Entrega Acumulada Parte #B

**Nombre:** Fernando Melillán T. (Civil), Beatriz Segura P. (Civil)

**Tema:** Gestión‌ ‌de‌ ‌avance‌ ‌curricular‌ ‌de‌ ‌los‌ ‌alumnos‌ ‌en‌ ‌un‌ ‌instituto:‌ ‌Registro‌ ‌de‌ ‌alumnos,‌ ‌profesores,‌ ‌

asignaturas‌ ‌y‌ ‌de‌ ‌nivel‌ ‌de‌ ‌avance‌ ‌de‌ ‌un‌ ‌alumno‌ ‌dependiendo‌ ‌de‌ ‌su‌ ‌avance‌ ‌en‌ ‌la‌ ‌malla.‌

**Parte A**

Entrega Acumulada

**Tema:** Gestión de avance curricular de los alumnos en un instituto: Registro de alumnos, profesores, asignaturas y de nivel de avance de un alumno dependiendo de su avance en la malla.

**Nombres:** Pablo Torreblanca L. (Ejecución)

Fernando Melillán T. (Civil)

Beatriz Segura P. (Civil)

**1.1 Análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido**

**a la realización del proyecto.**

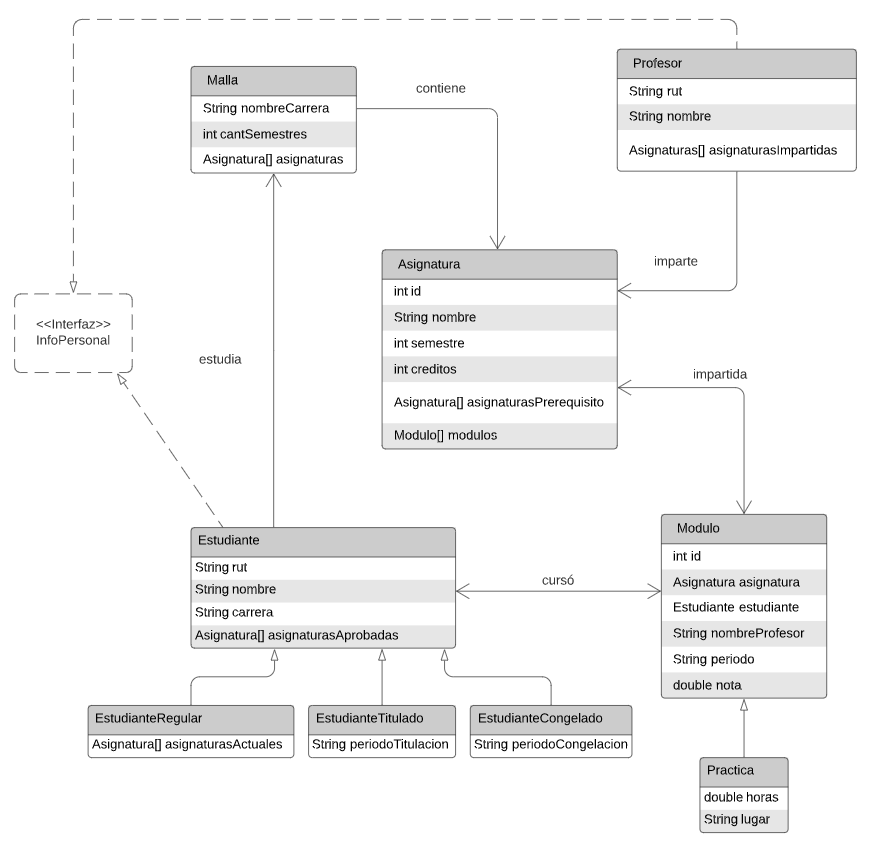
El proyecto consiste en gestionar el avance curricular de los estudiantes de un instituto, es decir, permite visualizar el avance del estudiante con respecto a la malla de la carrera, esto se logrará al almacenar las asignaturas aprobadas y a partir de esta información calcular el total de créditos aprobados hasta el periodo actual de forma acumulativa para saber en qué semestre se encuentra el estudiante.

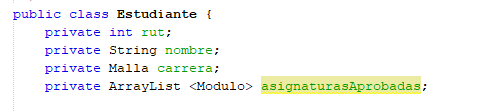
Además, se contará con los siguientes registros: información académica y personal de los estudiantes; asignaturas dictadas por cada profesor y las mallas curriculares con sus asignaturas correspondientes (teniendo en cuenta las asignaturas con prerrequisitos).

**1.2 Diseño conceptual de clases del Dominio y su código en Java**

Identificamos cuatro entidades principales del sistema:

* **Estudiante**: Esta entidad contiene los datos de los estudiantes del instituto, sus atributos son: el rut, nombre del estudiante, carrera a la que pertenece y una lista con las asignaturas aprobadas por este como módulos.Es una clase abstracta ya que de esta se crean 3 subclases hijas.
* **EstudianteRegular:** Esta subclase pertenece al estudiante habilitado para tomar ramos, este contiene un arreglo con las asignaturas que imparte el semestre actual.
* **EstudianteCongelado:** Esta subclase pertenece al estudiante que congeló su periodo de estudio, almacena el periodo en que realizó dicho tramite.
* **EstudianteTitulado:** Esta subclase pertenece al estudiante egresado que se tituló de la carrera, almacena el periodo en que egresó.
* **Profesor**:Esta entidad contiene los datos de los profesores del instituto, nombre del profesor, rut, y una lista con las asignaturas que imparte.
* **Malla**: Entidad que contiene la malla de una carrera en específica cuyos atributos son: nombre de la carrera a la que pertenece, la cantidad de semestres que conforman la malla, y un arreglo con las asignaturas que conforman la malla.
* **Asignatura**: Esta entidad contiene los datos de cada asignatura, cuyos atributos son: id de la asignatura para su fácil identificación, nombre de la asignatura, una lista con las asignaturas que son pre-requisitos de esta, es decir, aquellas asignaturas que el estudiante debe tener aprobadas para poder inscribirla, y finalmente una lista con los módulos asociados a esta asignatura, es decir, la entidad que corresponde a la vez que un estudiante cursó esta asignatura en un periodo determinado y la aprobó..
* **Módulos:** Esta entidad contiene los datos de los módulos que han cursado los estudiantes, es por ello, que almacena la asignatura a la que está asociada y que el estudiante cursó, el estudiante asociado, el nombre del profesor que dictó la asignatura, el periodo en que se dictó, la nota con la que aprobó el estudiante y un id único.
* **Practica:** Esta subclase es hija de la clase Módulo, y corresponde a la práctica realizada por el estudiante perteneciente a esa asignatura y al módulo correspondiente, almacena la cantidad de horas de práctica y el lugar en que la realizó.
* **InfoPersonal:** Esta clase es una interfaz y muestra la información básica de las clases profesor y estudiante inclusive las clases hijas.

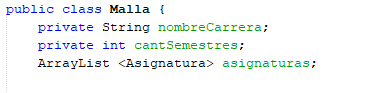


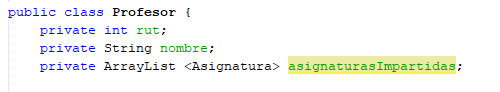


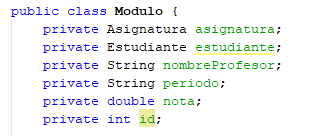


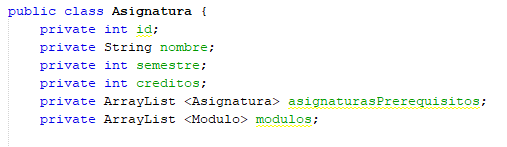








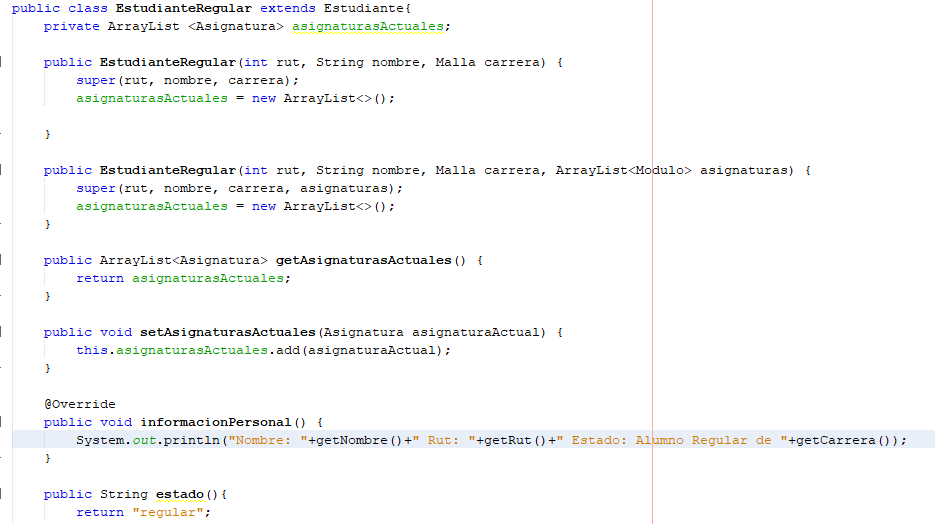


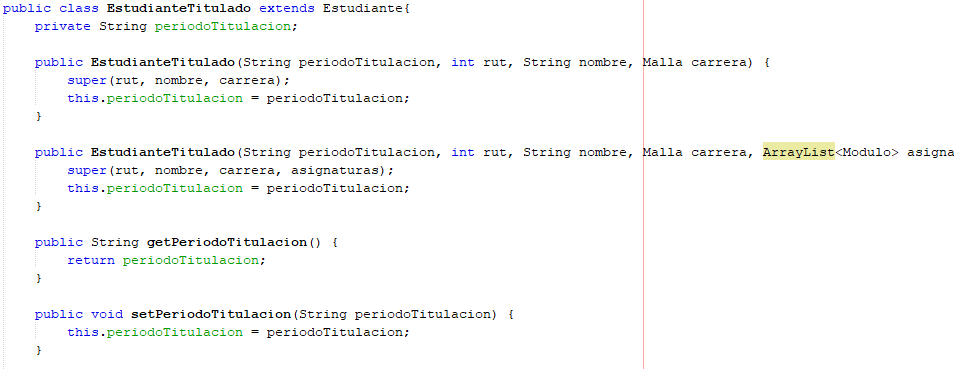


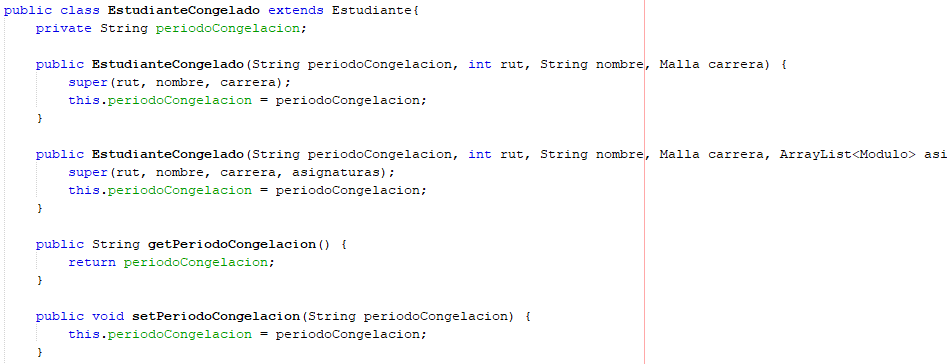


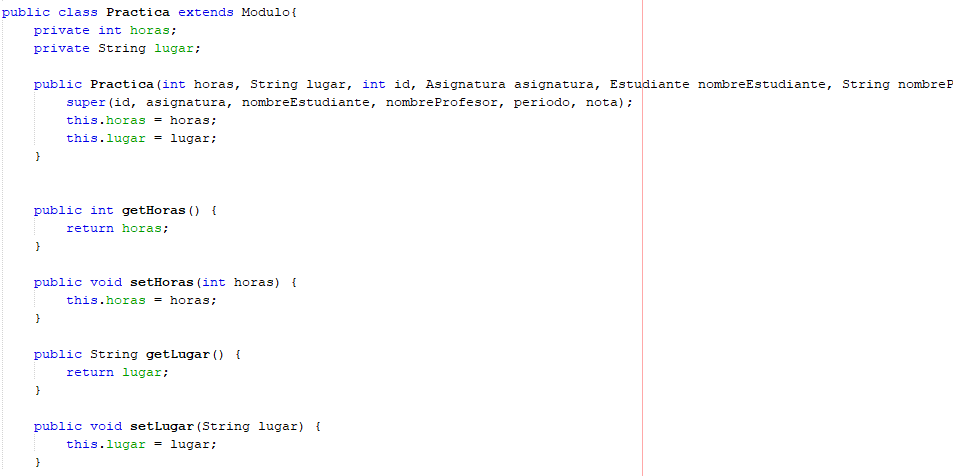
**1.3 Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).**

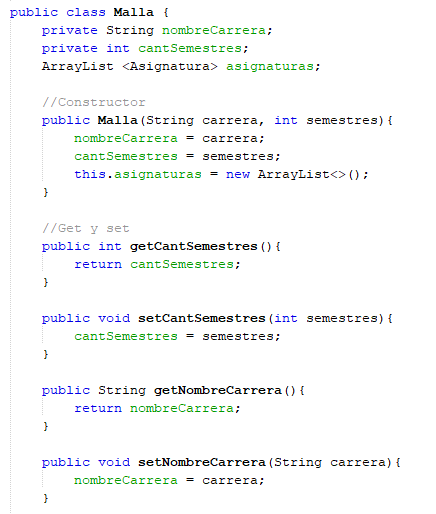
Se cumple que todos los atributos de las clases son privados como se observó en las capturas de código anteriormente. A continuación se muestran todos los métodos getter y setter de las clases creadas.

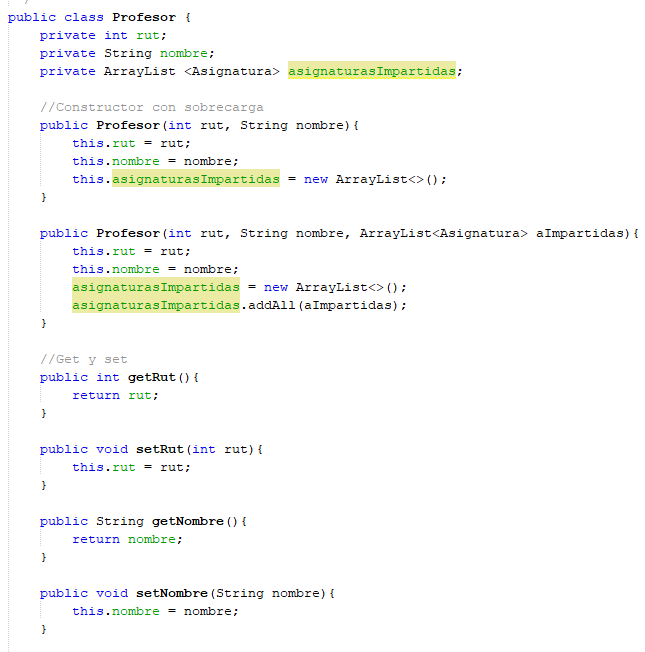


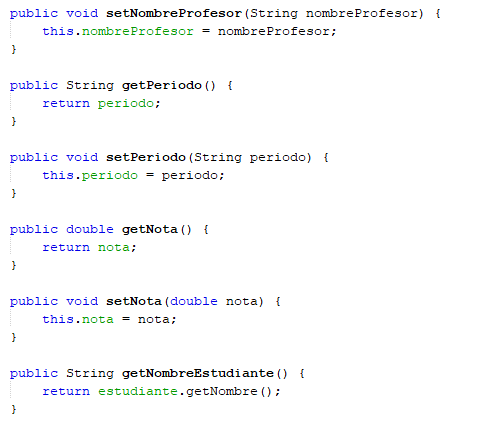


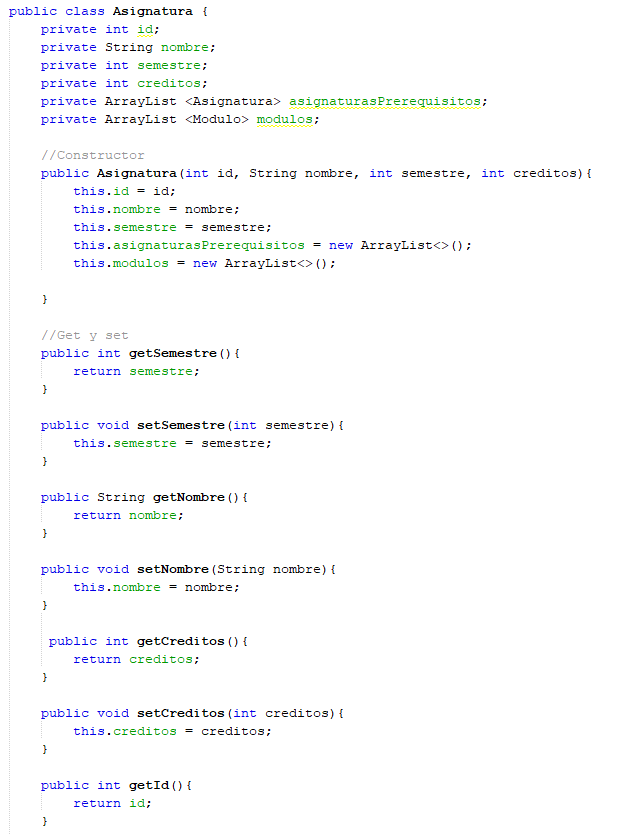












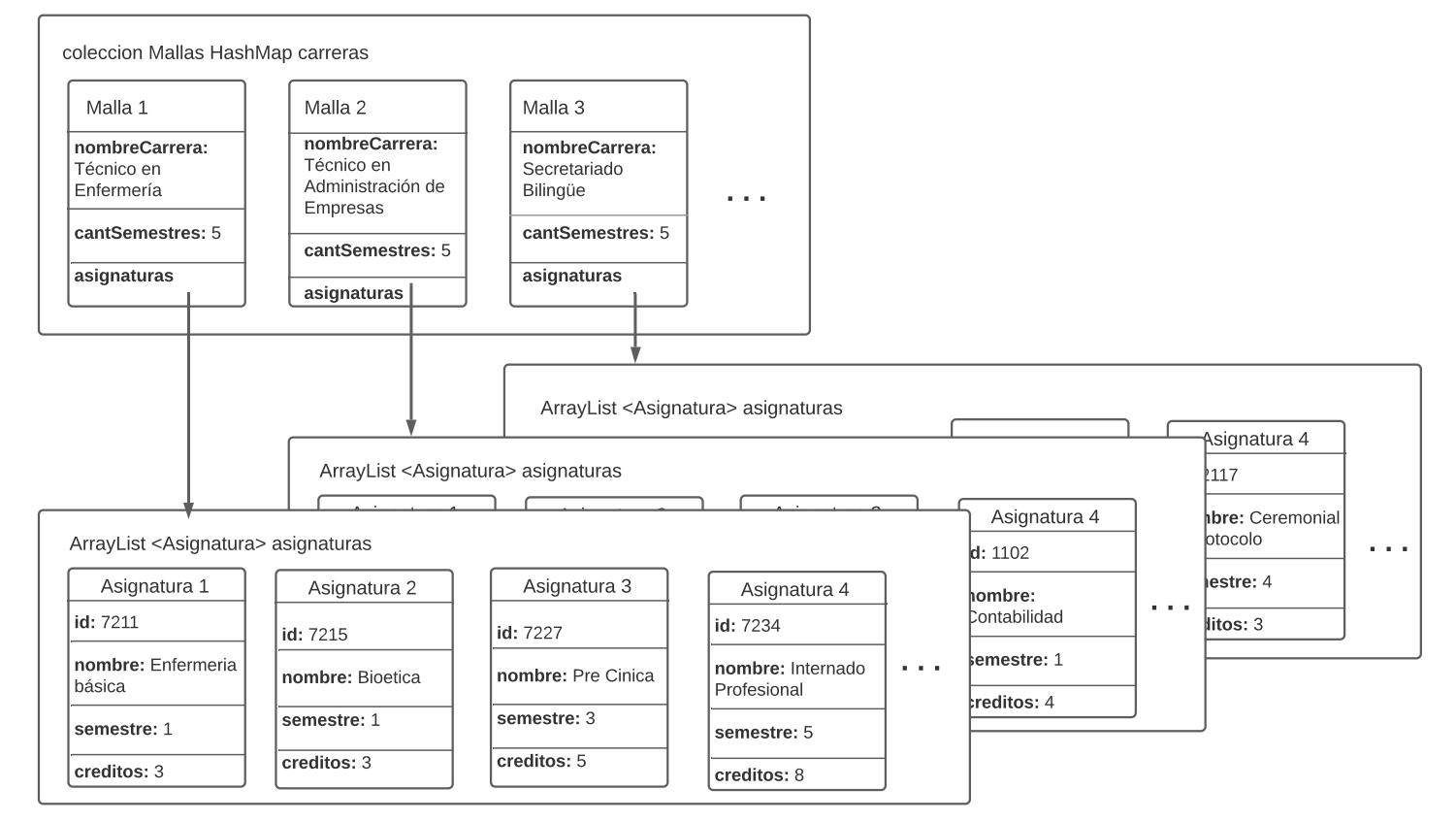
**1.4 Se deben incluir datos iniciales dentro del código.**

Los datos iniciales son ingresados a través de la carga de dos archivos .csv, estos contienen los datos de las carreras, es decir, las mallas con sus asignaturas asociadas y por otro lado un archivo con todas las asignaturas y su información. Estos archivos son: carreras.csv y asignaturas.csv.



**2.1 Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos.**

Las colecciones anidadas principales a crear son carreras, la cual es un mapa que contiene las mallas de todas las carreras junto a sus atributos: nombre carrera, cantidad de semestres y asignaturas de la carrera, es decir, en su interior anida una colección de asignaturas pertenecientes a dicha carrera, esta es una lista que contiene las asignaturas propias de la carrera con sus atributos: id de la carrera, nombre, semestre a la que pertenece y los créditos.

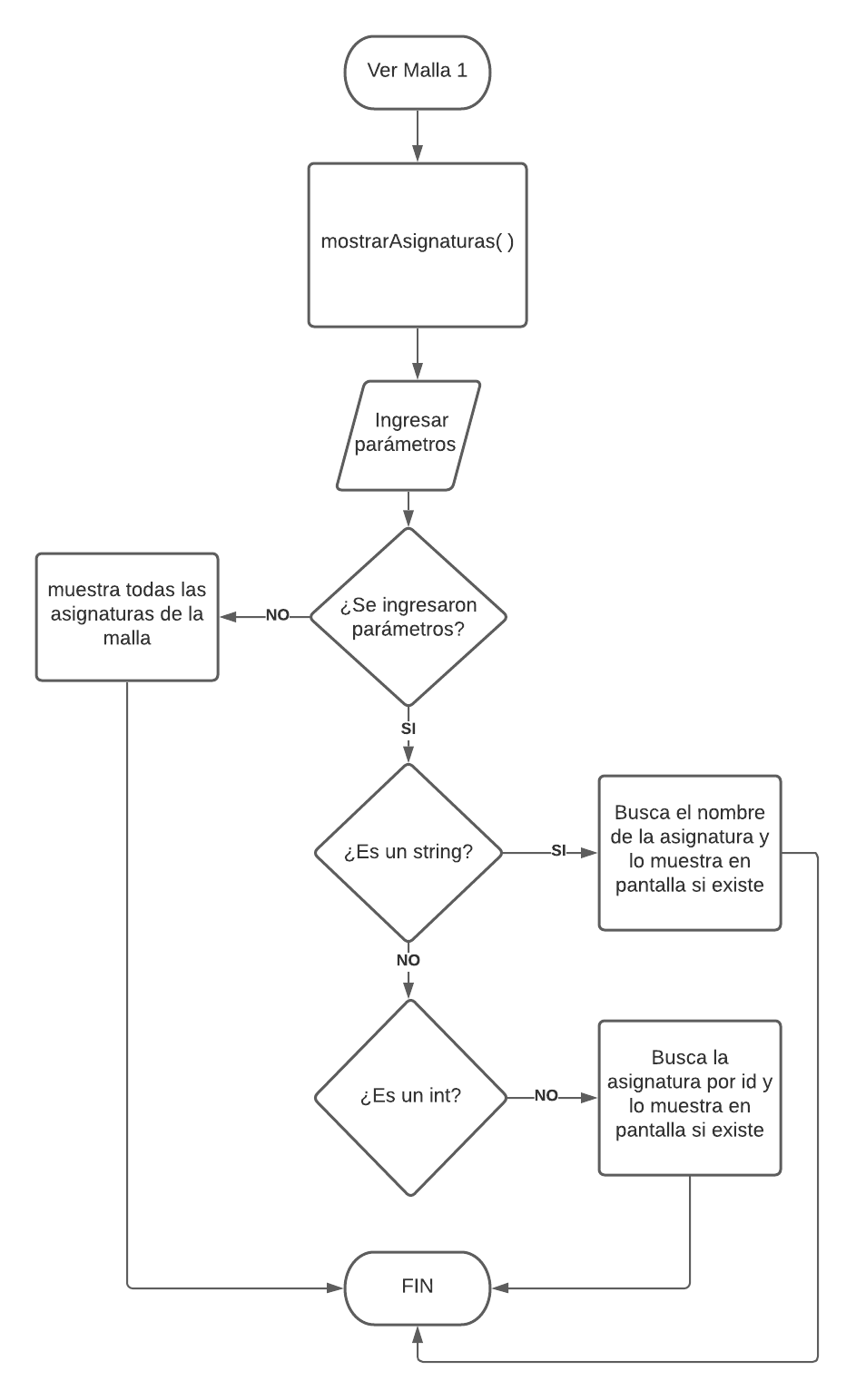


Como se puede observar la colección carreras contiene los objetos Malla, donde cada una tiene anidada sus asignaturas correspondientes, por ejemplo Malla 1 perteneciente a la carrera de *Técnico en Enfermería* tiene asociadas sus asignaturas: *Enfermería básica, Bioética, Pre Clínica, Internado Profesional, etc.*

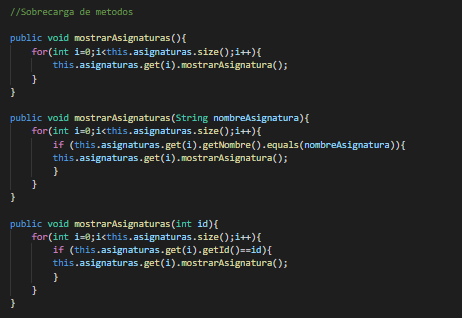
**2.2 Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos.**

La sobrecarga se implementa en dos clases:

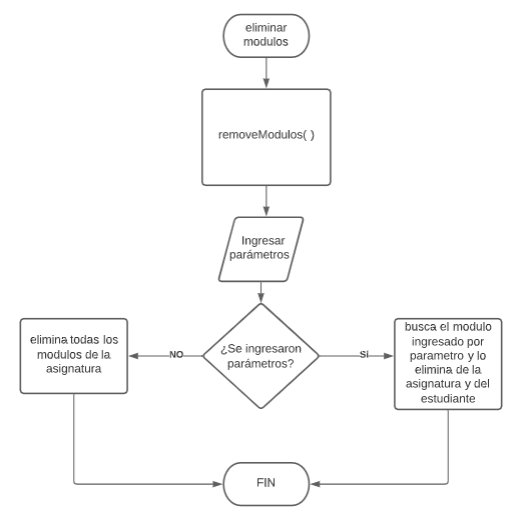
**Malla:** Se crean 3 métodos de mostrar Asignaturas donde la primera sin recibir parámetros muestra todas las asignaturas propias de la malla, el segundo al recibir String nombreAsignatura mostrará la asignatura con ese nombre dentro de la malla y por último al recibir un int id mostrará la asignatura que tiene esa id ingresada



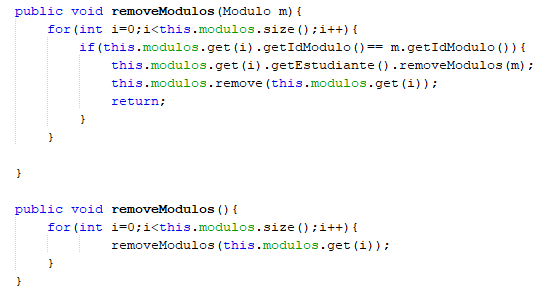
Código sobrecarga de clase Malla:



* **Asignatura:** Se crean 2 métodos de remover módulos llamado removeModulos, donde el primero al recibir un parámetro de tipo Modulo eliminará el módulo de la lista del Estudiante asociado y de la lista que contiene esta Asignatura. Mientras que el segundo método sin recibir parámetros, realiza el mismo procedimiento pero removiendo todos los módulos asociados a esta Asignatura.

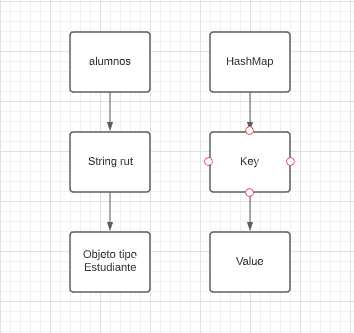


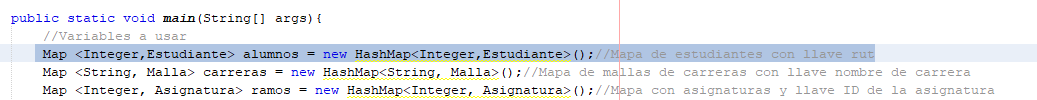
Código sobrecarga de clase Asignatura:

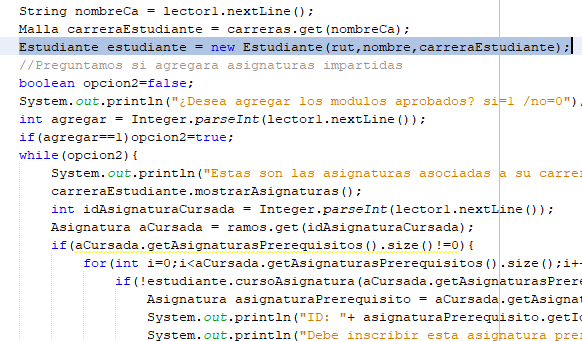


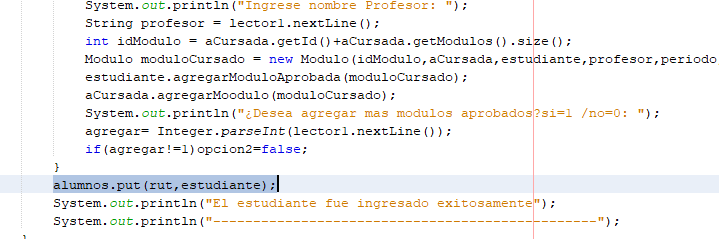
**2.3 Diseño conceptual y codificación de al menos una clase mapa del JCF.**

Hemos incorporado dentro de las funcionalidades del programa el HashMap “alumnos” que será utilizado para el almacenamiento de los datos de información personal tales como: nombre, apellido, rut y n° de semestre que se encuentra cursando el estudiante. Esto nos permitirá el manejo de los datos del alumno y poder utilizar el RUT como llave para la inscripción de asignaturas,mostrar notas, asignaturas activas, etc.







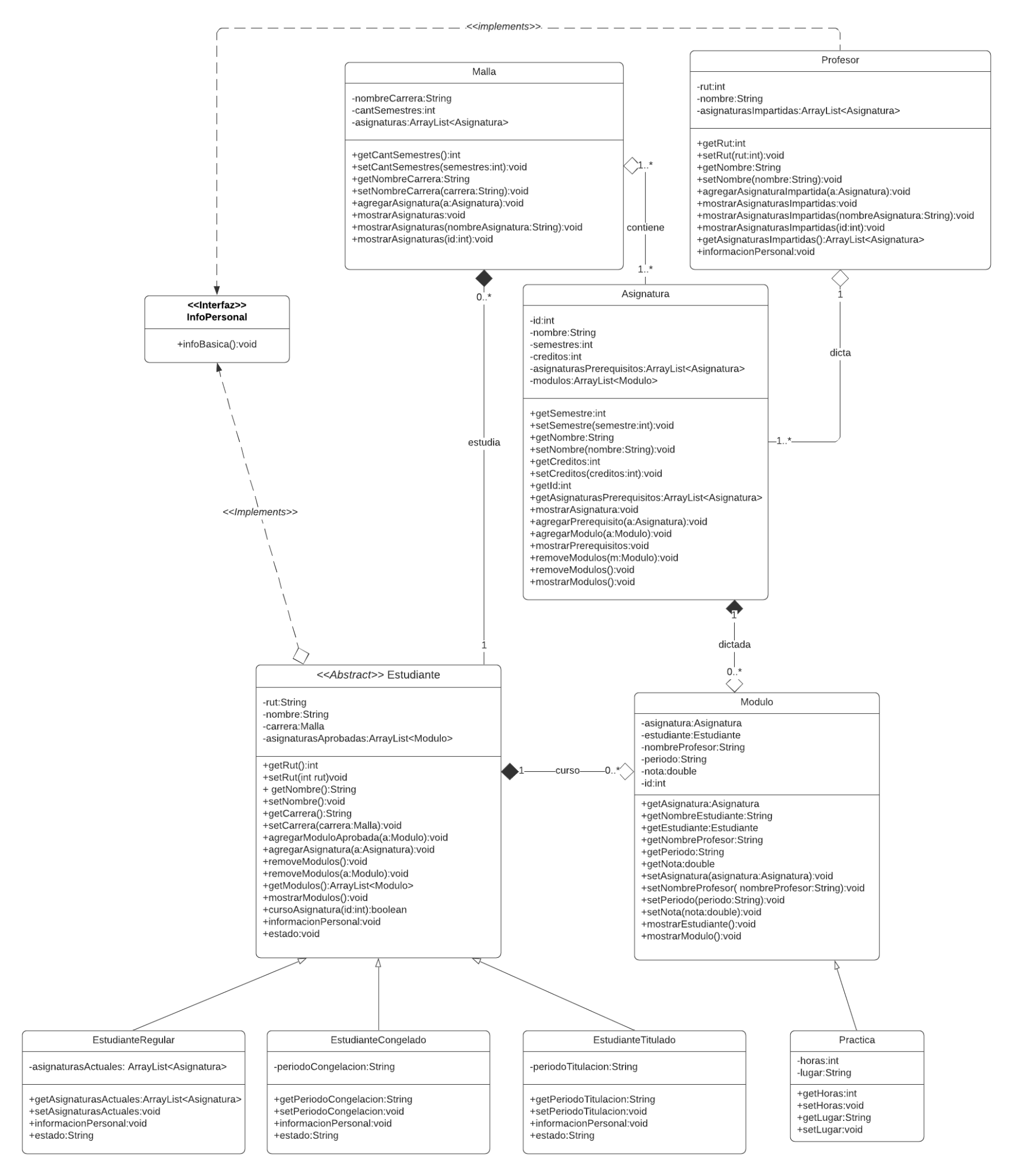


**3.1 Diseño de diagrama de clases UML**

En el diagrama de clases contamos con 5 clases:

* **Malla:** Contiene los datos importantes de las carreras que imparten en el instituto, guarda el nombre de la carrera, la cantidad de semestres que esta tiene y una lista con las asignaturas de la malla de la carrera. Estas carreras están almacenadas en un mapa cuya clave es el nombre de esta misma y el valor objetos de tipo Malla. Malla contiene 1 o muchas Asignaturas, y a su vez esta la estudian 0 o muchos Estudiantes.
* **Asignatura:** Contiene los datos de cada asignatura, una id única, nombre de la asignatura, semestre en el que es dictado dentro de la malla, créditos, lista con asignaturas que son prerrequisitos a esta y una lista con los módulos dictados de esta asignatura. Puede pertenecer a 1 o más mallas de carrera, puede ser dictada por solo 1 profesor, y pudo haber sido dictada 0 o más veces, es decir, 0 o más módulos.
* **Modulo:** Esta clase guarda toda la información de los módulos que han sido dictadas las asignaturas, contiene la id única del módulo, la asignatura a la que pertenece, el estudiante que la cursó, el nombre del profesor que la dictó, el periodo en la que fue dictada, y la nota que el estudiante obtuvo luego de aprobar. Este módulo existe solo si existe la asignatura y el estudiante que la cursaron, está asociada a una única asignatura y un único estudiante.
* **Estudiante:** Es una clase Abstracta. Contiene toda la información de un estudiante, su rut, el nombre, la carrera que cursa y una lista de los módulos que ha cursado. Este puede estudiar solo una carrera por lo que puede estar asociado sólo a una malla y no puede existir si no está asociado a una carrera, también puede haber cursado 0 o muchos módulos.
* **Profesor:** Contiene toda la información de profesor, nombre, rut, y una lista con las asignaturas que dicta. Este puede dictar 1 o más asignaturas.
* **EstudianteRegular**: Es una clase hija de Estudiante, por ende hereda sus métodos de la superclase, se puede almacenar en esta las Asignaturas cursadas actualmente, estas son almacenadas en un arreglo de Asignaturas.
* **EstudianteCongelado**: Es una clase hija de Estudiante, por ende hereda sus métodos de la superclase, se puede almacenar en esta el periodo en que congeló, este dato es de tipo String.
* **EstudianteTitulado**: Es una clase hija de Estudiante, por ende hereda sus métodos de la superclase, se puede almacenar en esta el periodo en que egresó, este dato es de tipo String.
* **Practica:**Esta clase es hija de la clase Modulo y almacena en esta las horas cursadas durante la práctica de tipo int y el lugar donde se realizó de tipo String.
* **InfoPersonal:** Esta clase es una interfaz, y está impementada por las clases Profesor y Estudiante, por lo que a su vez es heredada por EstudianteRegular, EstudianteCongelado y EstudianteTitulado.

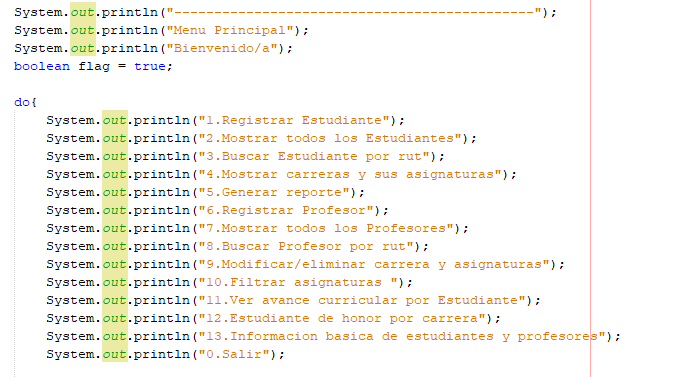
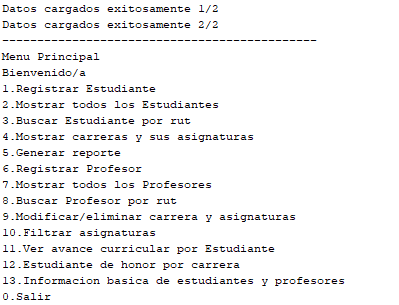
A continuación, se presenta el diagrama UML del proyecto:



**3.2 Implementación de menú del sistema para funcionalidades**

A través del menú principal, el usuario puede ingresar Estudiantes, ver el total de Estudiantes registrados, Buscar un estudiante en particular por su Rut y ver las carreras existentes (Mallas) y sus asignaturas respectivas.

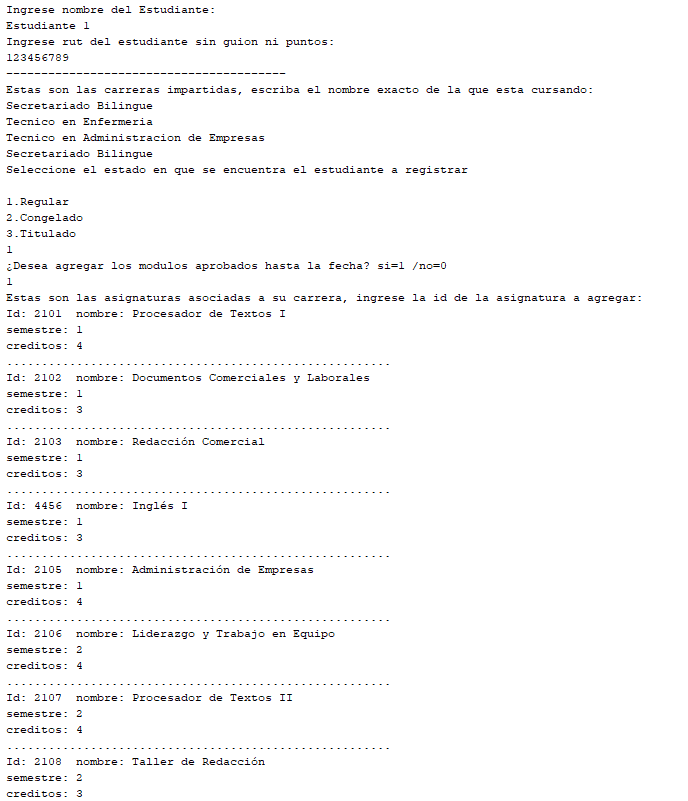
* En Registrar estudiante el usuario puede agregar un estudiante con sus atributos: rut, nombre y la carrera que cursa, además se le da la opción de agregar las asignaturas que ha aprobado, es decir, puede registrar sus módulos cursados ingresando la asignatura, periodo en el que rindió la asignatura, la nota de aprobación y el nombre del profesor.
* En mostrar todos los estudiantes, el usuario puede ver todos los estudiantes registrados con su nombre y carrera respectiva.
* En buscar estudiante, el usuario puede buscar a estudiante por su rut, este le mostrará el nombre del estudiante y la carrera que cursa.
* En Mostrar carreras y asignaturas, el usuario puede ver todas las carreras que se dictan y sus asignaturas correspondientes
* En generar reporte, el usuario puede almacenar la información referente a las carreras impartidas, junto con su duración en semestres y su malla curricular.
* En Registrar profesor, el usuario puede registrar un profesor junto a sus atributos, como también puede incorporar asignaturas que este imparte.
* En mostrar todos los profesores, el usuario puede ver todos los profesores registrados.
* En Buscar profesor por rut, el usuario puede buscar a un profesor por el rut de este y al encontrarlo mostrará el nombre del profesor y las asignaturas que este imparte.
* En Modificar/eliminar carrera y asignatura, el usuario puede modificar el nombre y duración de la carrera, como también puede eliminarla. Además, puede modificar los datos de las asignaturas como el nombre y los créditos, y a su misma vez puede eliminar asignaturas.
* En filtrar asignaturas permite al usuario ver las asignaturas por carrera filtradas por créditos, o semestre.
* En Ver avance curricular por Estudiante el usuario puede ver la información académica por estudiante pudiendo acceder a su promedio de notas y créditos aprobados.
* En Estudiante de Honor por carrera el usuario puede ver el mejor estudiante, es decir, aquel estudiante con el mejor promedio de notas perteneciente a la carrera. Esta funcionalidad permite al usuario ver al mejor alumno de todas las carreras impartidas en el Instituto.
* En Información Básica de estudiantes y profesores, el usuario puede ver la información básica de todos los estudiantes y profesores del Instituto, muestra el nombre, rut y estado de la persona, es decir, ya sea profesor, estudiante regular, estudiante congelado o estudiante titulado.



**3.2.1 Funcionalidad básica (1): Inserción manual o agregar elemento**

En la inserción manual, se implementó a las clases Estudiante y Módulo. Al registrar un Estudiante y almacenarlo en un HashMap es posible registrar sus asignaturas aprobadas a través de la creación de un módulo, cuya anidación es de 2 niveles, ya que, Estudiante tiene almacenada una lista con los módulos de las asignaturas aprobadas.

Por otro lado, se crean 2 mapas donde se registran Mallas y asignaturas, estas a su vez en mallas se almacenan las asignaturas asociadas a la carrera en una lista, mientras que en asignatura se almacena una lista con las asignaturas prerequisito según corresponda.



Ejemplo de cómo se registran los estudiantes manualmente y los módulos que ha cursado

**3.2.2 Funcionalidad básica (2): Mostrar por pantalla listado de elementos**

Se muestra por pantalla todas las carreras que se imparten y a su vez las asignaturas asociadas a cada carrera, también es posible ver la información de todos los estudiantes registrados como también al momento de registrar un estudiante se muestran las carreras a seleccionar y luego según la carrera seleccionada se muestran las asignaturas asociadas a esta.

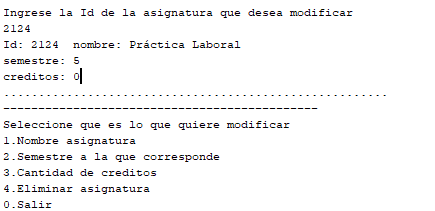
Texto

Descripción generada automáticamente

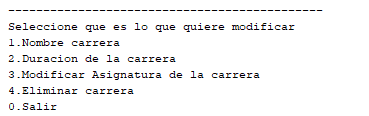
Ejemplo de salida en pantalla de como se muestran los estudiantes registrados

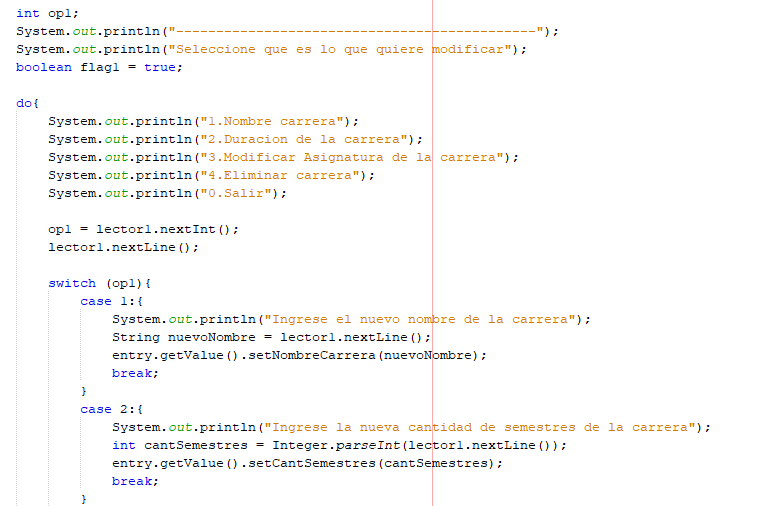
**A.2 Implementacion de las siguientes funcionalidades en el menú:**

**A.2.1 Funcionalidad básica (1): Edición/modificación de elemento**Se puede modificar información de carrera y de asignatura asociada a la carrera, en este caso, es posible modificar el nombre de la carrera y su duración en semestre, mientras que asignaturas se puede modificar el nombre de esta y la cantidad de créditos.

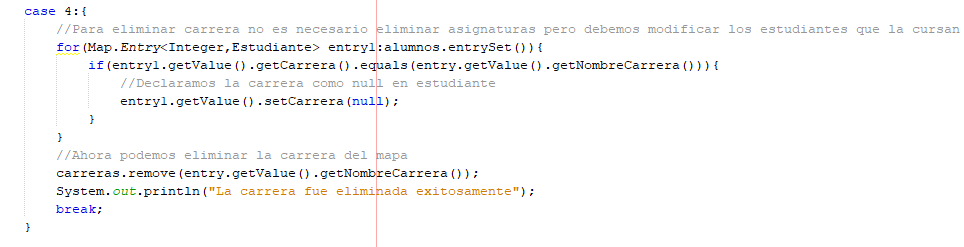
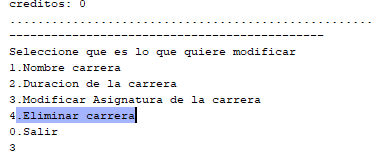
****

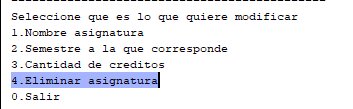
****

****

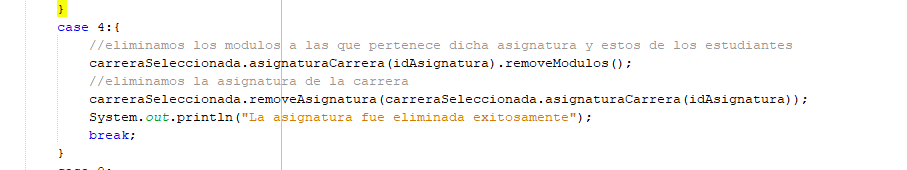
****

**A.2.2 Funcionalidad básica (2): Eliminación del elemento para las 2 colecciones anidadas.**Es posible eliminar una carrera, en este caso solo eliminamos la carrera sin las asignaturas porque hay asignaturas compartidas con otras carreras, además se modifican los alumnos asociados a esta carrera dejando el atributo carrera en null. Por otro lado, es posible eliminar la asignatura de la carrera, todos sus módulos asociados y estos últimos los eliminamos de asignaturas aprobadas de Estudiantes.

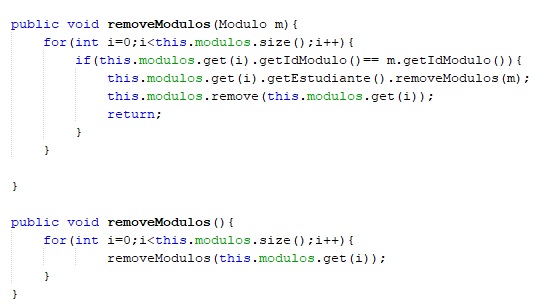




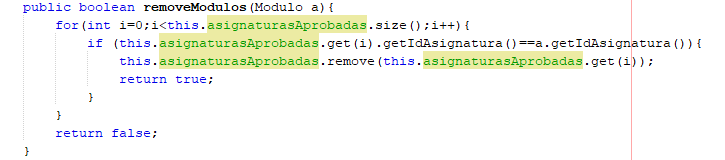
Eliminamos la asignatura junto a los módulos de la asignatura.



Métodos utilizados para eliminar módulos en la clase Asignatura.



Método para eliminar el módulo de Estudiante



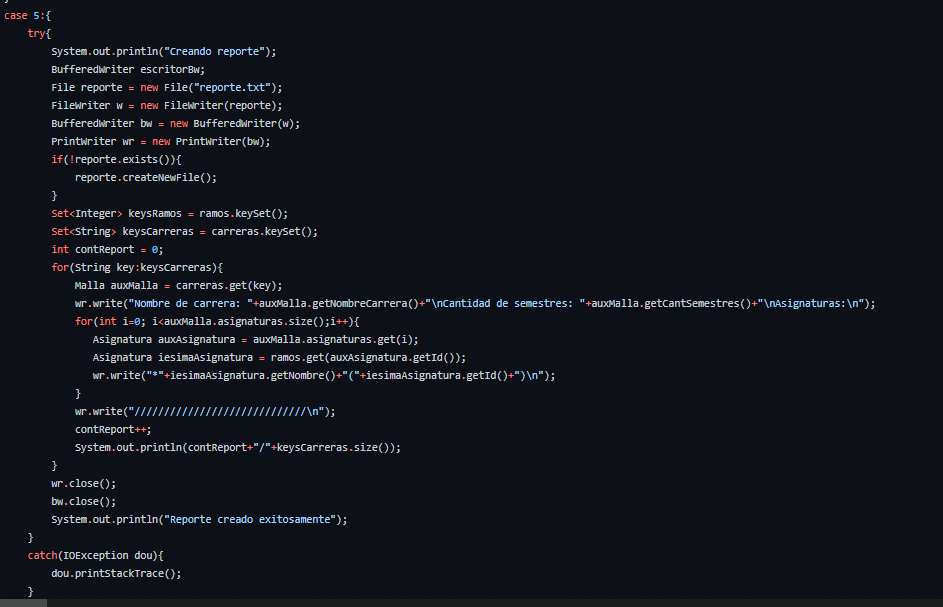
**A.3 Se debe generar un reporte en archivo txt/csv que considere mostrar datos de las 2 colecciones anidadas**

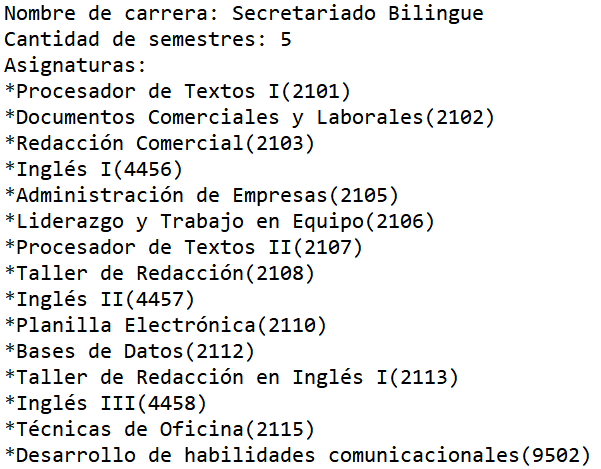
Implementada la funcionalidad de generar un reporte en un archivo de texto llamado “reporte.txt”, el cual almacena la siguiente información:

-Nombre de carrera

-Cantidad de semestres

-Listado de asignaturas y su código

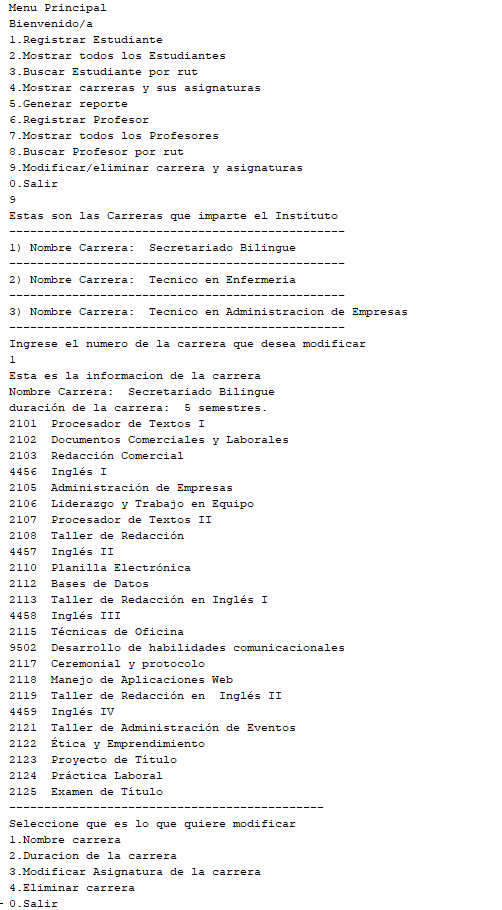


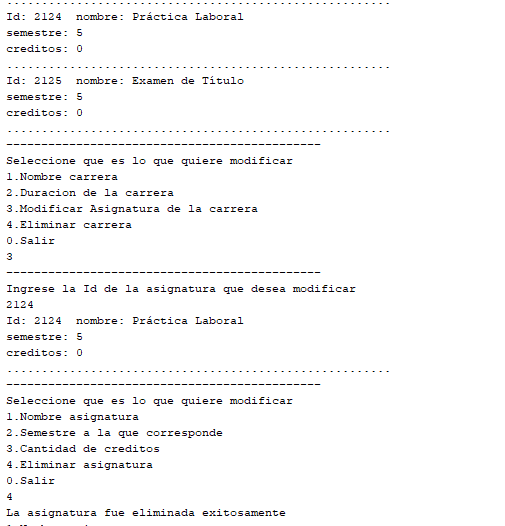


Ejemplo del archivo generado

**A.5 Todas las funcionalidades pueden ser implementadas mediante consola.**

Exceptuando la clase Menu, todas las funcionalidades del programa pueden ser implementadas mediante la consola en el menú principal.



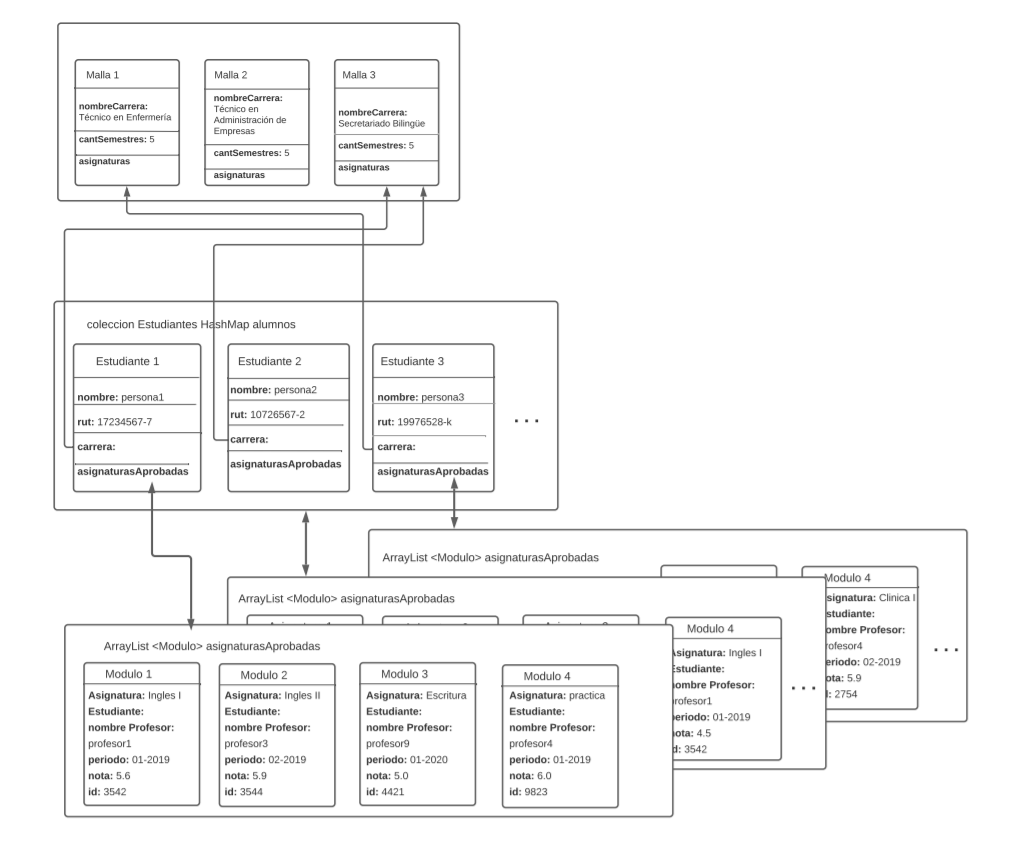


Listado de las funciones que pueden ser implementadas mediante consola.

**Parte B**

1. **Funcionalidades propias del negocio.** 
   1. **Seleccionar un objeto por criterio**

Considera la selección de un objeto basado en un criterio específico, involucrando dos o más colecciones anidadas.

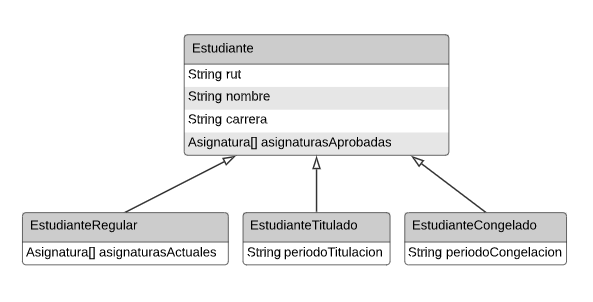


El criterio de selección es solicitar información de estudiante de honor por carrera dentro del instituto, la encontramos en el menú principal en la opción 12, se busca el estudiante con mejor promedio que esté cursando cada carrera. Para realizar esto, buscamos dentro de la colección alumnos de estudiantes a aquellos que pertenezcan a la carrera adquirida de la colección carreras de Malla, de este modo, se calcula el promedio de notas de cada estudiante de acuerdo a los módulos que ha cursado y aprobado, finalmente en cada iteración se busca el que tenga el promedio mayor.

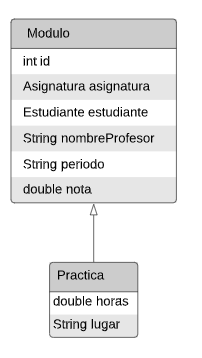
* 1. **Subconjunto filtrado por criterio**

Considera la selección de un subconjunto de objetos basado en un criterio específico, involucrando dos o más colecciones anidadas.

1. **Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos.**

****

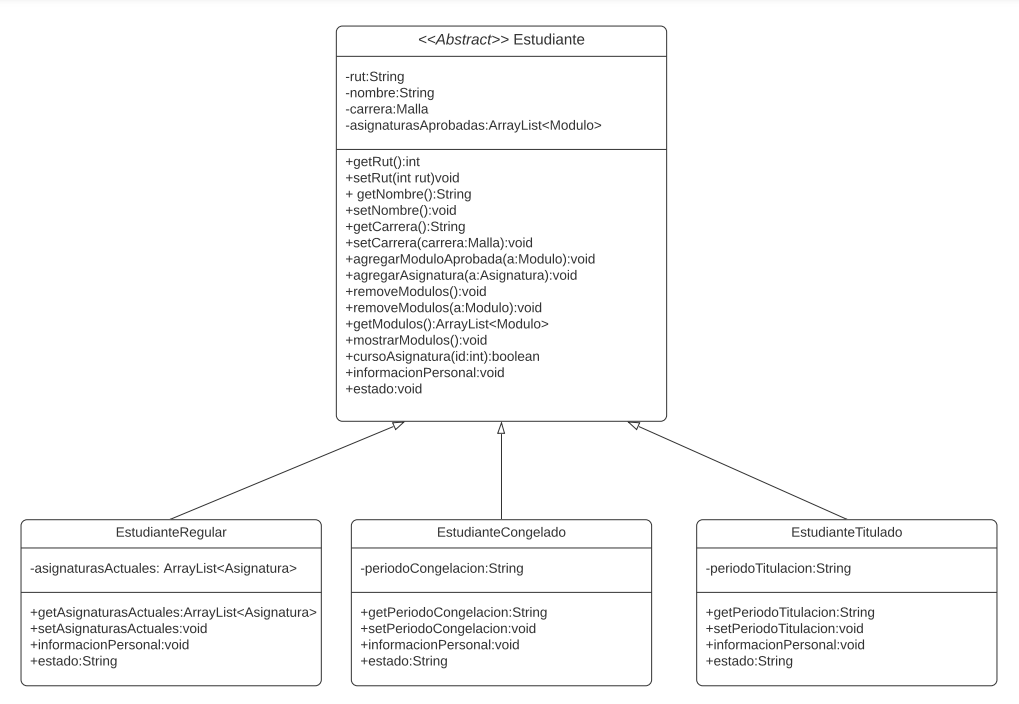
A partir de la clase Estudiante, se derivan 3 tipos de estudiante: EstudianteRegular el cual es un estudiante habilitado para tomar ramos, EstudianteTitulado el cual es un estudiante que terminó la carrera y se almacena el periodo en el que este egresó, y finalmente, EstudianteCongelado el cual es un estudiante que congeló su periodo de estudio y no está habilitado para tomar ramos.



También tenemos la clase Practica, la cual registra las horas realizadas para realizar la práctica asociada al módulo, esta clase no necesariamente es obligatoria pero es un tipo de módulo cursado por el estudiante.

1. **Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases.**

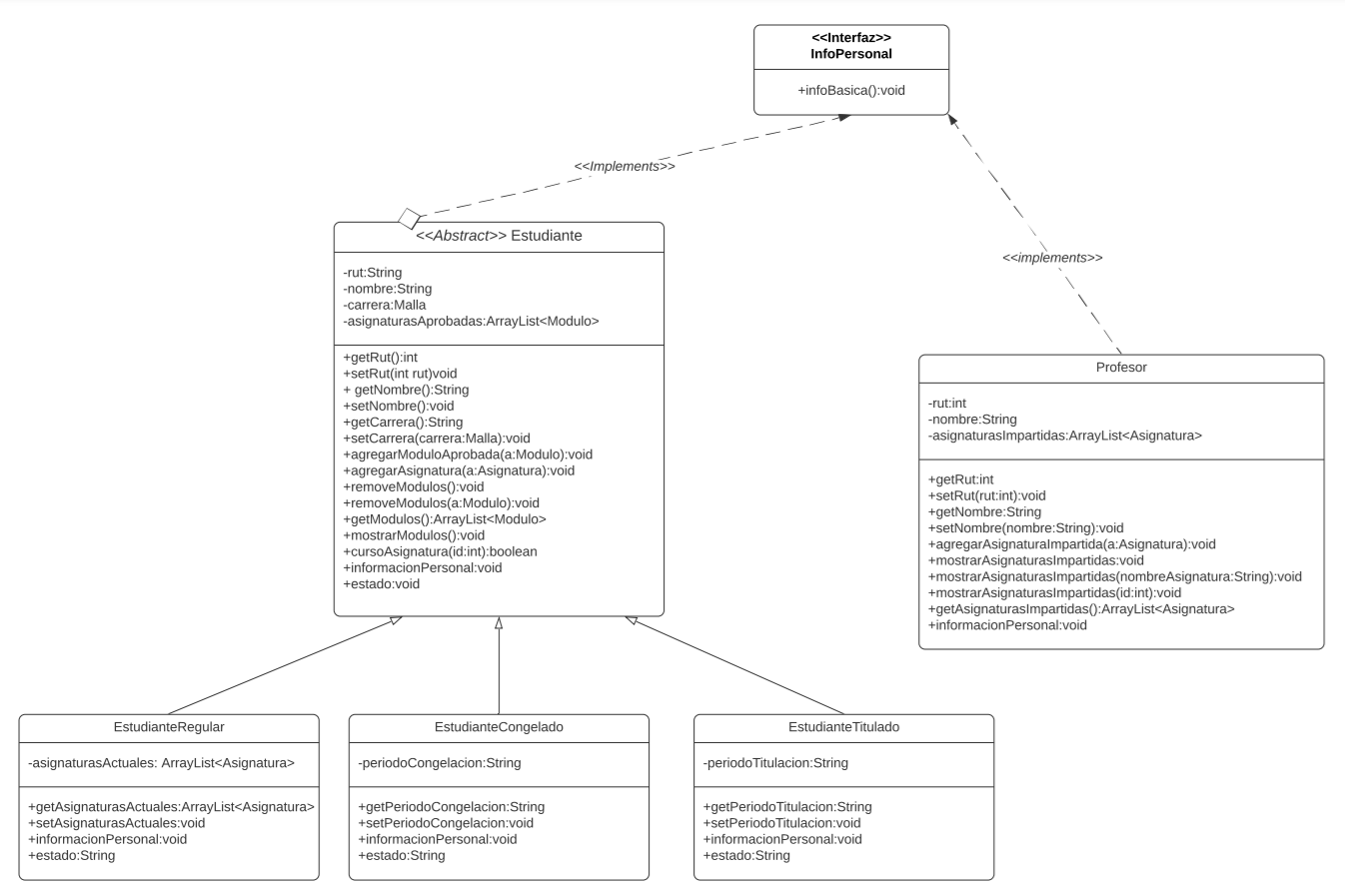
La clase abstracta debe ser utilizada por alguna otra clase (contexto)



La clase abstracta permite que el registro de estudiantes sea con un criterio dependiendo el tipo de estudiante, es decir, de este modo sabemos en qué estado se encuentra dicho estudiante. Podemos ingresar a la colección total de estudiantes según si está congelado, es alumno regular o si este ya se tituló.

1. **Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases.**

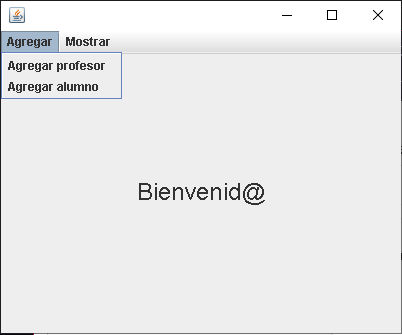
La interfaz debe ser utilizada por alguna otra clase (contexto) - (opcional) generar documentación con Javadoc.

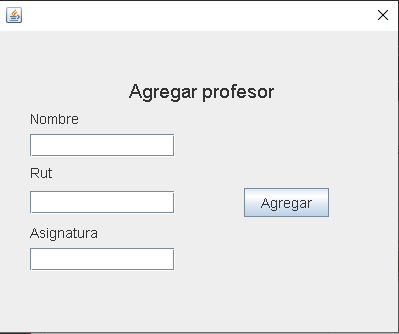


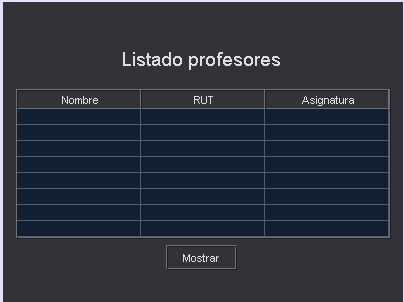
Creamos la interfaz inforPersonal la cual muestra la información personal de las clases Estudiante y Profesor que tienen implementadas el método informacionBasica que permite invocar a dicha interfaz. A traves de esta, el usuario puede visualizar la información básica de todos los estudiantes y profesores del instituto en la sección 13 del menú principal.

1. **Se deben implementar al menos 3 ventanas gráficas (GUIs en AWTo SWING):**

1 ventana de menú, 1 ventana de agregar elemento y 1 ventana de listar elementos.







Implementado en interfaz gráfica la base del Menú Principal y las funciones mostrarProfesores y addProfesores.

Momentáneamente ninguna se encuentra operativa debido a problemas con la integración de la interfaz gráfica con el resto del programa.

Link GitHub: <https://github.com/beatrizsegurap/Proyecto_PA-POO-AvanceCurricular.git>