***Taller 1 Patron***

***Link:***

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

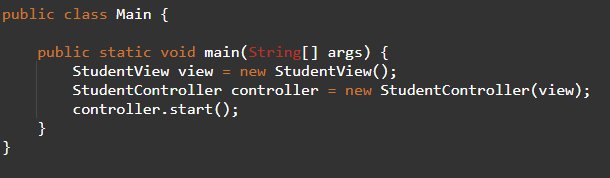
|  |  |  |  |
| --- | --- | --- | --- |
| ***RUBRICA PARA EVALUACIÒN*** | | |  |
|  |  |  |  |
|  |  |  |  |
| ***Preguntas*** | ***Puntos*** | ***Calificación*** | ***Observación*** |
| 1. La clase main, llama a la vista (View), y al controlador (Controler) | 1 |  |  |
| 1. Elaborar el modelo en base 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 |  |  |
| EJECUCION |  |  |  |
| **TOTAL** | **5** | **/5** |  |

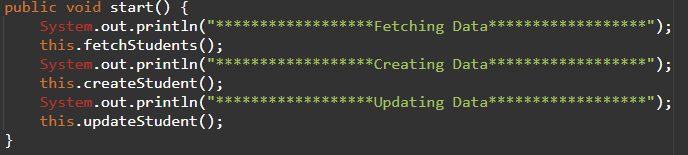
REQUISITOS:

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

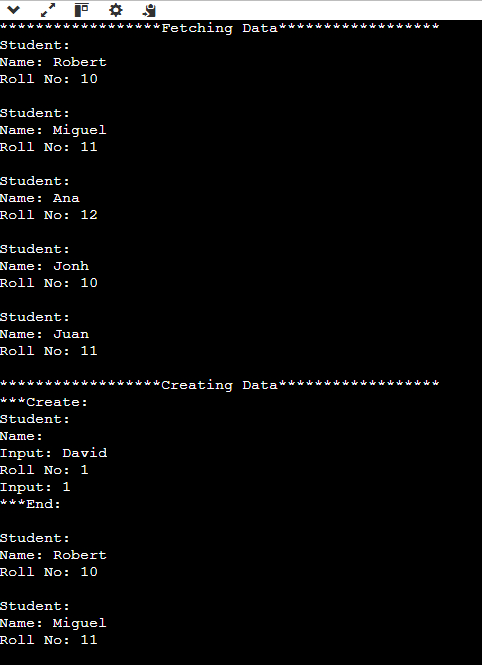
1. La clase main, llama al View, y al controlador

Codigo:





Ejecución:



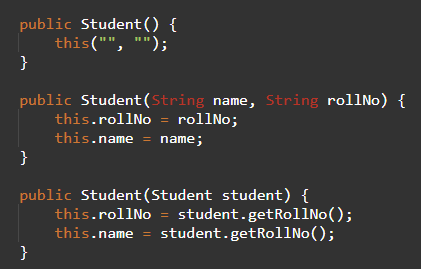
Explicación:

El programa inicia con la clase Main, donde instancia la clase StudentView y una instancia de la clase StudentController donde su constructor requiere un objeto de la clase StudentView.

El controlador llama a su método *start()* donde por pantalla se presenta la información de los estudiantes existentes en la base de datos. Se agrega automaticamente un nuevo estudiante y se muestra la información actualizada.

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

Código:



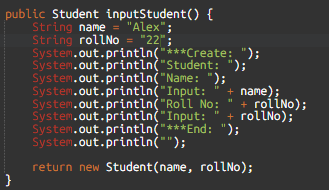
Ejecución:



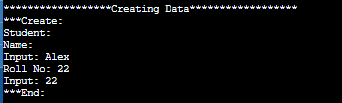
Explicación:

En la clase Student se sobrecargo 3 constructores, donde el primero crea una instancia de dicha clase con información vacía. El segundo constructor recibe por parámetros los datos de nombre y el número de rol del estudiante. El tercer constructor recibe como parámetro la instancia de otro objeto de la clase estudiante, donde se copian sus atributos (rol y nombre)

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

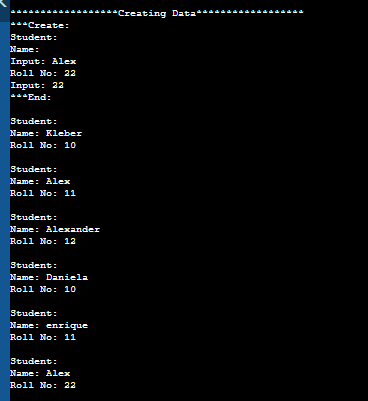


Ejecución:

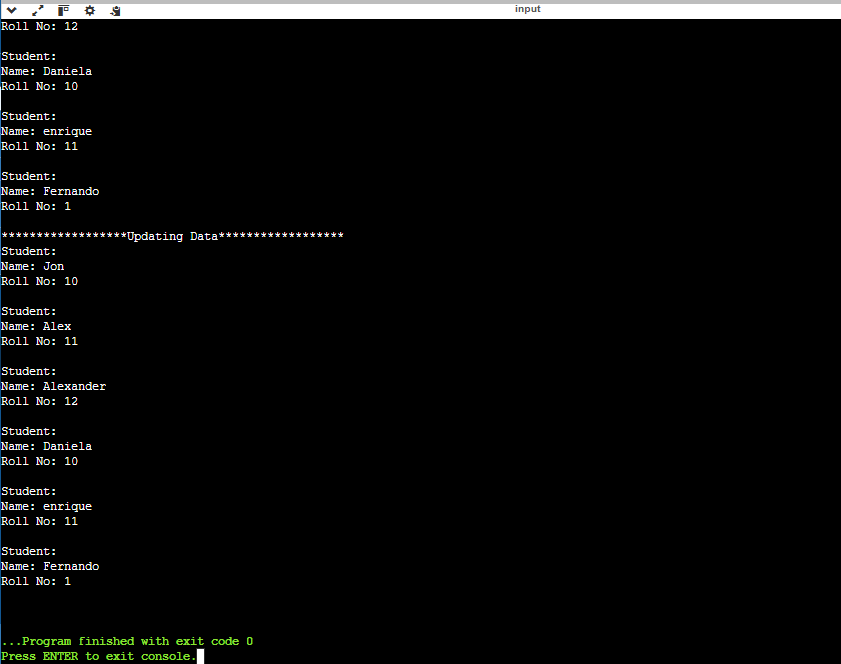
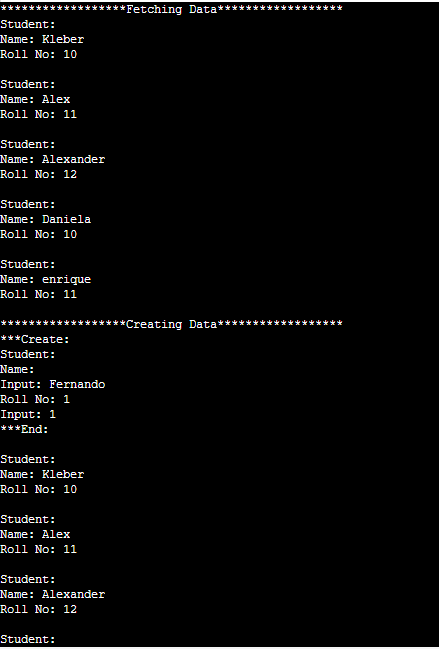
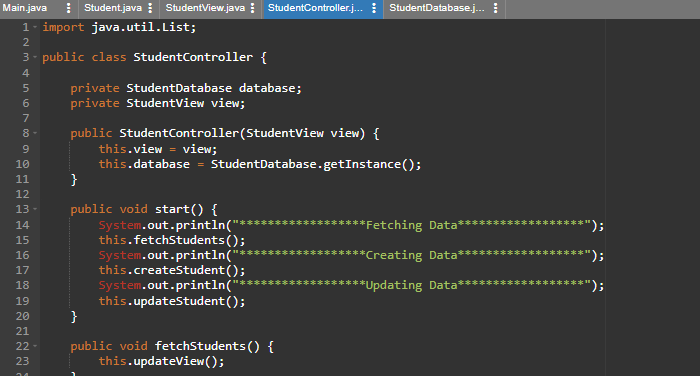


Explicación

En la clase de StudentView se creó la función de inputStudent, en donde se simula la inserción normal que haría un usuario, esto por medio del uso de datos predefinidos que son {name: Alex ,rollNo= 22}, la función retorna un nuevo objeto de tipo Student con los datos de {name: Alex ,rollNo= 22}; el cual es usado por el StudentController desde la función createStudent, en donde inicializa un nuevo student y llama a la función de inputStuden. Luego el objeto que devuelve la función mencionada es guardado en la base y se presenta toda la información almacenada en donde se verifica que el nuevo estudiante en efecto se guarda.



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

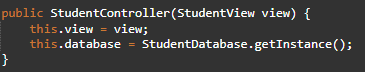


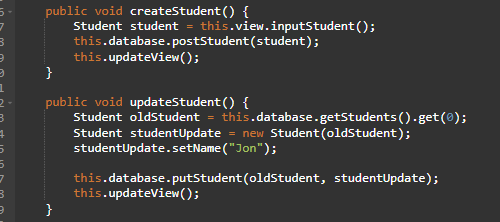
Explicación:

El controlador cumple con su rol de intermediario entre el modelo y la vista, lo que se ajusta demuestra perfectamente que el controlador ha sido modificado para hacer uso únicamente de los métodos que actúan como intermediario entre el modelo y la vita.

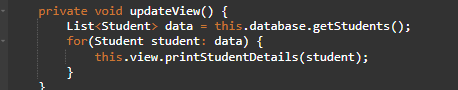
El controlador no está haciendo ninguna manipulación directa de los datos más que pedir y enviar datos entre la vista y la base de datos, ni está realizando ninguna lógica de negocio compleja.

Su función principal es:

* Pedir datos de la vista StudentDatabase.getInstance();  
  
* Enviar los datos al modelo para su almacenamiento o modificación



* Actualizar la vista con los cambios o los datos obtenidos del modelo

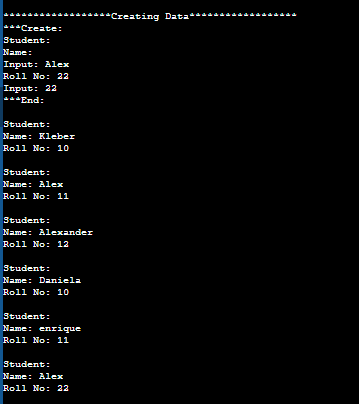


1. se creó una clase que simula una base de datos, la cual brinda apoyo en la administración de los datos quemados.

Código:



Ejecución:



Explicación:

La clase Student simula métodos básicos de un CRUD, los cuales están especializados para recibir, modificar y retornar información del modelo Student, dicha clase agrega estudiantes prefijados en el código. Además, hace uso del patrón Singleton, ya que su constructor es privado y para poder acceder a una instancia de la clase, se lo hace mediante su método *getInstance()*, de esa manera se evita duplicar la información de la base de datos, y se mantiene centralizada el uso de la clase.

Ejecucion del codigo/ Pantallas