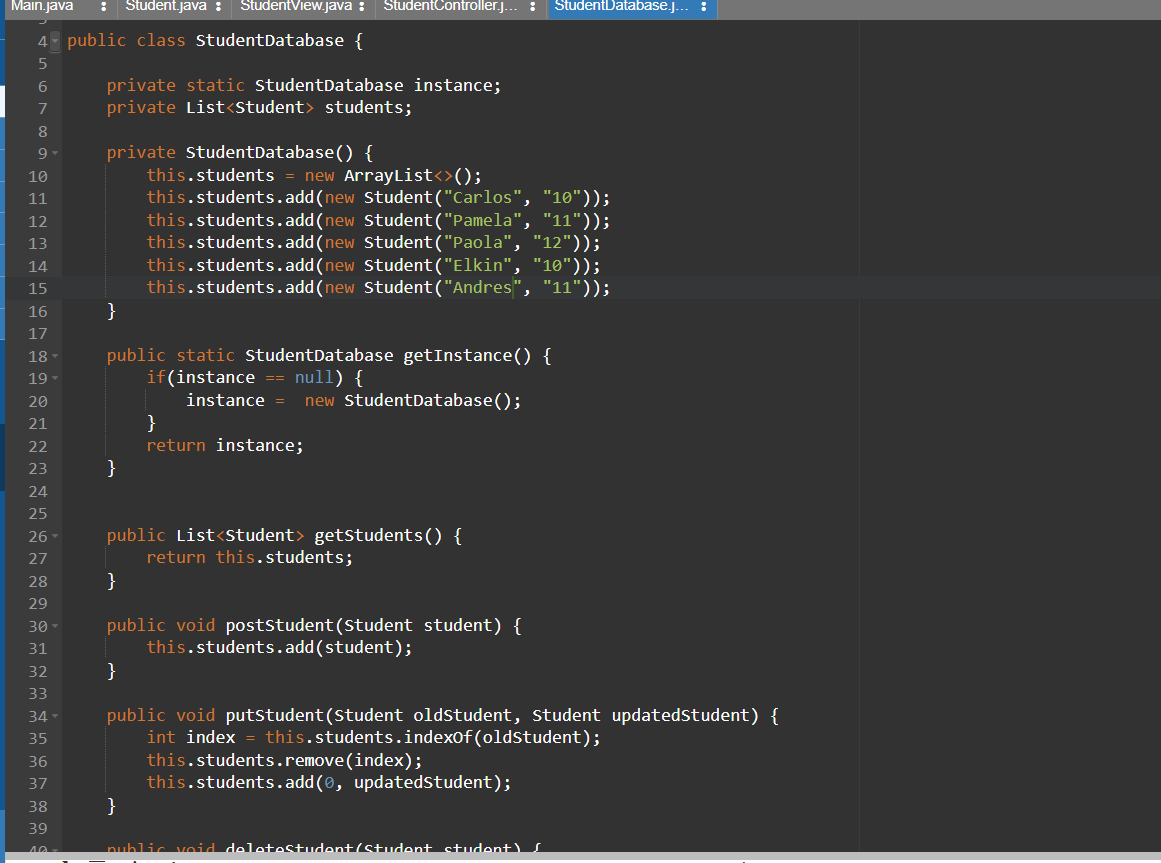


1. **El controlador se cambió en su mayoría, haciendo uso únicamente de los métodos que se requieren para hacer un intermediario entre el modelo, y la vista**  
   El controlador maneja las interacciones entre la vista y el modelo, la función principal consiste en llamar a métodos de la vista para simular la entrada de datos, interactúa con el modelo para obtener, modificar los datos y luego actualizarlos, además, esto lo realiza mediante los métodos fetchStudents(), createStudent(), y updateStudent()



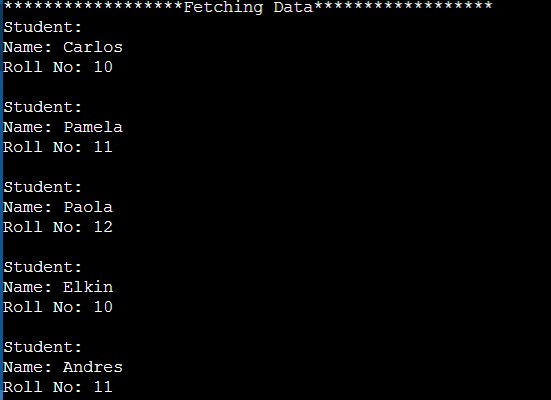
1. **Se creó una clase que simula una base de datos, la cual brinda apoyo en la administración de los datos quemados.**

Si, la clase StudentDatabase simula una base de datos utilizando una lista interna para almacenar los registros de estudiantes predefinidos. Esta clase incluye métodos como post() y delete(), que permiten simular la persistencia de datos. En la implementación, los estudiantes se cargan correctamente en la lista. Además, al insertar nuevos datos, la información se actualiza correctamente, lo que demuestra que la base de datos simulada puede gestionar los cambios en los registros.  
En la Figura 1 se muestra cómo la información predefinida de los 5 estudiantes está cargada, mientras que en la Figura 3 se evidencia que esta información se actualiza al realizar una nueva inserción.



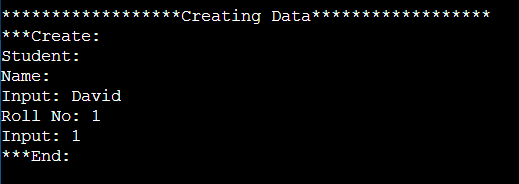
**Ejecucion del codigo/ Pantallas**

Esta parte muestra cómo la vista se relaciona con el controlador, que obtiene información de la base de datos y la vista muestra esta información en consola

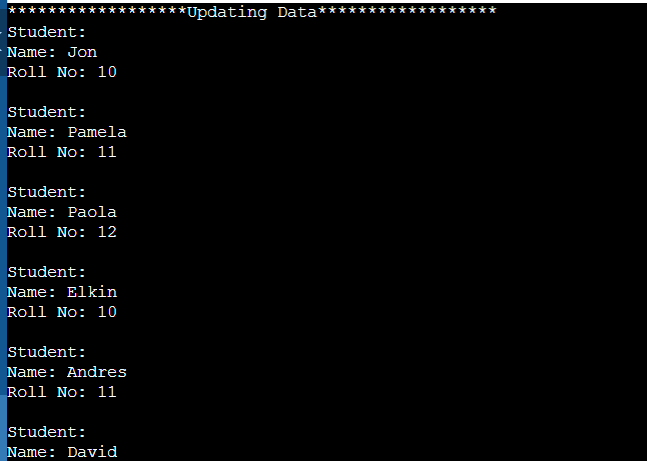
**

*Figura 1. Captura de ejecución (método fetchStudents).*

Aquí se evidencia el funcionamiento de método del view *InputStudent()* que crea un estudiante en la base de datos usando el segundo constructor del modelo *Student.java*

*  
Figura 2. Creación de un estudiante.*

El método updateView() del controller que obtiene la información de la base de datos, lo guarda en una lista y luego muestra la información de cada registro de la lista usando el método *view.printStudentDetails()* del view que da formato a la impresión de la información de los estudiantes

*  
Figura 3. Actualización de la información en la base de datos simulada.*