Proyecto 1. Colecciones en Java

Calderón Jiménez, David
Hernández Olvera, Humberto Ignacio
Ordiales Caballero, Iñaky

05/Noviembre/2020

Índice general

Objetivo

Que el alumno conozca los principales aspectos teóricos y prácticos de las colecciones y sus aplicaciones en el lenguaje de programación Java, así mismo que ponga en práctica los conceptos básicos de la programación orientada a objetos y el trabajo en equipo.

Introducción

Las colecciones en Java son una estructura que proveen una arquitectura para almacenar y manipular un grupo de objetos. Estas pueden llevar a cabo todas las operaciones que se realizan en datos tales como búsqueda, ordenamiento, inserción, manipulación y eliminación.

Antes de la versión JDK 1.2 los métodos estándar para agrupar objetos en Java (o Colecciones) era con *Array, Vector* o *HashTables*. Estas no tenían una interfaz en común ni un método estándar para acceder a miembros de la colección. Es por esto que se introdujo la Interfaz de Colecciones *(Collection Framework)* en la versión JDK 1.2. La ventaja de tener esta interfaz, es que ahora todas las clases que implementan las interfaces *Collection, Set, List, Map* tiene algunos métodos en común, reducen el esfuerzo al programar debido a que no se tiene que preocupar por el diseño de la Colección sino en la utilidad de su programa, y finalmente, incrementa la velocidad y calidad del programa.

Una colección en Java significa una unidad singular de objetos. Provee interfaces tales como *(Set, List, Queue, Deque)* y clases *(ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet)*.

La paquetería **Java.util** contiene todas las clases e interfaces para el marco de Colección. De donde las que se consideran raíz de las interfaces son **java.util.Collection** y **java.util.Map**

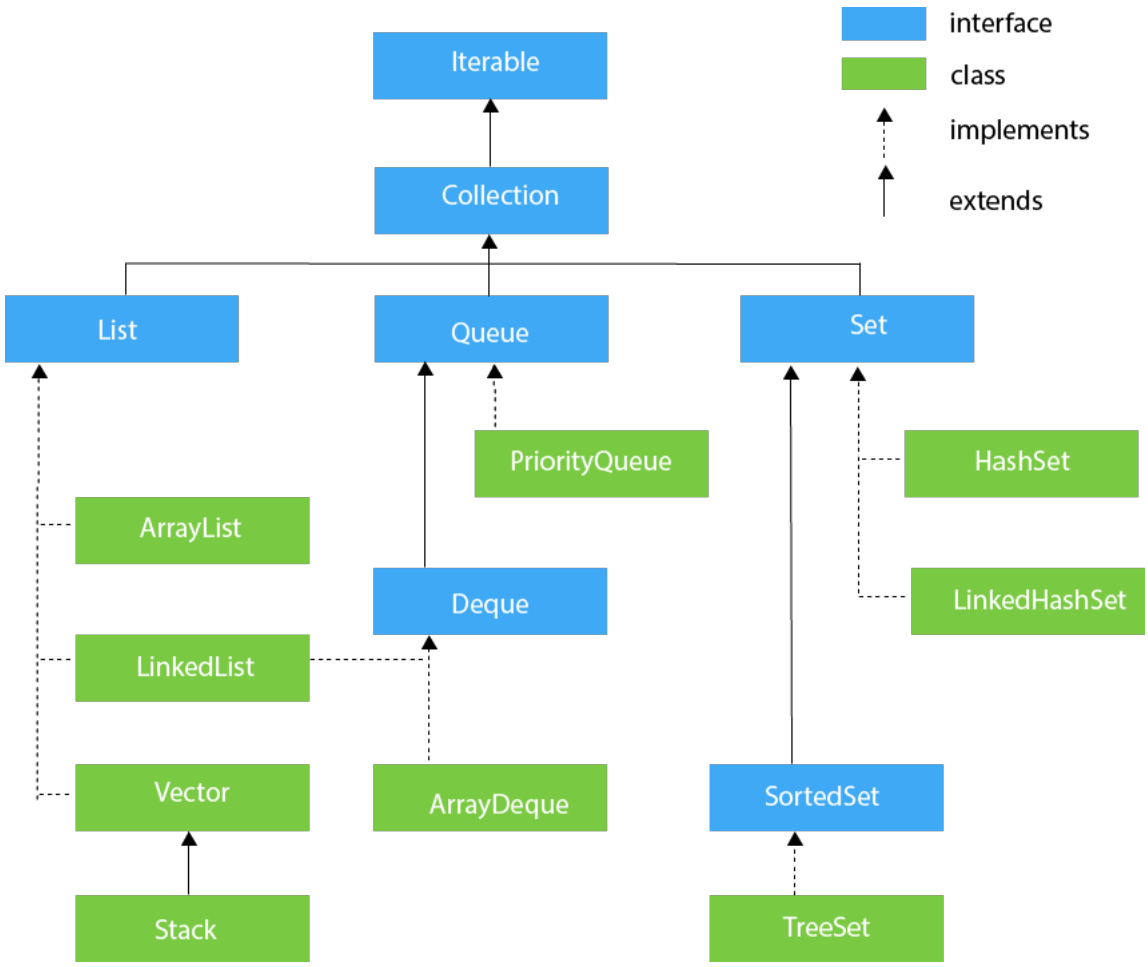

Figura 1: Jerarquía de las colecciones en Java.

Una **Clase** es un molde o prototipo definido por el usuario de donde se van a crear objetos. Representa el conjunto de propiedades o métodos que son comunes a todos los objetos de un tipo.

Una **Interfaz**, al igual que una Clase, pueden tener métodos y variables, pero estos métodos declarados son por defecto abstractos. Las interfaces especifican lo que una clase debe hacer y no cómo hacer. Es el molde de una clase.

Métodos de la Interfaz Collection

Método	Descripción
add(Objeto)	Se usa para agregar un Objeto a la Colección.
addAll(Colección c)	Agrega todos los elementos de la Colección c a la Colección dada.
clear()	Elimina todos los elementos de esta Colección.
contains(Objeto o)	Regresa *true* si la Colección contiene al elemento especificado.
containsAll(colección c)	Regresa *true* si todos los elementos de la Colección c se encuentran en la Colección dada.
equals(Objeto o)	Compara el Objeto especificado con esta Colección para encontrar una igualdad.
hashCode()	Este método se utiliza para regresar el valor de código hash para esta Colección.
isEmpty()	Regresa *true* si la Colección no contiene elementos.
iterator()	Regresa un iterador sobre los elementos en esta Colección.
max()	Regresa el valor máximo presente en la Colección.
parallelStream()	Regresa un Stream paralelo con esta Colección como origen.
remove(Objeto o)	Elimina el Objeto dado de la Colección. Si existen valores duplicados, se elimina la primera aparición del Objeto.
removeAll(Colección c)	Elimina todos los objetos de la Colección c.
removeIf(Predicate filtro)	Elimina todos los elementos de esta Colección que satisfacen el Predicate.
retainAll(Colección c)	Se utiliza para conservar solamente los elementos en esta Colección que están contenidos en la Colección especificada.
size()	Regresa el número de elementos en esta Colección.
spliterator()	Se usa para crear un Spliterator sobre los elementos de esta Colección.
stream()	Se usa para regresar un Stream secuencial con esta Colección como origen.
toArray()	Se usa para regresar un Arreglo que contiene todos los elementos de esta Colección.

Interfaces que se extienden de la Interfaz Collection

Interfaz Iterator

Provee la facilidad de iterar los elementos en dirección hacia adelante. Contiene tres métodos solamente:

1. hasNext(). Regresa true si el iterador tiene más elementos adelante, si no regresa false.
2. next(). Regresa el elemento y mueve el puntero del cursor al siguiente elemento.
3. remove(). Remueve el último elemento regresado por el iterador. (Es el menos usado)

Interfaz Iterable

Es la interfaz raíz para el marco entero de la colección. Su funcionalidad principal es proveer un iterador para las colecciones y sólo contiene un método *Iterator iterator()*.

Interfaz Collection

Esta interfaz contiene todos los métodos básicos que cada colección tiene, como por ejemplo, agregar, eliminar, eliminar todos los elementos, etc. Estos métodos se implementan en esta interfaz porque están implementados por todas las clases sin importar su estilo de implementación, y también para asegurar que los nombres de los métodos son universales para todas las colecciones. Esta interfaz crea una base a partir de la cuál las clases de la colección están implementadas.

Interfaz List

Es un derivado de la interfaz Collection. Está dedicada al tipo de dato de la lista en donde se almacenan todas las colecciones ordenadas de los objetos, permite elementos duplicados. Ejemplos de instanciación:

- List<T >[identificador] = new ArrayList<>();
- List<T >[identificador] = new LinkedList<>();
- List<T >[identificador] = new Vector<>();

En donde T es el tipo de objeto.

ArrayList

Es una clase de la interfaz List, y provee un arreglo dinámico, en donde su tamaño se redimensiona según se agreguen o quiten elementos del arreglo. Cabe mencionar que puede ser mas lento que un arreglo estándar, pero es útil para programas en donde se necesita mucha manipulación en el arreglo.

ArrayList no puede ser usado para tipos primitivos, se necesita una clase Envolvente si requerimos dicho caso.

LinkedList

Esta clase es una implementación de la estructura de datos LinkedList (Lista Ligada), en donde los elementos no están almacenados en locaciones de memoria contiguas y cada elemento es un objeto separado con una parte para datos y otra para dirección. Los elementos están ligados utilizando apuntadores y direcciones, cada elemento se conoce como nodo.

Vector

Provee arreglos dinámicos en Java. Es idéntico a un ArrayList en términos de implementación, pero la diferencia principal radica en que Vector está sincronizado y un ArrayList es no-sincronizado.

Stack

Esta clase modela e implementa la estructura de datos Stack (Pila). Está basada en el principio *last-in-first-out* (El último que entra es el primero que sale). Además de las operaciones básicas *push y pop*, la clase provee las funciones *empty, search y peek*. Esta clase también puede ser referenciada como una subclase de Vector. **Nota:** Es de hilo seguro, una contraparte es ArrayDequeue que no es de hilo seguro y con una implementación de arreglo más rápida.

Interfaz Queue

Esta interfaz mantiene el orden de *first-in-first-out* similar a una cola del mundo real. En esta interfaz importa el orden de los elementos. Ejemplos de instanciación:

- Queue<T >[identificador] = new PriorityQueue<>();
- Queue<T >[identificador] = new ArrayDeque<>();

En donde T es el tipo de objeto.

Priority Queue

Es utilizada cuando los objetos supuestamente deben ser procesados con base a una prioridad. Esta clase utiliza como base una *priority heap*. Los elementos de la cola de prioridad (Priority Queue) se ordenan acorde al ordenamiento natural, o por un Comparator proveído al momento de construcción de la Cola, dependiendo del constructor utilizado.

Interfaz Deque

Es una pequeña variación a la estructura de datos Cola. Deque es conocido como una cola doble, en donde se puede agregar y remover elementos al principio y al final de la cola. La clase que implementa esta interfaz es ArrayDeque. Se puede instanciar así:

- Deque<T >[identificador] = new ArrayDeque<>();

En donde T es el tipo de objeto.

ArrayDeque

Provee una manera de aplicar arreglos redimensionables. Es un tipo especial de arreglo que crece y permite al usuario agregar o eliminar un elemento a ambos lados de la Cola. No tiene restricciones de capacidad y crece tanto como sea necesario para soportar el uso.

Interfaz Set

Set es una colección desordenada de objetos en donde no se permiten valores duplicados, y se utiliza principalmente cuando se desea almacenar objetos únicos. Esta interfaz es implementada por varias clases como *HashSet, TreeSet, LinkedHashSet*, por mencionar algunos. Se pueden instanciar como:

- Set<T >[identificador] = new HashSet<>();
- Set<T >[identificador] = new LinkedHashSet<>();
- Set<T >[identificador] = new TreeSet<>();

En donde T es el tipo de objeto.

HashSet

Es una implementación inherente de la estructura de datos Tabla Hash. Los objetos insertados no es garantía que se inserten en el mismo orden, estos son insertados basados en su código hash y también permite elementos nulos.

LinkedHashSet

Es muy similar a HashSet, la diferencia radica en que usa listas ligadas dobles para almacenar los datos y **mantiene** el orden de los elementos.

Interfaz SortedSet

Es muy similar a Set, pero tiene métodos extra para mantener el ordenamiento de los elementos. Se extiende de Set y es usado para manejar datos que necesitan ser ordenados, la clase que implementa esta interfaz es TreeSet, y se puede instanciar como:

- SortedSet<T >[identificador] = new TreeSet<>();

En donde T es el tipo de objeto.

TreeSet

Esta clase utiliza un Árbol para almacenamiento. El orden de los elementos es mantenido por un Set utilizando el ordenamiento natural, independientemente de si se proporcionó un Comparator o no. Debe ser consistente con *equals* para implementar correctamente la interfaz Set.

Map Interface

Existe una interfaz más que se debe mencionar, y es Map.

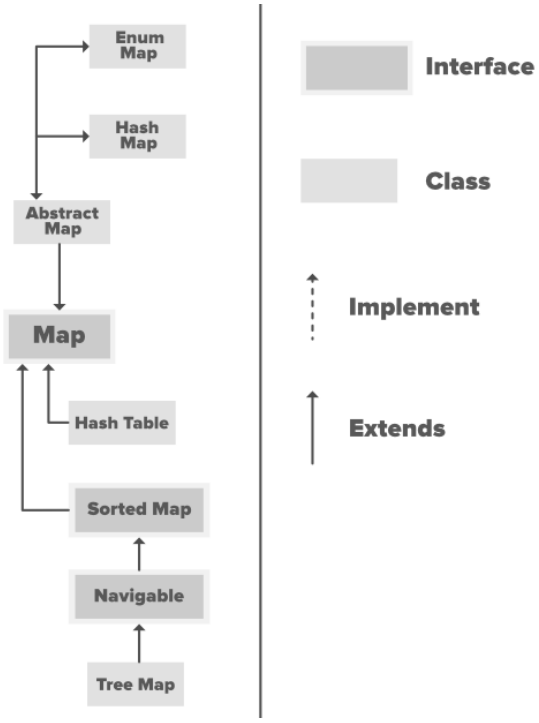

Figura 2: Jerarquía de Interfaz Map.

Map es una estructura de datos que soporta el mapeo de parejas "llave-valor" para los datos. No soporta valores duplicados, puesto que la misma llave no puede tener múltiples mapeos. Un Map es útil si se tienen datos a los que se quiere realizar operaciones basados en la llave. Esta interfaz es implementada por clases como *HashMap, TreeMap*, por mencionar algunos. Se pueden instanciar como:

- Map<T >[identificador] = new HashMap<>();
- Map<T >[identificador] = new TreeMap<>();

En donde T es el tipo de objeto.

HashMap

Provee la implementación básica de la interfaz Map. Para acceder a un valor en un HashMap, debemos conocer su llave. Utiliza una técnica llamda *Hashing*, la cuál convierte un String largo a un String corto que representa el mismo String, tal que la indexación y búsqueda son más rápidas. HashSet también utiliza HashMap internamente.

Planteamiento del problema y propuesta de solución

Para este proyecto se nos planteo un problema el cual era simular el sistema de inscripciones de la facultad de ingeniería de la UNAM todo esto implementando los conocimientos adquiridos hasta este momento de la materia de Programación Orientada a Objetos.

Este tenia que permitirnos:

⇒Registrar Alumnos.
⇒Registrar Profesores.
⇒Registrar asignaturas.
⇒Abrir un grupo nuevo y poder asignarle asignatura, profesor y un máximo de 50 alumnos.
⇒Opciones para mostrar cada uno de los datos registrados

Para proceder a plantear una solución se tomo en cuenta el mayor limitante que se tenia, el cual era el tiempo pues el proyecto era grande y el tiempo era corto.Por lo cual se planteo un diseño sencillo pero eficiente, el cual constaba en que la estructura principal fuera la clase grupo, la cual contendrá las demás clases (Una lista de Alumnos,Profesor y Asignatura) para que de esta forma en el grupo registremos a los alumnos, el profesor y la asignatura correspondientes a ese grupo en especifico.

Figura 3: Diseño base del programa

De esta forma se evitaba duplicar la creación de un profesor o asignatura y así evitar tener 2 objetos para una misma asignatura o profesor.

Funciones específicas de cada Objeto

El desarrollo del las clases principales se basaba en funciones especificas las cuales nos permitirán introducir los datos de los respectivos objetos, de esta forma tendremos una cohesión alta, con esto tendremos un manejo mas limpio con los datos para que no se mezclen la información y nuestra base de datos tenga el menor numero de errores.

Métodos y atributos de Asignatura

La estructura de esta clase es:

Figura 4: Métodos y atributos de la clase Asignatura

Los atributos como lo dicen sus nombres solo son datos de la asignatura a registrar.Los métodos sin contar los gets y los sets, solo se reduce a informacionasignatura la cual nos imprime en pantalla todos los atributos registrados en nuestro objeto de tipo asignatura.

Métodos y atributos de Profesor

La estructura de esta clase es:

Figura 5: Métodos y atributos de Profesor

Esta clase es mas complicada que asignatura pues el numero de atributos es mas extenso pero la mayoría solo son datos personales del profesor de estos vale la pena destacar:

⇒Fechas: Estas están basadas en datos generados a partir de las librerías "java.timez "java.util.Calendar".Por lo cual no solo son enteros o cadenas de texto.
⇒NumGrupo: Esta variable de tipo entero nos dice en cuantos grupos esta registrado el profesor.
⇒Clases: Esta variable es una colección implementada para que el profesor no se registre 2 veces en un grupo.
⇒ProfesoresRegistrados: Esta variable de tipo entero es estática para mostrarnos el numero de profesores que se han registrado.
Al igual que los atributos el numero de métodos es mas extenso aunque algunos de ellos solo son variaciones.Los métodos a destacar son:
⇒AsignarGrupo():Añade las claves de los grupos a los que se va asignando el profesor.
⇒MostrarProfesorClases(): Por medio de un foreach se itera la lista de las clases del profesor e imprime la clave de grupo.

Métodos y atributos de Alumno

La estructura de esta clase es:

Figura 6: Métodos y atributos del Alumno

Los atributos de esta clase consta solo de datos personales y algunos atributos a destacar son:

⇒AlumnosRegistrado: Este es un valor entero que nos indica el numero de alumnos registrados.
⇒ClavesGrupo: Esta es una lista de enteros que guarda las claves de los grupos en los que se registro el alumno.
Los métodos son mas sencillos y solo cabe destacar:
⇒AsignarGrupo: Asigna un valor entero el cual es la clave de un grupo previamente creado.
⇒MostrarGruposAlumno:Por medio de un foreach genera una variable con todos los grupos para que esta se pueda imprimir donde fue llamada.

Esta es la clase de la que mas objetos tendremos por lo cual el registro de esta tiene que ser claro, por esto tiene tantos atributos los cuales podríamos unir algunos de ellos pero para evitar problemas los dejaremos de esta manera para que tengamos un manejo de datos mas limpio en cada uno de los objetos creados.

Métodos y atributos de Grupo

La estructura de esta clase es:

Figura 7: Métodos y atributos del Grupo

Esta al ser la clase en la que estaremos manejando el mayor numero de datos, es la que tiene mas datos abstractos como atributos.Los que cabe destacar de aquí son:

⇒GruposRegistrados: Esta es una variable de tipo entero que registra cuantos grupos se han creado.
⇒AlumnosRegistrados: Esta es una variable de tipo entero que registra cuantos alumnos hay en ese grupo.
⇒ListadeAlumnos: Esta es una lista de objetos de la clase alumno en la cual guardaremos a todos los alumnos que se registren en este grupo, la cual limitaremos a un cupo máximo de 50 alumnos.
⇒Asignatura: Esta es una variable de tipo Asignatura donde se guardara la asignatura asignada a este grupo.
⇒Profe: Esta es una variable de tipo Profesor donde guardaremos a el profesor asignado a este grupo.
⇒Clave: Este es un valor de tipo entero que contendrá la clave (numero) del grupo.

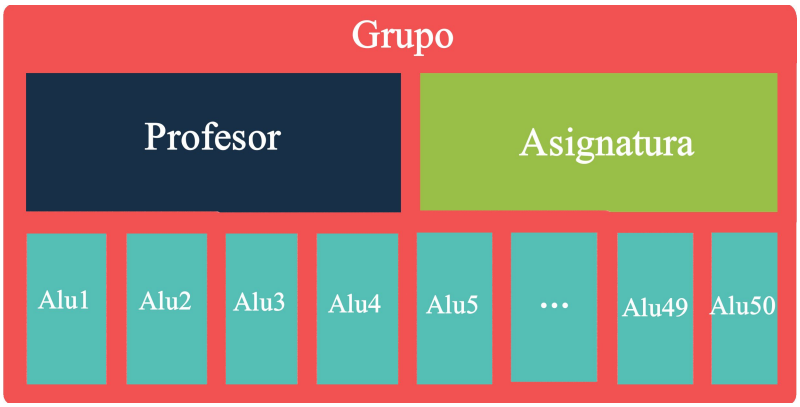

Figura 8: Objetos contenidos en la clase Grupo

En la parte de los métodos la mayoría sirven para asignarle un objeto a nuestro grupo, el método que mas cabe destacar es:

⇒AsignarAlumnos: Este método sirve para ir asignando los objetos de tipo alumno a una lista de este mismo tipo, para que de esta forma sea mas fácil llevar un control de los alumnos.

Con la base de nuestro programa definida solo hacia falta definir de que manera leer los datos y levar un mejor control de la información.

Lectura y manejo de la información

Al tener la estructura del proyecto solo hacia falta definir de que manera leeríamos los datos y de que forma manejaríamos esa información.Para ello nos apoyamos en una clase nueva que resulto ser uno de los pilares de nuestro programa.

Esta clase es utilerías y sus funciones son:

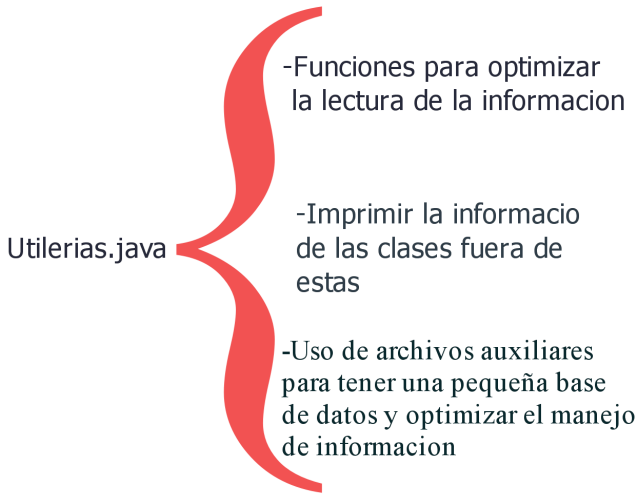

Figura 9: Funciones de Utilerías

A partir de esta clase pudimos optimizar el funcionamiento del programa pues teníamos una función clave la cual era almacenar los datos previamente registrados para que así se pudiera manejar la información aunque el programa se reiniciara.

Implementación de colecciones

Uno de los requisitos de este proyecto era que debíamos implementar una colección, una lista y una tabla hash.Estas implementaciones se tenían que hacer considerando en que momento era mas óptimo utilizar uno u otro.

Nuestra implementación fue de la siguiente forma:

Colección	Función	Importancia
Lista	Utilizada en las listas de alumnos y profesores.	Nos permiten crear un conjunto ordenado de datos y fue implementado en estas funciones porque no tenían ningún limitante extra
Colecciones	Implementada en la lista de grupos de la clase profesor.	Esta fue implementada aquí porque el programa fallaría si se repiten 2 claves iguales en los grupos del profesor

HashMap	Implementada en la lista de clases y asignaturas.	Implementada aquí para que por medio de las claves de alumno o asignatura pudiéramos encontrar el objeto asociado a estas

Corrección de errores

Aunque el diseño del programa estaba claro y la estructura de este era buena, se presentaron algunos errores a los cuales se corrigieron poco a poco.

El error principal que nos costó mucho trabajo corregir fue el siguiente:

Error al no inscribir ningún alumno en un grupo creado

Este error se presento debido a que si no se inscribía ningún alumno al grupo, al momento de escribir ese grupo en el archivo y volverlo a leer, este no detectaba a todo el grupo por la ausencia de alumnos inscritos.

La solución de este error no fue nada fácil pero llegamos a la conclusión de que la existencia de un grupo sin alumnos no tenia sentido, a lo cual tomamos la decisión de que el programa al detectar un grupo sin alumnos no lo tomaría en cuenta al momento de escribir la información en los archivos auxiliares.De esta forma el grupo desaparecería y al momento de leer la información ya no marcaría error el programa.

Funcionamiento final del proyecto

El uso del proyecto esta especificada en el manual de usuario pero aquí veremos su funcionamiento para poder ver si los objetivos se cumplieron.

Menú principal

Aquí nos muestra las funciones principales del programa pues, en un sistema de inscripciones profesional no tienen los mismos permisos los alumnos. profesores o técnicos.

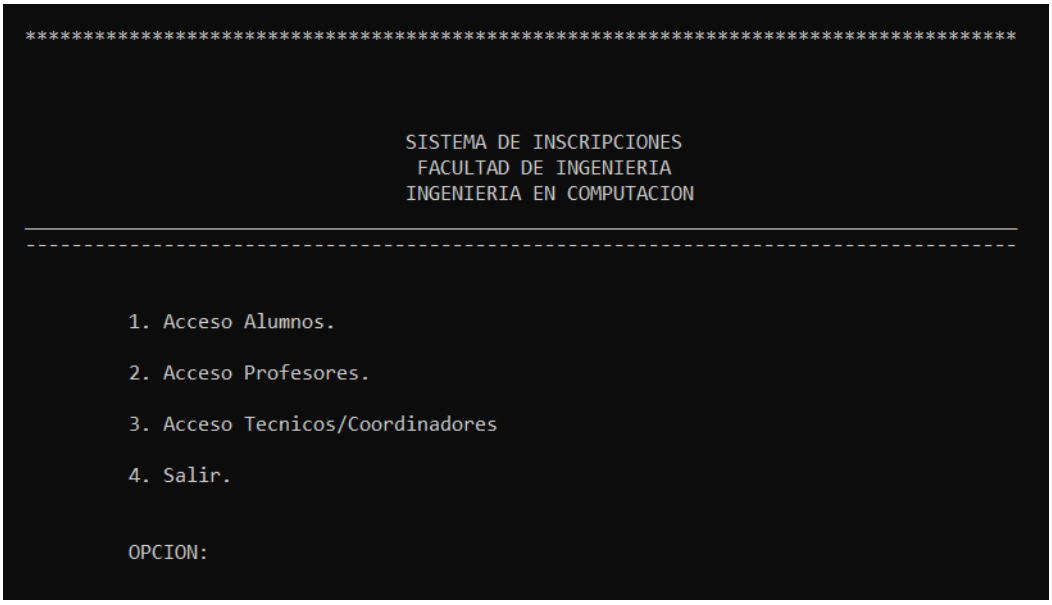

Figura 10: Menú Principal

Acceso Alumnos

Este menú nos muestra las funciones a las que tiene acceso un alumno.

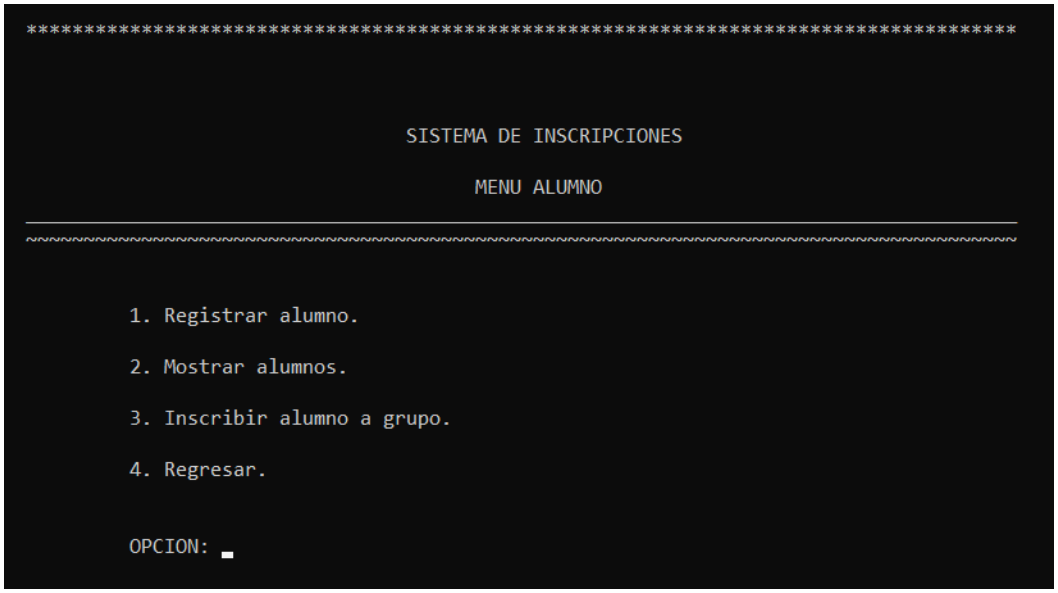

Figura 11: Menú para Alumnos

1.-Registrar alumno
Este menú nos permite registrar un alumno para ser usado en registros posteriores.

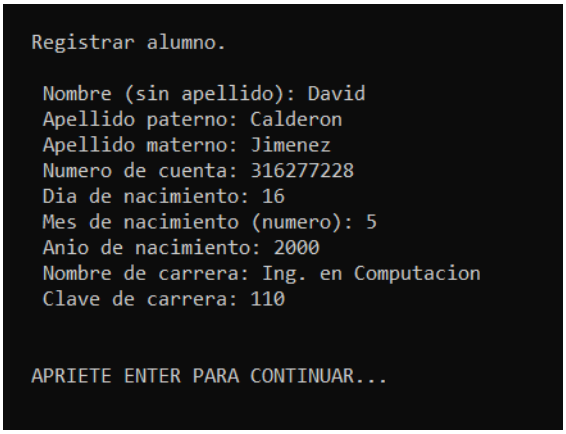

Figura 12: Registro de alumnos

2.-Mostrar alumnos
Esta función nos muestra a los alumnos registrados junto con toda su información.

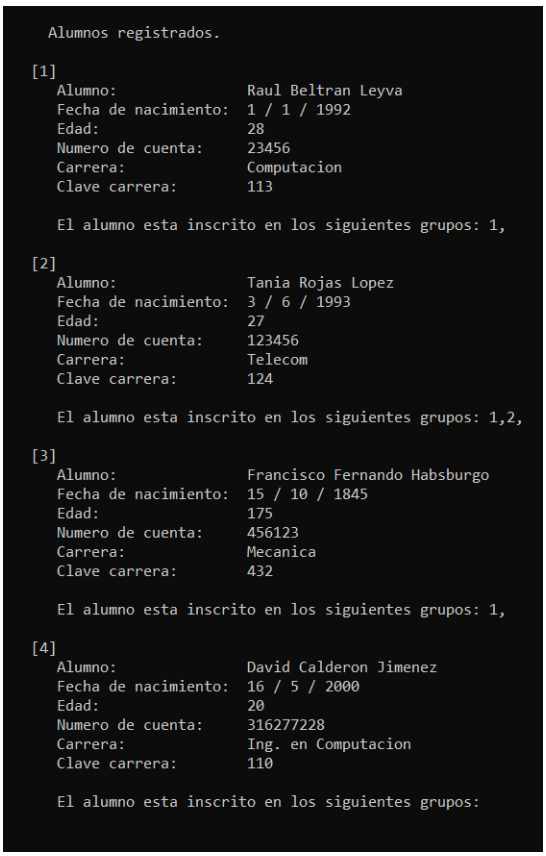

Figura 13: Alumnos Registrados

3.-Inscribir alumno a grupo
Esta función es para asignarle un grupo en especifico al alumno, cabe aclarar el el profesor Tista aparece 2 veces pero solo esta registrado una vez esto se puede gracias a que un profesor puede tener mas de 1 grupo.

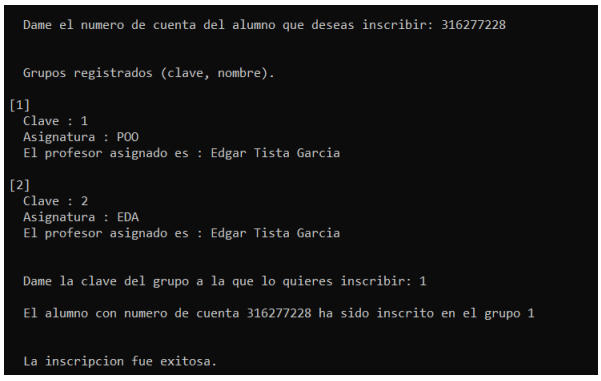

Figura 14: Inscripción del alumno al grupo del profesor Tista (No lo haga compa)

Acceso Profesor

El menú del profesor es limitado pues solo le permite registrarse y ver que otros profesores hay, lo cual deja al profesor bajo las decisiones de los técnicos para asignarle grupo.

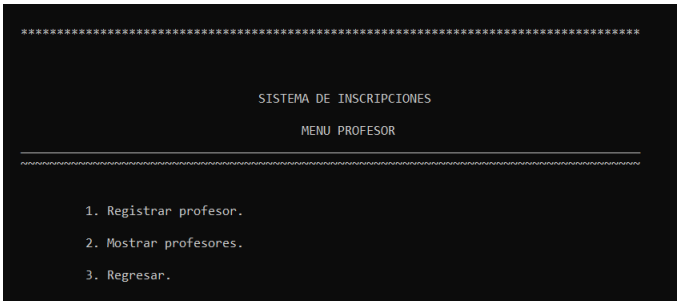

Figura 15: Menú para profesores

1.-Registrar profesor.
Esta función nos permite hacer el registro de un nuevo profesor.

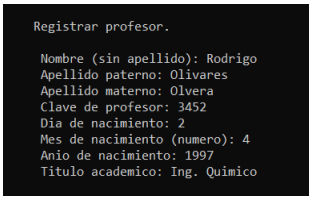

Figura 16: Registro de un profesor

2.-Mostrar Profesores
Esta función solo nos muestra a los profesores registrados

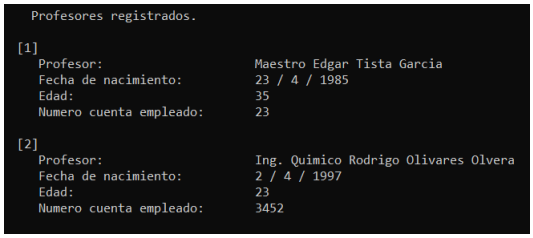

Figura 17: Muestra a los profesores registrados

Acceso Técnicos

Este es el usuario con mas permisos por lo cual nos pedirá una contraseña para ingresar a este perfil.

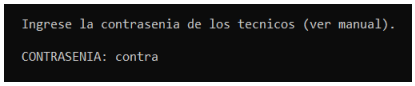

Figura 18: Acceso por contraseña para técnico

Una vez dentro tendremos acceso a este menú:

Figura 19: Menú para el técnico

1.- Crear asignatura
Esta función nos ayuda a crear una nueva asignatura.

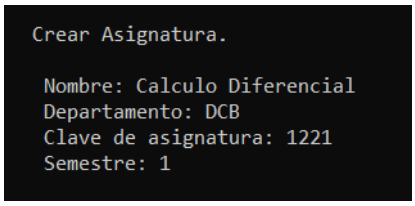

Figura 20: Crea una asignatura

2.-Crear Grupo
Esta función es de las mas importantes pues nos ayuda a crear un nuevo grupo

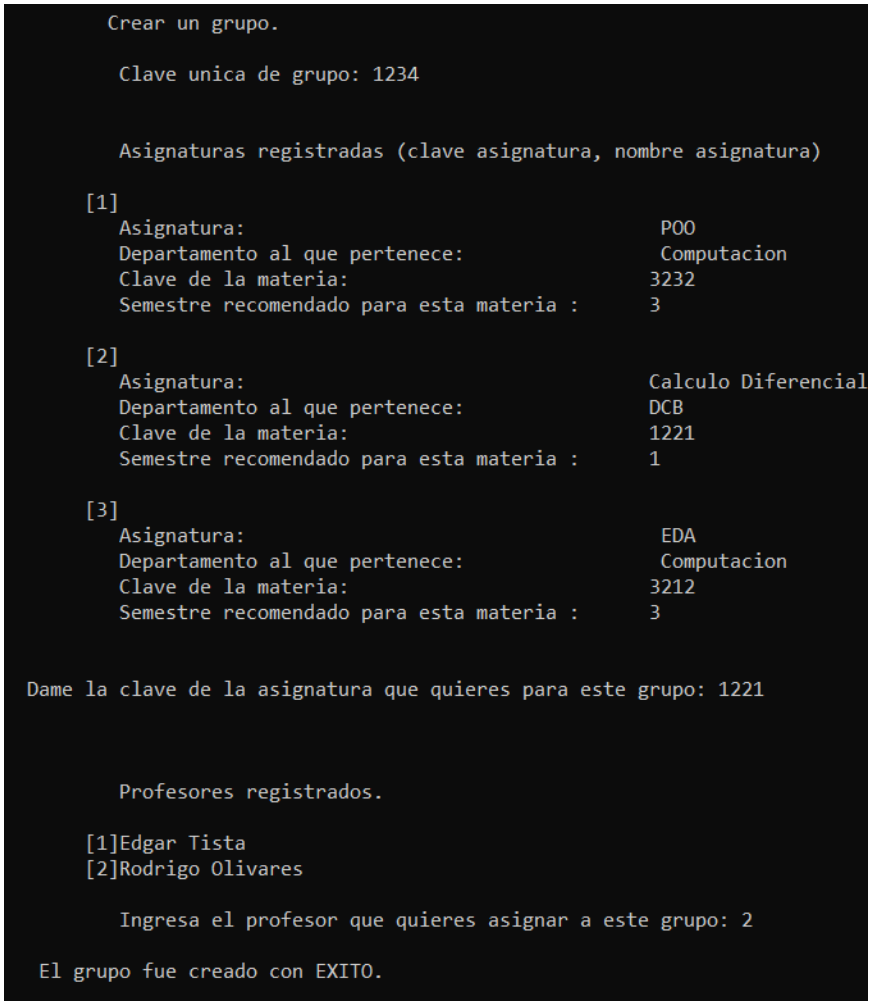

Figura 21: Crea un grupo

3.-Mostrar asignaturas
Muestra las asignaturas registradas.

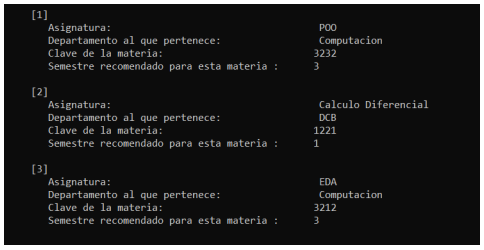

Figura 22: Asignaturas Registradas

4.-Mostrar grupos
Muestra los grupos registrados.

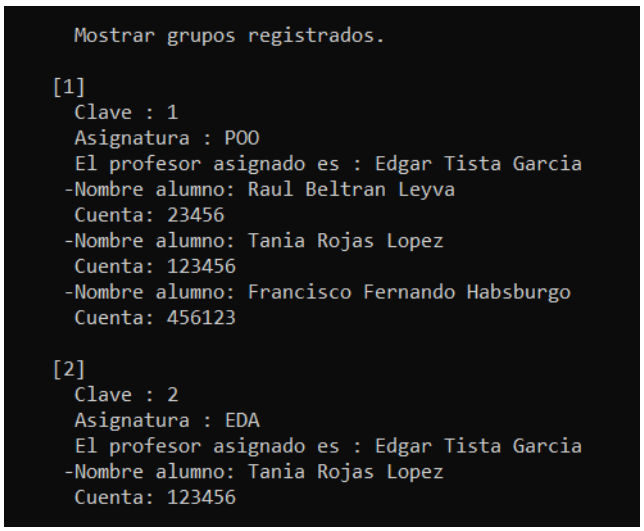

Figura 23: Grupos Registrados

Conclusiones

Calderón Jiménez, David

El desarrollo del proyecto se sintió como algo mas profesional pues no solo fue escribir código día y noche, este tuvo un avance poco a poco, iniciando primero con un diseño del cual partió todo con esto y un objetivo claro el desarrollo se fue haciendo mas solido. A partir de esto el proyecto empezó a presentar dificultades pero estas se resolvían de buena manera gracias a que todos empezamos a conocer nuestro código hasta entender las funciones mas profundas.

La parte mas importante de este proyecto fue tener una buena comunicación como equipo pues con la situación actual esto era mas difícil, para mi el punto clave que nos ayudo fue que con mi equipo compartía otro proyecto de otra materia diferente pero que va muy relacionada.Gracias a esto nos adaptamos más rápido a nuestra forma de trabajar y la solución de los problemas que nos tocaba a cada uno era más rápida y constante.

Finalmente de mi parte puedo decir que aplicamos de muy buena manera los conocimientos de la programación orientada a objetos más un trabajo bueno y solido en equipo podemos concluir que los objetivos fueron cumplidos.

Hernández Olvera, Humberto Ignacio

Realizar este proyecto, te hace sentir en otro nivel, porque es distinto a las prácticas de laboratorio, a nivel de escala. Aunque a *groso modo* es similar: crear un programa para resolver una problemática y generar un documento escrito que exponga el razonamiento y metodología para la solución del problema, en esta ocasión se multiplicó como por 100. Porque el programa requiere un acercamiento al problema desde un punto de vista primero general, y luego particular. Cosa que en lo particular, creo que no estamos acostumbrados a hacer en las prácticas.

Nos juntamos en *Discord* para discutir cómo íbamos a abordar la situación, dividimos el problema en problemas más pequeños, estando en comunicación constante para ayudarnos lo más posible, también con el propósito de estar al corriente con los cambios que se estaban realizando en el programa. Al principio nos costó trabajo adecuarnos, pero logramos encontrar nuestro ritmo de trabajo a lo largo de los días de desarrollo del programa.

Una vez que tuvimos armado el programa, y al estar haciendo pruebas semi-exhaustivas, encontramos los casos particulares, que si no se inscriben alumnos al grupo, que si el profesor no tiene grupo asignado, el manejo de errores también lo estuvimos verificando, situaciones con el manejo de archivos que en ocasiones se borraban los datos çomo por arte de magia". Modularizamos los más posible el programa, de tal forma que sea algo general, no tan particular, pero algunas cosas son particulares para este programa, como el formato de escritura en los archivos.

Terminar el proyecto es una sensación de alivio, aunque hay algunos pequeños casos particulares que no pudimos resolver y que mencionamos en el manual, pero eso no quita la sensación de logro. Complementamos nuestro aprendizaje con este tipo de proyectos, puesto que no solo utilizamos lo visto en clase, también investigamos en línea, en tutoriales (tanto en video como escritos) para apoyarnos con algunas situaciones que nos estaban dando dolor de cabeza.

Después de esta experiencia, considero que estoy más familiarizado con el paradigma orientado a objetos, y principalmente con el lenguaje Java, la forma en como se crean objetos, añadirle atributos, métodos que utilicen dichos atributos, cómo acceder a estos métodos desde otra clase, personalmente no me gusta la palabra *static*, porque como dijimos en clase, le quita algo de la esencia a crear objetos. Espero con ansias el siguiente proyecto para poder reutilizar lo aprendido en éste.

Ordiales Caballero, Iñaky

No es muy difícil escuchar a profesores, compañeros o inclusive familiares decir que realmente los estudios sólo son la base para tu desarrollo profesional, que no será hasta que empieces a trabajar cuando realmente aprendas cómo se deben hacer las cosas. Yo no puedo decir por cuenta propia que esto sea real, sin embargo entiendo por qué podría suceder y se me hace lógico. Al hacer una comparativa a mucho menor escala, siento que los proyectos de programación tanto este, como los de asignaturas anteriores, es donde uno empieza a darse cuenta y a enfrentarse con los problemas y dificultades de la elaboración de Software real. No es sino hasta que te dan un problema un poco más complejo y aplicado que los de clases, cuando nos damos cuenta que en muchas ocasiones no entendemos el funcionamiento de cierta característica o concepto tan bien como creíamos. Al llevar a cabo la elaboración de este tipo de proyectos es cuando aprendemos enormemente y acabamos de afianzar los conceptos estudiados.

El proceso de elaboración del proyecto por si sólo no fue tan complicado, pero llegó un punto en el que sí sentíamos que el tiempo se nos empezaba a acabar. El trabajar en equipo no siempre es lo más sencillo, y hacerlo totalmente en línea agregó cierta dificultad extra a todo el proceso. Sin embargo conforme logramos ponernos de acuerdo para irnos reuniendo en videollamadas, las cosas se fueron aclarando y pudimos crear una mejor distribución del trabajo. Creo que en todo momento el equipo se comportó a la altura del proyecto, nos ayudábamos y checábamos el trabajo del otro. Sin duda en los proyectos es cuando nos damos cuenta de lo beneficioso que puede ser trabajar en equipo. Uno sólo aún con toda la ayuda de los foros en Internet no podría hacer el trabajo de un equipo bien compenetrado. No por esto me refiero a que no hayamos tenido dificultades, porque sí las hubo. Pero lo importante es que mediante la comunicación no pudimos poner de acuerdo para acabar con un programa hecho por los tres y totalmente funcional.

Ahora, concluyendo más enfocado hacia las codificaciones y los elementos del paradigma orientado a objetos que utilizamos, podemos decir que el proyecto nos dio la oportunidad de juntar todo lo visto en la clase de teoría ahora de forma práctica. El uso de las clases para la abstracción del problema queda bastante claro al tratarse de un problema al cual estamos tan cercanos. El tener como requerimiento el implementar las estructuras listas, conjuntos y tablas hash nos hizo pensar mucho en las interacciones entre clases y nos hizo aprender a sacarle provecho a la API de Java para poder usar los métodos de cada clase. Realmente el programa se fue armando poco a poco con múltiples pruebas de ejecución para cada particularidad que se quería hacer funcionar. El trabajo fue cansado, pero finalmente se acabo con un buen programa.

Fuentes de información

[1] GeeksForGeeks. *Collections in Java.* URL: https://www.geeksforgeeks.org/collections-in-java-2/ (visitado 02-11-2020).

[2] GeeksForGeeks. *Different ways of Reading a text file in Java.* URL: https://www.geeksforgeeks.org/different-ways-reading-text-file-java/ (visitado 25-10-2020).

[3] GeeksForGeeks. *Set in Java.* URL: https://www.geeksforgeeks.org/set-in-java/ (visitado 25-10-2020).

[4] API Java. *Class HashMap¡K,V¿.* URL: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html (visitado 28-10-2020).

[5] API Java. *Interface List¡E¿.* URL: https://docs.oracle.com/javase/8/docs/api/java/util/List.html (visitado 28-10-2020).

[6] JavaTPoint. *Collections in Java.* URL: https://www.javatpoint.com/collections-in-java (visitado 02-11-2020).

[7] tutorialspoint. *How to create a new directory by using File object in Java?* URL: https://www.tutorialspoint.com/how-to-create-a-new-directory-by-using-file-object-in-java (visitado 25-10-2020).