ALUMNOS Y ASIGNATURAS

INSTRUCCIONES

Lee cuidadosamente cada una de las cuestiones que se te presentan a continuación y responde de acuerdo con lo solicitado.

El Colegio Latinoamericano necesita desarrollar una aplicación que permita llevar el control de las notas de los estudiantes de 1ro medio y debe cumplir con las siguientes características:

DESCRIPCIÓN:

MENÚ

La aplicación debe contar con un menú el cual permita cargar un archivo con datos, crear alumno, agregar materia a alumno, agregar notas y exportar datos a un archivo.

AGREGAR DATOS

Se debe permitir crear nuevos alumnos, materias, asignarlas a alumnos, y agregar notas a distintas materias.

CÁLCULO DE PROMEDIOS

Se debe agregar la funcionalidad para calcular los promedios de cada materia y alumno.

EXPORTAR DATOS

Se debe generar un archivo .txt con el promedio de notas de cada alumno existente. El resultado en el archivo .txt puede ser similar a lo siguiente:

```
Alumno : 17.423.112-4 - Samuel
Materia : MATEMATICAS - Promedio : 4.1

Alumno : 18.546.232-1 - Pepe
Materia : MATEMATICAS - Promedio : 4.4
Materia : LENGUAJE - Promedio : 3.4
```

HITO 1

1. Crear proyecto a través de eclipse como Proyecto Maven configurado como proyecto simple y crear clase MenuTemplate en carpeta vistas.

   Esta clase debe implementar un scanner para recibir datos por consola. Además, contiene los siguientes métodos:

   - *exportarDatos*, sin mayor implementación.
   - *crearAlummno*, sin mayor implementación.
   - *agregarMateria*, sin mayor implementación.
   - *agregarNotaPasoUno*, sin mayor implementación.
   - *listarAlummnos*, sin mayor implementación.
   - *terminarPrograma*, sin mayor implementación.
   - *iniciarMenu*, muestra el menú principal y recibe la entrada del teclado a través del scanner. Contiene la lógica para denotar los demás métodos en base a la entrada del teclado.

El objetivo de tener MenuTemplate es que se deben sobrescribir los métodos cada vez que se herede la clase, de esta forma se puede tener un menú diferente, pero siempre en el mismo orden y con el mismo comportamiento. El único método que no se debe sobrescribir es iniciarMenu, ya que este ya contiene su implementación final.

2. Crear clase Menu en carpeta vistas, Menu debe heredar de MenuTemplate. Y debe contener los siguientes métodos:
   - *alumnoServicio*, instancia de AlumnoServicio.
   - *archivoServicio*, instancia de ArchivoServicio.

Además, sobrescribir estos los siguientes métodos:

- *archivoServicio*, instancia de ArchivoServicio.
- *exportarDatos*, llama a método para exportar promedios.
- *crearAlummno*, solicita ingreso de datos y llena objeto de tipo Alumno.
- *agregarMateria*, muestra menú para asignar materia a un alumno.

- *agregarNotaPasoUno*, muestra menú para asignar nota a un alumno. listarAlumnos, muestra lista de alumnos.
- *terminarPrograma*, el cual finaliza la ejecución del sistema.

Aparte, se sugiere crear una clase Utilidad en carpeta utilidades, que contenga métodos reutilizables para el menú como limpiar pantalla, mostrar mensajes, etc.

3. Crear clase Alumno en carpeta modelos, con los siguientes atributos:
   - *Rut*
   - *Nombre*
   - *Apellido*
   - *Dirección*
   - *Materias*, lista de materias del alumno, se puede seleccionar Set o List como tipo de dato.
   - Crear getters y setters para los atributos

4. Crear Enumeración llamada MateriaEnum, en carpeta modelos.

   Debe contener los siguientes enums:

   - *MATEMATICAS*
   - *LENGUAJE*
   - *CIENCIA*
   - *HISTORIA*

5. Crear clase Materia en carpeta modelos, con los siguientes atributos:
   - *Nombre*, de tipo MateriaEnum.
   - *Notas*, lista de notas de la materia del alumno, se puede usar un tipo List para esta propiedad.
   - Crear getters y setters para los atributos.

6. Crear clase AlumnoServicio, en carpeta servicios.
   - Crear atributo llamado listaAlumnos de tipo Map<String, Alumno>.
   - Crear método crearAlummno, recibe un parámetro de tipo alumno.

- Crear método agregarMateria, recibe rutAlumno de tipo String, y currentMate de tipo Materia.
- Crear método materiasPorAlumnos, retorna List, recibe rutAlumno de tipo String.
- Crear método listarAlumnos, retorna un Map<String, Alumno>.

HITO 2

1. Crear clase ArchivosServicio, en carpeta servicios. Esta clase se compone de:

   - *alumnosACargar*, de tipo List en donde se mantendrán los datos mientras se itera el archivo.
   - *promediosServicioImp*, instancia de clase PromedioServicioImp.
   - Crear método *exportarDatos*, recibe alumnos de tipo Map<String,Alumno>, y la ruta en donde se exportará el archivo.

2. Crear clase PromedioServicioImp, en carpeta servicios.

   - Crear método *calcularPromedio*, el cuál recibe una lista de valores y retorna el promedio.

3. Añadir dependencias para pruebas:

   - Mediante una dependencia maven o añadiendo .jar de JUnit5 y Mockito al proyecto.

4. Escribir pruebas unitarias para PromedioServicio.

   - Método *calcularPromedioTest* para verificar el funcionamiento de calcularPromedio.

5. Escribir pruebas unitarias para AlumnoServicio y ejecutarlas.

   - Atributo alumnoServicio, instancia de AlumnoServicioImp.
   - Atributo alumnoServicioMock, mock de AlumnoServicioImp para simular comportamiento.
   - Atributo matemáticas, instancia de una nueva Materia.
   - Atributo lenguaje, instancia de una nueva Materia.
   - Atributo mapu, instancia de Alumno.

- Método setup, como fixture para reutilizar datos de prueba.
- Método crearAlumnoTest para verificar el funcionamiento de crearAlumno.
- Método agregarMateriaTest para verificar el funcionamiento de agregarMateria.
- Método materiasPorAlumnosTest, usando mock para verificar el funcionamiento de materiasPorAlumnos.
- Método listarAlumnosTest para verificar el funcionamiento de listarAlumnosTest.

VISTAS AL MOMENTO DE EJECUTAR:

AL EJECUTAR EL MENÚ:

```
1. Crear Alumnos
2. Listar Alumnos
3. Agregar Materias
4. Agregar Notas
5. Salir
6. exportarDatos
Selección:
```

AL CREAR UN ALUMNO

```
--- Crear Alumno ---
Ingresa RUT: 1.111.111-1
Ingresa nombre: Milton
Ingresa apellido: P.
Ingresa dirección: Casita 1

 --- ¡Alumno agregado! ---
```

Cabe destacar que una vez finalizado la creación de un alumno se debe volver a la vista del menú con sus 6 opciones.

AGREGAR MATERIA

Ingresar la opción 3, debe llevar a Agregar Materias Lo cual ejecuta el método agregarMaterias y solicitará la siguiente información. Ingresando el rut del alumno, y seleccionando la materia a agregar.

```
--- Agregar Materia ---
Ingresa rut del Alumno: 1.111.111-1

1. MATEMATICAS
2. LENGUAJE
3. CIENCIA
4. HISTORIA

Selecciona una Materia: 1

 --- ¡Materia agregada! ---
```

AGREGAR NOTA

Ingresar la opción 4, debe llevar a Agregar Notas. Lo cual ejecuta el método agregarNotaPasoUno y solicitará el rut del Alumno. Una vez ingresado el rut, solicita seleccionar una de las materias pertenecientes al alumno, para ingresar la nota.

```
--- Agregar Nota ---
Ingresa rut del Alumno: 1.111.111-1
Alumno tiene las siguientes materias agregadas:
1.MATEMATICAS

Seleccionar materia: 1
Ingresar nota: 6.5

 --- ¡Nota agregada a MATEMATICAS! ---
```

LISTAR ALUMNOS

Ingresar la opción 2, debe llevar a Listar Alumnos Lo cual ejecuta el método listarAlumnos y muestra a los alumnos existentes con sus materias y las notas.

```
--- Listar Alumnos ---
Datos Alumno
	RUT: 1.111.111-1
	Nombre: Milton
	Apellido: P.
	Dirección: Casita 1

	Materias
		MATEMATICAS
			Notas:
			[6.5]
```

EXPORTAR DATOS

Al seleccionar la opción 6, se solicitará la ruta en donde se exportará el archivo **promedios.txt**

```
--- Exportar Datos ---
Ingresa la ruta en donde se encuentra el archivo notas.csv :
/home/usuario/datos-exportados
Datos exportados correctamente.
```

En nuestro archivo promedios.txt el resultado debe asemejarse a lo siguiente:

```
Alumno : 1.111.111-1 - Milton[materia [nombre=MATEMATICAS, notas=[6.5]]]
	Materia: MATEMATICAS - Promedio: 6.5
```