Universidad Nacional Autónoma de México
Facultad de Ingeniería

Proyecto 1: Colecciones en Java

Programación Orientada a Objetos

Grupo 03

Equipo 1

Martínez Olmos Osiris

Rosales López Luis André

Núñez Quintana Luis Axel

Ciudad de México 05 de octubre, 2020

Índice general

Objetivo General

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 a distancia.

Objetivos Específicos

- Realizar una investigación sobre las principales colecciones disponibles en Java y cuando se debe usar una u otra.
- Elaborar un programa que simule un sistema de inscripciones donde se implementen los principales tipos de colecciones y se siga el paradigma de Programación Orientada a Objetos.
- Realizar un análisis del código generado indicando por qué se eligió la estructura dada al programa y cómo se seleccionaron los tipos de colecciones implementados.
- Generar un manual de usuario que describa cómo se debe usar el programa para que no se generen errores en la ejecución.

Marco teórico

Principales Colecciones en Java

En Java, una colección es una representación de un conjunto de objetos. Los objetos se guardan dentro de la colección y se convierten en los elementos de la misma. La interfaz genérica de Java, Collection, se usa con el propósito de trabajar con colecciones y realizar diferentes acciones con ellas como añadir un elemento, eliminar un elemento, obtener el tamaño de la colección, etc. La interfaz Collections contiene un conjunto de clases e interfaces del paquete java.util para gestionar colecciones de objetos. Los principales tipos de colecciones incluidas por defecto en Java son: Sets(Conjuntos), Lists(Listas) y Maps. Cada uno de estos tipos cuenta con diferentes implementaciones y características diferentes que nos ayudan a elegir cuando es más apropiado usar una u otra. A continuación se describen más a detalle los tipos mencionados.

SET

Los tipos de colecciones Set se caracterizan por no permitir contener elementos duplicados o repetidos. Los Sets cuentan con todos los métodos heredados de la interfaz Collection añadiendo la restricción de que los elementos repetidos no están permitidos.

La interfaz Set tiene diferentes tipos de implementaciones. Ninguna de las implementaciones que a continuación se enlistan son sincronizadas, por lo que no se asegura el estado del Set cuando dos o más hilos acceden al mismo tiempo a este. Las implementaciones son:

- **HashSet:** Almacena lo elementos en una tabla hash. Tiene el mejor rendimiento de todas las implementaciones, pero no garantiza ningún orden cuando se realizan iteraciones. Se usa ampliamente gracias a su buen rendimiento y cuando no es necesario establecer un orden de los elementos. El tamaño inicial de la tabla definirá el rendimiento de la implementación.

- **TreeSet:** Los elemento guardados se ordenan en función de su valor. Es bastante más lento que HashSet. Los elementos almacenados deben implementar la interfaz Comparable. Esta implementación garantiza, siempre, un rendimiento de log(n) en las operaciones básicas, debido a la estructura de árbol empleada para almacenar los elementos.

- **LinkedHashSet:** Esta implementación almacena los elementos en función del orden de inserción. Es, simplemente, un poco más costosa que HashSet.

LIST

Esta interfaz define una secuencia de elementos. Acepta elementos duplicados o repetidos y cuenta con los métodos heredados de Collection, además de otros que permiten acceder a los elementos de la lista a partir de su posición, buscar elementos de la lista y obtener su posición, iterar sobre los elementos y realizar operaciones sobre un rango determinado de los elementos.

Las implementaciones de la interfaz List, al igual que las implementaciones de la interfaz Set, tampoco son sincronizadas. Las implementaciones son:

- **ArrayList:** Es una implementación muy común. Consiste en un array que se puede redimensionar aumentando su tamaño conforme se agregan elementos a la colección. En la mayoría de los casos es la que mejor rendimiento presenta.

- **LinkedList:** En ocasiones presenta una mejora del rendimiento. Esta implementación se basa en una lista doblemente enlazada de los elementos, teniendo cada uno de los elementos un puntero al anterior y al siguiente elemento.

MAP

La interfaz Map consiste en la asociación de claves y valores. Se caracteriza por que no permite claves repetidas o duplicadas y cada clave tiene asociado sólo un valor. Map cuanta con diferentes implementaciones, las cuales al igual que las ya mencionadas, no son sincronizadas. Las implementaciones de esta interfaz son:

- **HashMap:** Almacena las claves en una tabla hash. No asegura ningún orden al momento de iterar sobre la misma y presenta el mayor rendimiento de todas. Esta implementación proporciona tiempos constantes en las operaciones básicas siempre y cuando la función hash disperse de forma correcta los elementos dentro de la tabla hash. Es importante definir el tamaño inicial de la tabla ya que este tamaño marcará el rendimiento de esta implementación.

- **TreeMap:** Esta implementación almacena las claves ordenándolas en función de sus valores. Es bastante más lento que HashMap. Las claves almacenadas deben implementar la interfaz Comparable.
- **LinkedHashMap:** Esta implementación almacena las claves en función del orden de inserción. Es, simplemente, un poco más costosa que HashMap.

Elección de la Colección más adecuada

Las estructuras mencionadas ofrecen diversas funcionalidades como: ordenación de elementos, mejora de rendimiento, rango de operaciones, etc. Es importante conocer cada una de ellas para saber cuál es la mejor situación para utilizarlas. Un buen uso de estas estructuras mejorará el rendimiento de nuestra aplicación.

Para conocer qué tipo de colección usar, podemos emplear el diagrama de la Figura 1:

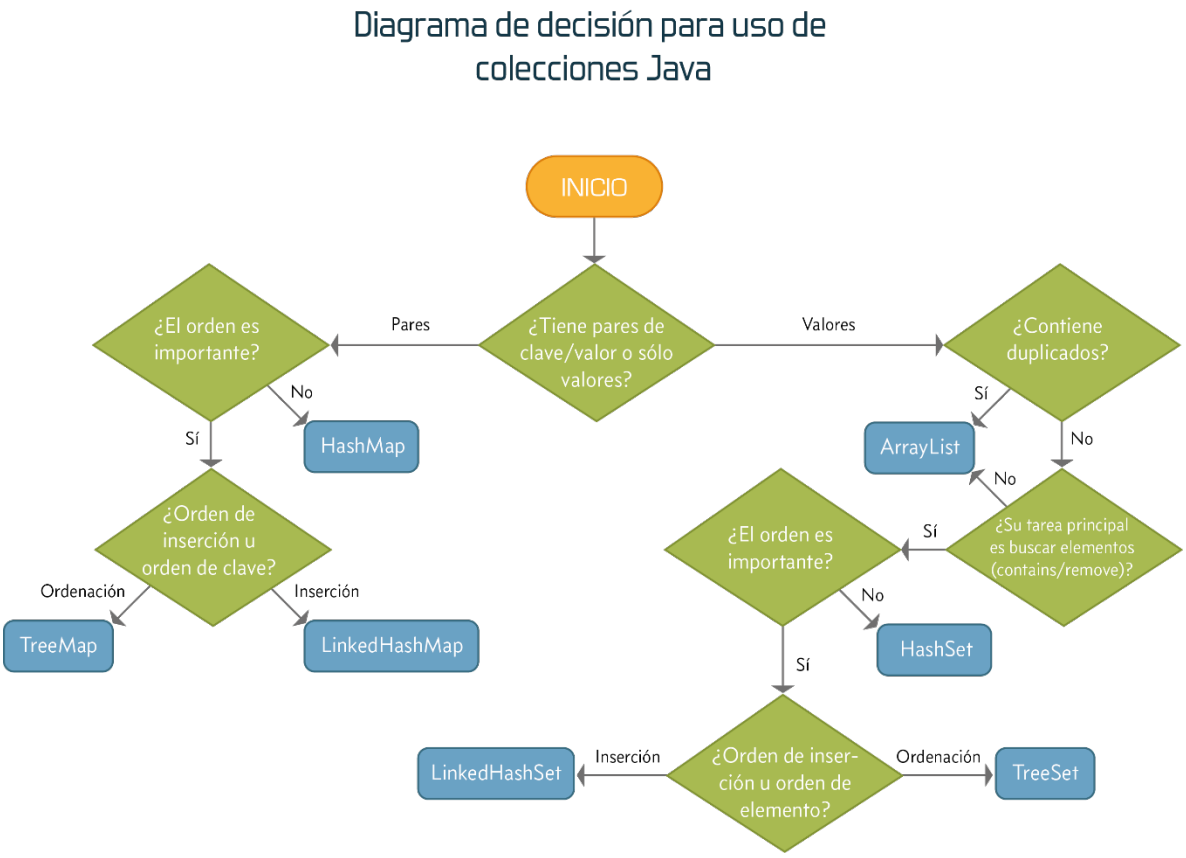

Figura 1: Diagrama de decisión para elegir el tipo de colección más adecuando según las características de los elementos que se almacenarán, el orden de los elementos y la forma en que se puede acceder a los elementos. Recuperada de: Vindel Amor, R., 2015, Introducción a Colecciones en Java, www.adictosaltrabajo.com

Análisis y Desarrollo

Para la elaboración del programa que simula un sistema de inscripciones y con el objetivo de seguir el paradigma de programación orientada a objetos, se modelaron diferentes clases que modelan a los objetos que intervienen en el método principal. Estas clases son: Alumno, Profesor, Asignatura y Grupo.

Se podría decir que las clases mencionadas son los principales factores involucrados en el Sistema de Inscripciones creado. Tienen ciertas relaciones entre sí y también comparten una estructura similar. A continuación se detallará sobre las relaciones que hay entre ellas, la estructura general de estas clases y las diferencias que hay entre unas y otras.

Relaciones entre las Clases

La relación entre las clases se basa en que un grupo debe tener un conjunto de alumnos inscritos, un profesor asignado y la asignatura que se imparte. Grupo es donde se conjugan todas las demás clases.

Mientras tanto, Asignatura tiene una lista de grupo en donde se imparte esa asignatura y no hay un límite de grupos en donde se imparte la misma asignatura. Los objetos Profesor tendrán una lista de los grupos en los que da clase dicho profesor, y puede dar clase a más de un grupo de la misma asignatura. Y la Clase Alumno cuenta con un arreglo de los grupos a los que está inscrito (máximo 3 grupos y deben ser grupos diferentes).

Cabe mencionar que cada clase tiene una clave (atributo) que le sirve como identificador, Alumno se identifica con su numero de cuenta, Profesor se indentifica con su clave de profesor, Grupo tiene una clave única y Asignatura tiene una clave de asignatura. En lugar de trabajar directamente con objetos se trabajó con estas claves, entonces, por ejemplo profesor tiene una lista de las claves de grupo y no directamente una lista de objetos grupo.

Estructura General de las Clases

Todas las clases cuentan con cierto número de atributos, a dichos atributos se les asignó el modificador de acceso private, con el motivo de que sólo se puedan acceder mediante los métodos setters y getters que cada clase tiene.

Ninguna clase ocupa el constructor vacío, todos los contructores solicitan inicializar por lo menos uno de los atributos. Los atributos que se incializan en el constructor los llamamos "atributos básicos", pues son los datos indispensables para poder crear una instancia de dicha clase. Usualmente estos atributos son nombre, grado académico, correo, etc. dependiendo de la clase. También se creó un método en cada clase que imprime todos los atributos básicos de cada clase.

Como ya se mencionó, todos los atributos tienen sus respectivos setters y getters. Cada setters tiene una restricción para poder inicializar el atributo correspondiente. Por ejemplo, el método setNombre sólo incializa el atributo nombre cuando recibe una cadena con más de dos caractéres y menos de 50 caractéres.

Un aspecto importante es que cada clase tiene un atributo que es una colección, el tipo varía dependiendo la clase, pero todas incluyen un atributo que guarda un conjunto de datos. Se profundizará más acerca de estos atributos en la siguiente sección. Por lo tanto, las clases

también incluyen un método que agrega un elemento a la respectiva colección y otro método que imprime cada elemento de la colección.

Diferencias entre las Clases

La principal diferencia en la estructura de una clase y otra, está dada por el tipo de colección que cada una implementa.

La clase Profesor implementa una lista de cadenas donde guarda las claves de los grupos en los imparte clase dicho profesor, y la clase Asignatura tiene una lista de cadenas donde se almacenan todas las claves de los grupos donde se imparte dicha asigantura. Miestras tanto. Para estas dos clases se eligió usar listas debido a que no hay restricción en el número de grupos que puede impartir un Profesor, ni en el número de grupos en los que se imparte una Asignatura.

La Clase grupo cuenta con set de numeros tipos long, los numeros representan los números de cuenta de los alumnos inscritos a dicho grupo. Se decidió usar un set porque no acepta valores repetivos y un grupo no puede tener inscrito al mismo alumno dos veces.

A diferencia de las clases anteriores, la clase Alumno utiliza un arreglo de cadenas de tamaño 3 que contiene las claves de los grupos a los que está inscrito el alumno. Se decidió usar un arreglo de tamaño definido porque un alumno sólo puede estar inscrito en 3 grupos.

Menú prinicipal (método Main)

El método main del programa se encuentra dentro de la clase ProyectoColeccionesEnJava, este método es el corazón del sistema de inscripciones y es donde se encuentran las colecciones con los objetos que intervienen en el proceso de inscripción y el menú prncipal.

En este método se almacenan todos los objetos que se van creando durante la ejecución del programa dentro de Tablas Hash. Las tablas Hash relacionan a los objetos (Alumnos, Profesores, Asignaturas, y Grupos), que representan los values dentro la tabla, con las claves o identificadores de cada objeto (numero de cuenta, numero de cuenta o clave de profesor, clave de asignatura y clave de grupo), que representan las llaves de la tabla.

Por lo tanto hay 4 tablas hash que durante la ejecución son constantemente recorridas, se les agregan elementos, se verifica si un elemento ya existe dentro de la misma, se busca a un elemento específico mediante una llave, etc. según la opción seleccionada en el menú principal.

El menú prinicipal es una estructura switch case. Se evalua el valor de la variable selector, que está incializada en 0 provocando que al ingresar al programa se reprodusca el primer case que es el menú prinicipal donde se indican los tamañanos de las tablas hash, es decir se muestra cuantos alumnos, profesores, asignaturas y grupos se han creado. Al iniciar el programa todas marcaran 0. También se muestras las posibles opciones, que son las diferentes entidades sobre las que se puede trabajar (Alumnos, Profesores, Asignaturas o Grupos).

Con el motivo de tener un código más limpio se implementó la clase Utilerias, que es un compendio de métodos estaticos. Cada método representa el submenú de cada entidad y como parametro reciben las tablas hash necesarias para hacer las evaluaciones conrrespondientes, agregar elementos a las tablas, etc. según sea en caso de cada entidad. Por lo tanto en el switch case del main solo se observa la invocación a cada submenú y la estructura con todas

las opciones del submenú y las estructuras if-else anidadas donde se evaluan a las tablas hash se encuentran dentro de la clase Utilerias.

A continuación se describe más a dentalle los submenús contenidos dentro de la clase Utilerias.

Clase Utilerías

Los métodos estáticos de la clase Utilerías son los submenús para Alumno, Profesor, Asigantura y Grupo. En general, cada submenú tiene un estrututra similar y muestran las mismas opciones que son: Crear, Mostrar, Mostrar todos y Volver. Entonces, esas son las prosibles acciones por hacer después de elegir a alguna entidad. A continuación se explicará la estructura general de cada submenú usando el submenú Alumnos como ejemplo.

La estructura del submenú es un switch case, las posibles opciones son: Crear Alumno, Mostrar Alumno, Mostrar todos los Alumnos y Volver. Este método recibe a la tabla Hash que contiene a los alumnos y a la tabla hash de grupos.

- Crear Alumno: En esta opción se solicita al usuario ingresar el numero de cuenta del alumno que desea registrar. Inmediatamente se evalúa si dicho numero de cuenta ya existe dentro de la tabla hash alumno, de ser el caso se indica que el alumno ya está registrado y si no, procede a solicitar los demás datos (atributos básicos) de alumno. Se crea un objeto alumno de prueba y se evalua si se ingresaron datos válidos, recordando que cada setter tiene restricciones y que si no se cumplen el valor de los atributos será null, entonces mientras los atributos sean null, se volverá a solicitar que ingrese un dato válido. Finalmente, se agrega al objeto alumno a la tabla hash alumno, con su respectiva llave (numero de cuenta).

- Mostrar Alumno: Se solicita al usuario ingresar el numero de cuenta del alumno que desea ver, este numero de cuenta se busca dentro de la hash table alumno, si lo encuentra imprime la información básica del alumno y si el alumno está no inscrito en algun grupo (se revisa el arreglo de claves de grupo del objeto alumno) entonces se indicará que no está incrito a ningun grupo. Pero si, sí tiene claves dentro del arreglo, se buscaran estas claves dentro de la tabla Hash clases y se imprimirá la información sobre cada una de los grupos en los qe está inscrito el alumno.

- Mostrar Alumnos: Esta opción revisa si la tabla hash alumnos ya contiene objetos alumnos. Si el tamaño de la tabla hash alumnos es 0 entonces indica que aún no hay alumnos registrados, Si no,entonces recorreo la tabla hash es imprime los datos de cada alumno.

- Volver: Esta opción hace que la variable selector vuelva a ser 0 y por lo tante regresa al menú principal.

La estructura y validaciones mencionadas basicamente se repiten para los submenús de las entidades Profesor y Asignatura, considerando que cada submenú recibirá diferentes tablas hash dependiento de la entidad.

El submenú de la Grupos es diferente a los mencionados, pues en el grupo es donde se conjugan todos los objetos. Este submenú tiene las opciones: Crear Grupo, Mostrar Grupo, Añadir alumno a Grupo y Volver. A continuación se detalla cada una de estas opciones:

- Crear Grupo: Para crear un grupo se necesitan como mínimo un profesor y una asignatura, por lo tanto primero se evalúa si las tablas hash profesores y asignaturas, ya contienen objetos, de ser así se muestran los profesores y asignaturas registradas. Se solicita que se elija un profesor y una asignatura y se genera una clave de grupo comprendida por el nombre de la asignatura y una clave ingresada por el usuario. Posteriormente se evalua si esta clave ua existe dentro de la tabla hash grupos, si no existe dentro de la misma se crea un objeto grupo y al mismo tiempo se agrega dentro de la tabla hash grupos, asociandolo con su respectiva clave única. También se agrega la clave del grupo creado a la lista de claves de grupos del profesor y a la lista de claves de grupo de la asignatura.

- Mostrar Grupo: Se evalúa si la tabla hash grupos ya contiene grupos, si es así, entonces se muestra en pantalla la información básica de cada grupo. Después se solicita al usuario seleccionar uno de los grupos y se muestra los alumnos incritos a ese grupo en específico.

- Inscribir Alumno: Primero se evalúa si ya hay grupos disponibles de la tabla hash grupos. Posteriormente se despliega la información básica de los grupos disponibles y se solicita al usuario que elija uno. A continuación se solicita al usuario que ingrese el número de cuenta del alumno que se desea inscribir. Se verifica si el numero de cuenta está entre las llaves de las tabla hash alumnos, de ser el caso se revisa si el alumno asociado aún puede inscribirse a un grupo , puesto que tiene un límite de 3, por lo tanto se revisa el arreglo de claves de grupo del objeto alumno. Si el arreglo no está lleno entonces se agrega la clave al arreglo.

- Volver: Esta opción hace que la variable selector vuelva a ser 0 y por lo tante regresa al menú principal.

Esa es la estructura general del método main y de la clase utilerias que contiene a los submenús del menú principal. Como se puede observar el uso de colecciones fue muy recurrente, el manejo de claves o identificadores para cada objeto facilitó el manejo de las relaciones entre los objetos. Además dependiendo de la colección se pudieron usar diferentes métodos como add, getKeys, etc. que facilitaron el acceso a los elementos de las colecciones.

Conclusiones

- Martinez Olmos Osiris: Considero que las colecciones son una alternativa para guardar/agrupar varios datos, que es mucho más facil de manejar que los arreglos. No es necesario indicar el tamaño de la colección (son mutables), lo cual hace que sea muy sencillo seguir agregando elementos. Además cuentan con métodos que facilitan su uso como add, size, contains, etc. Para la elaborción de este proyecto tuvimos que analizar qué tipo de colección era la más adecuada según las características del grupo de datos que queriamos almacenar en ella, la forma en que accederíamos a los valores y si había elementos repetidos o no. La colección más utilizada en el proyecto y la que desempeñaba los papeles más importanes dentro del programa fueron los tablas Hash, esto porque la estructura del programa establecía las relaciones de los objetos mediante claves asociadas a cada objeto y dentro de las tablas Hash se almacenaban los objetos que se iban creando en la ejecución del sistema de inscripciones, con su respectiva clave. Pienso que se cumplieron los objetivos del proyecto, porque utlizamos listas conjuntos y tablas hash de forma correcta, y apoyandonos de la Figura 1 para poder elegir qué tipo era el más adecuado. También, se usaron algunos métodos de la documentación de estos objetos, cosa que optimizó mucho el desarollo del proyecto. Con respecto a lo que hace el programa, sí se logró que realizara lo requerido, además de que se incluyeron restricciones mediante setters y getters, y se respetó el paradigma de Programación Orientada a Objetos pues todos los métodos usados son métodos de instancia. Finalmente, la dinámica de trabajo a distancia fue un poco complicada al principio pues no estaba familiarizada con GitHub, pero tuve el apoyo de mis compañeros de equipo, quienes me enseñaron. Pienso que todos aportamos de forma importante al proyecto y primordialmente trabajamos en conjunto, ayudandonos entre todos y aportanto ideas.

- Rosales López Luis André: En este proyecto hicimos un uso y análisis muy extensivo de las colecciones en Java. El proyecto nos permitió identificar los mejores casos de uso para cada una de las colecciones utilizadas (Hashmap, Sets y Listas). Hicimos uso de las colecciones Hashmap con el fin de aprovechar las llaves de estas tablas para relacionar las diferentes entidades entre sí. Así mismo utilizamos los sets en las situaciones en donde no deseabamos tener elementos repetidos (por ejemplo, en la lista de alumnos dentro de cada grupo). Finalmente usamos las listas en aquellas situaciones en donde necesitabamos almacenamiento dinámico. Considero que los objetivos se cumplieron satisfactoriamente ya que ahora conozco de mejor manera las colecciones de Java, sus métodos, sus ventajas y desventajas. Asi mismo me resultó muy interesante el reto de realizar el diseño del programa de tal manera que fuese posible mostrar cada una de las colecciones generales de los elementos y además mostrarle al usuario las relaciones entre ellos. También fue muy gratificante el lograr crear un flujo de trabajo con mi equipo, aprovechando algunas herramientas como git para el manejo de versiones y overleaf para escribir este documento de manera colaborativa. Aún quedan algunos aspectos por mejorar en cuanto a la organización pero considero que hicimos un gran trabajo para ser nuestro primer trabajo en equipo de manera virtual.

- Núñez Quintana Luis Axel: Concluyo que se cumplieron los objetivos específicos y el objetivo general porque se llevó a cabo una investigación en donde se profundizó en los

métodos de cada colección mostrando el uso adecuado de cada una; se pudo crear un programa para la organización de alumnos, profesores, asignaturas y grupos; se generó la documentación del programa; se escribió un manual de usuario en donde se explica el uso correcto del sistema; y finalmente se logró crear un proyecto en equipo por medio de GitHub.

Considero que gracias a este proyecto pude conocer a fondo las coleciones que brinda el lenguaje de programación orientado a objetos Java y también tuve la oportunidad de comprender el uso correcto de conjutos, ya que anterior a este trabajo los implementabá a base de mapas. Esta implementación errónea no creaba problemas en el funcionamiento de los programas, sin embargo ahora con el correcto uso de colecciones será más eficiente en tiempo y óptimo en espacio.

Llego a la reflexión de que las dificultades que vivimos actualmente por la pandemia fueron un excelente pretexto para dominar el uso de herramientas para el trabajo en línea, también gracias al uso de ellas tuve la oportunidad de obtener una noción del trabajo en equipo en el campo laboral.

Referencias

1. Vindel Amor, R., 2015, Introducción a Colecciones en Java. Adictos al trabajo. Recuperado de: https://www.adictosaltrabajo.com/2015/09/25/introduccion-a-colecciones-en-java/. Fecha de consulta: 01/11/2020

2. Máster en Desarrollo de Aplicaciones Android, 2017. Las colecciones en Java. Recuperado de: http://www.androidcurso.com/index.php/tutoriales-java-esencial/462-las-colecciones-en-java. Fecha de Consulta: 01/11/2020

3. Lichtmaier, N., Collections de Java. Recuperado de: http://www.reloco.com.ar/prog/java/collections.html. Fecha de consulta: 02/11/2020

4. Joyanes, J. (2020). Programacion En C/C++ Java Y Uml (2.a ed.). MCGRAW HILL EDDUCATION.

5. $Java^{TM}$ Platform, Standard Edition 8 API. Recuperado de: https://docs.oracle.com/javase/8/docs/api/ Fecha de consulta 31/10/2020