**OBSERVACIONES IMPORTANTE ANTES DE INICIAR:**

Cada trabajo o actividad académica que sea desarrollado por usted como evidencia de su proceso de aprendizaje, debe estar 100% libre de frases copiadas desde cualquier fuente original (documento, libro, página web, proyecto de código fuente, video tutorial, etc) de información.

En ningún momento se le está pidiendo que haga las cosas de 0 y olvide por completo los conocimientos sobre los cuales se basa algún tema de esta asignatura, por el contrario, lo estamos animando a que lea, investigue, comprenda, deduzca, analice, sintetice, concluya, interprete, argumente, domine, practique, critique, niegue o afirme con bases sólidas basadas en la lectura y el estudio autónomo y autodidacta.

Tenga muy presente que mientras estudias esta profesión y durante toda tu vida, siempre seras evaluado y medido de acuerdo por un docente, por un instructor, por tu equipo de trabajo, por tus jefes, por la sociedad o por tus clientes.

Antes de realizar cualquier acción indebida en el desarrollo de esta y cualquier otra actividad académica, recuerda que actualmente existen muchas herramientas modernas y avanzadas que existen analizar, buscar, identificar, rastrear y comparar la idoneidad y originalidad de cada contenidos digital (texto, imágenes, archivos, etc), con especial profundidad en archivos de documentos académico o que contienen fuentes de programas informáticos, las cuales permiten detectar plagio de una manera tremendamente simple, rápida y automatizada.

Evita que el pensamiento del mínimo esfuerzo y el facilismo invada tu mente y tus acciones, ya que este destruirá tus sueños de crecimiento y éxito profesional.

**OBSERVACIONES**:

##### **1.** **ACTIVIDAD GRUPAL (GRUPOS DE 3 PERSONAS)**

1. CADA INTEGRANTE (SIN EXCEPCIÓN ALGUNA) DEBE SUBIR LA ACTIVIDAD DESDE SU RESPECTIVA CUENTA EN SIMA-PESAD
2. SOLO SE EVALUARÁ Y CALIFICARÁ A LOS ESTUDIANTES QUE HAYAN SUBIDO LA ACTIVIDAD, LOS ALUMNOS QUE NO ENTREGUEN LA ACTIVIDAD SERÁN EVALUADOS CON NOTA 1.0 AUNQUE APAREZCAN EN LA LISTA DE INTEGRANTES DEL TRABAJO.

FORMATO DE ENTREGA:

Un archivo comprimido en formato ZIP (NO USAR OTRO FORMATO)

○ El archivo comprimido debe contener como mínimo los siguientes archivos.

■ Documento en formato que contiene el desarrollo y explicación detallada del desarrollo de la actividad. En formato WORD (NO USAR OTRO FORMATO)

■ Adicionalmente, cada estudiante debe entregar un documento en word que contenga una explicación descriptiva y sustentación conceptual sobre el desarrollo de cada ítem.

■ Se debe adjuntar la carpeta con el proyecto de NetBeans con el código fuente de Java en el que se desarrollan los ejemplos de cada ítem.

■ El desarrollo y entrega del trabajo debe contener todos los ítem típicos y característicos de un trabajo académico.

1. Presentación
2. Tabla de contenido
3. Introducción
4. Objetivos
5. Justificación
6. Desarrollo
7. Síntesis y argumentación individual (es decir, una por cada integrante del grupo)
8. Bibliografía

**Normas APA para la elaboración de esta actividad**

<https://www.colconectada.com/normas-apa/>

**INFORMACIÓN SOBRE LA ACTIVIDAD ACADÉMICA**

Para esta actividad académica ustedes como estudiantes del programa de Ingeniería de Software en modalidad a distancia, de la Universidad de cartagena debe desarrollar, presentar y sustentar en tutoría la siguiente a:

Realizar una investigación documental sobre los siguientes temas:

1. Explique textual y gráficamente que es un una Lista y cuáles son las clases ofrece el SDK de Java para trabajar con listas

R//

**Lista**

Las listas en java son variables que permiten el almacenamiento de grandes cantidades de datos tal como los arreglos y matrices.



Java SDK nos ofrece las clases *LinkedList*, *Stack,* *Vector* y *ArrayList* para trabajar con listas.

1. Explique textual y gráficamente que es un una Cola, y cuales son las clases ofrece el SDK de Java para trabajar con Colas

R//

**Cola**

Las colas son las listas cuyos datos se introducen únicamente por un extremo “Final de la cola” y se extraen únicamente por el extremo contrario “Frente de cola”. El primer elemento en entrar a la cola siempre será el primero en salir.

Pantalla de video juego

Descripción generada automáticamente con confianza media

Java nos ofrece *Queue* y *PriorityQueue* para trabajar con colas, de esta manera nos permite llamar a todos sus métodos para modificar los datos de dicha cola.

1. Explique textual y gráficamente que es un una Pila, y cuales son las clases ofrece el SDK de Java para trabajar con Pilas

R//

Las pilas son listas cuyos datos se introducen y extraen por un único extremo, por tanto, el último elemento en entrar siempre será el primero en salir.

Diagrama

Descripción generada automáticamente con confianza media

Java nos ofrece la clase *Stack* para el manejo de pilas, de esta manera nos permite llamar a todos sus métodos para modificar los datos de dicha pila.

1. Explique por qué las listas, colas y pilas se consideran elementos fundamentales en las estructuras de datos

R//

**Listas, Pilas y Colas Como Pilares De La Estructuración De Datos**

Estos elementos presentes en la Estructura de Datos son de vital importancia dentro de la programación debido a que propician distintas formas de organización y manejo de datos que poseen cierta relación o cierto valor. De tal manera que permiten el procesamiento eficiente de grandes cantidades de información.

1. Explique textual y gráficamente el funcionamiento de una Lista simples enlazadas, hacer un ejemplo en Java

R//

**Listas simples enlazadas**

Las listas simples enlazadas consisten en un conjunto de nodos que guardan campos de datos y uno o dos enlaces pertenecientes al nodo anterior o siguiente.

El ultimo nodo no apunta a otro nodo.

Asimismo, la única manera de acceder a la lista es a través del primer nodo por ende no se puede acceder aleatoriamente, se debe recorrer toda la lista hasta llegar al valor deseado

Para modificar sus elementos, estas poseen operaciones básicas como add(), get(), remove(), set(), entre otros.

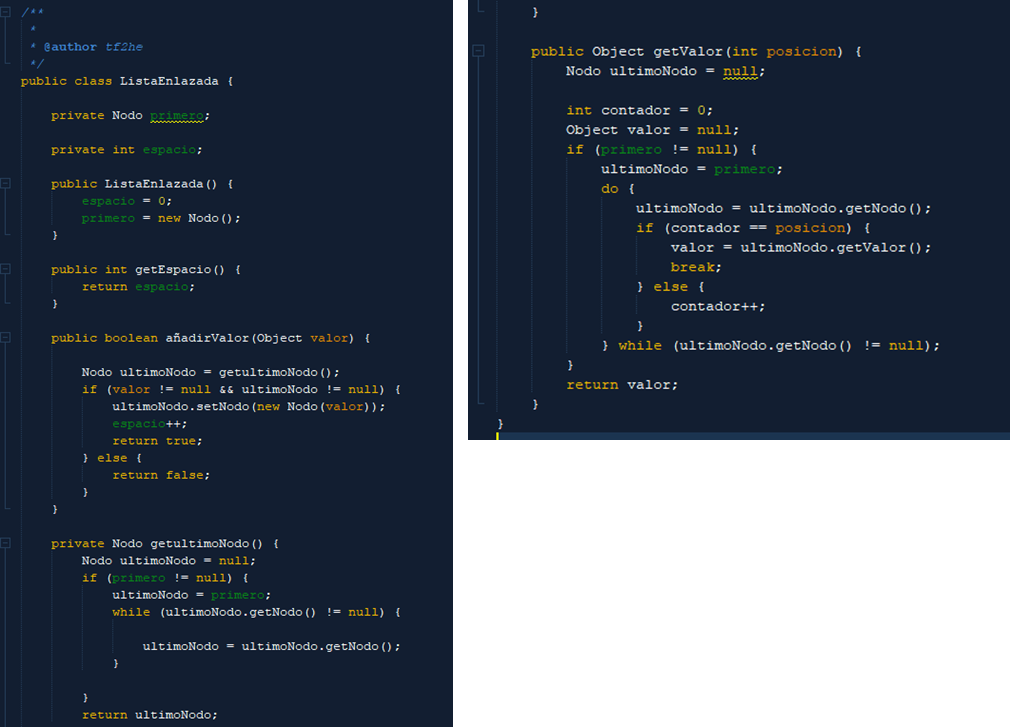
Un ejemplo básico de listas simples enlazadas en java es:

**Clase Nodo:**

Texto

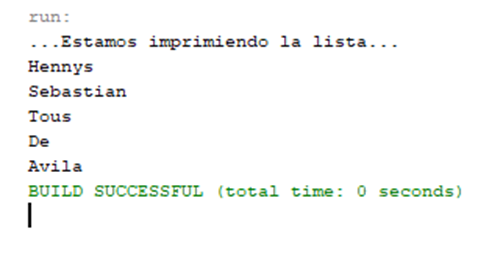
Descripción generada automáticamente

**Clase ListaEnlazada:**

****

**Clase Main y salida por consola:**

**Texto

Descripción generada automáticamente**

1. Explique textual y gráficamente el funcionamiento de una Lista doblemente enlazadas, hacer un ejemplo en Java

R//

**Listas Doblemente Enlazadas**

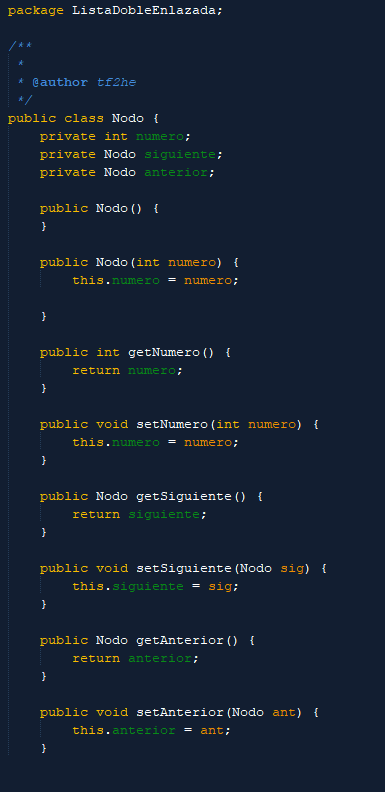
Este tipo de lista posee características similares a las listas simples enlazadas, sus diferencias radican en que las Listas doblemente enlazadas poseen dos enlaces por cada uno de sus nodos; Un enlace referencia al nodo anterior y el otro al posterior.

De esta manera, las listas doblemente enlazadas pueden ser recorridas en ambas direcciones.

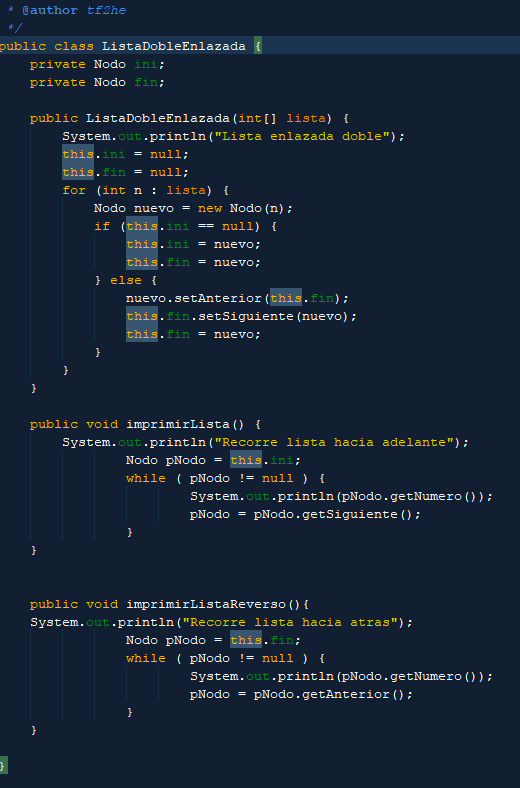


Un ejemplo básico de listas doblemente enlazadas en java es:

**Clase nodo:**

****

**Clase ListaDobleEnlazada:**

****

**Clase main:**

**Texto

Descripción generada automáticamente**

1. Explique textual y gráficamente el funcionamiento de una Lista enlazadas circulares

R//

**Lista Enlazada Circular**

Una lista circular es una estructura de datos donde el ultimo nodo apunta al primer nodo, por lo que es una lista interminable, cada nodo siempre tiene un nodo anterior y un nodo siguiente, su estructura es muy similar a una lista simple, por eso comparten características tanto en su implementación como en su gestión.

La lista no tiene fin porque cuando llega al último nodo, la lista comienza de nuevo.

Se accede a la lista a través del primer nodo o también conocido como la parte superior de la lista.

Si no se tiene cuidado al manipular listas circulares, se pueden crear bucles infinitos.

Diagrama

Descripción generada automáticamente

1. Explique textual y gráficamente el funcionamiento de una Lista enlazada simple circular, hacer un ejemplo en Java
2. Explique textual y gráficamente el funcionamiento de una Lista enlazada doblemente circular

R//

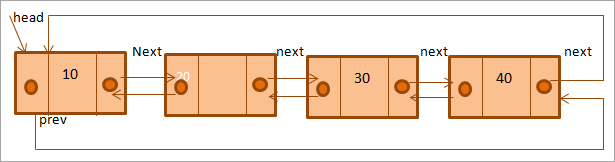
**Lista Circular Doble**

La lista doble circular es una especie de lista enlazada doblemente enlazada, pero tiene una función más para desplazarse por la lista, no tiene fin y tiene 2 punteros a sí misma.

Para hacer interminable la lista, el próximo puntero del último elemento apuntará al primer elemento y el puntero anterior del primer elemento apuntará al último elemento de la lista en lugar de apuntar a un valor NULL, como hemos visto en el caso de una lista simple o doblemente enlazada.

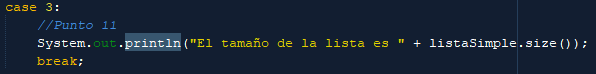
En una lista circular doble, nunca llegará a una posición en la que ya no sea posible desplazarse.

Cuando se llega al último elemento, el desplazamiento comenzará de nuevo desde el primer elemento.



1. Explique textual y gráficamente cual es el algoritmo o proceso para **contar** los elementos que tiene una lista. Hacer un ejemplo en Java.

R//



Debido a que implementamos la lista (List<ListaOBJ> listaSimple = new ArrayList<ListaOBJ>();) mediante la clase List y Arraylist, estas poseen un método que nos permite obtener el tamaño de la lista, el cual es .size().

1. Explique textual y gráficamente cual es el algoritmo o proceso para **saber si una lista está vacía o no**. Hacer un ejemplo en Java.

R//

Texto

Descripción generada automáticamente

Gracias a la implementación de la lista por medio de List y Arraylist, podemos llamar al método .isEmpty() el cual regresa true si la lista no contiene elementos y false si contiene elementos.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **buscar** un elemento en una lista. Hacer un ejemplo en Java.

R//

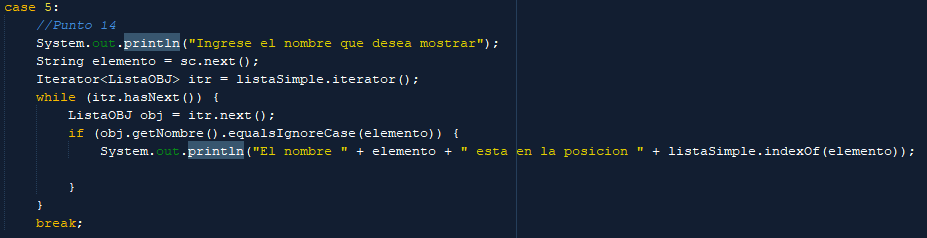
Texto

Descripción generada automáticamente

Para buscar un elemento en la lista, pediremos que nos den el indice de dicho elemento. Para esto declaramos que el objeto a buscar es igual al metodo .get(indice por teclado). Luego imprimimos dicho objeto.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **obtener la posición de un elemento** en una lista. Hacer un ejemplo en Java.

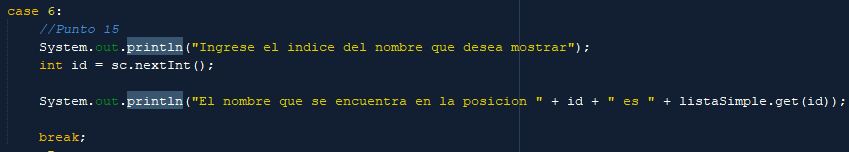
R//



Para este punto usamos un método llamado .indexOf() y un Iterador Iterator. Lo que haremos será recorrer la lista mediante el Iterator y cuando este encuentre un Nombre igual al ingresado por teclado, devolverá el índice usando el método indexOf.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **obtener un elemento en la posición X** de una lista. Hacer un ejemplo en Java.

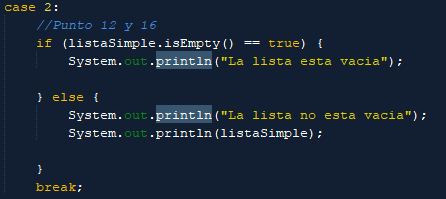
R//



Para este punto usamos un método llamado .get(id) el cual nos permite obtener el objeto de la lista que se encuentra en el índice id ingresado por teclado.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **saber si una lista está vacía o no**. Hacer un ejemplo en Java.

R//



Gracias a la implementación de la lista por medio de List y Arraylist, podemos llamar al método .isEmpty() el cual regresa true si la lista no contiene elementos y false si contiene elementos.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **agregar** un elemento al **inicio** de una lista. Hacer un ejemplo en Java.

R//

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

En este punto, usamos el método .add() en donde le indicamos que queremos agregar en el índice 0 (en el inicio) el nombre que se ingresara por teclado.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **agregar** un elemento al **final** de una lista. Hacer un ejemplo en Java.

R//

Texto

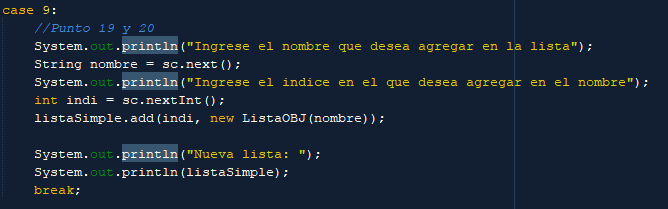
Descripción generada automáticamente

En este punto, usamos dos métodos:

* .size(), el cual es el tamaño de la lista. A este le restamos uno para dar el ultimo índice (ya que los índices empiezan de 0).
* .add() en donde le indicamos el índice usando la variable ultimoindi y el nombre que será ingresado por teclado.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **agregar** un elemento al en cualquier parte entre el i**nicio y el final** de una lista. Hacer un ejemplo en Java.

R//

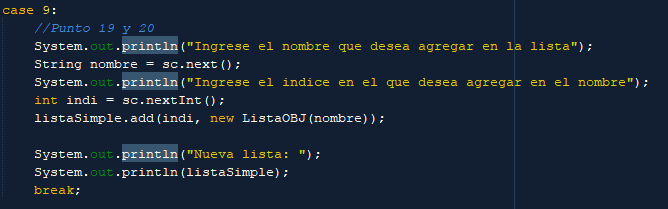


En este método pedimos por teclado el índice en el que queremos agregar el nombre y pedimos el nombre por teclado.

Luego agregamos el índice usando el método .add().

1. Explique textual y gráficamente cual es el algoritmo o proceso para **agregar** un elemento al **en cualquier parte de la lista.** Hacer un ejemplo en Java.

R//

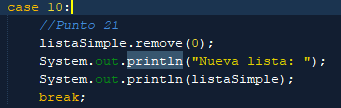


En este método pedimos por teclado el índice en el que queremos agregar el nombre y pedimos el nombre por teclado.

Luego agregamos el índice usando el método .add().

1. Explique textual y gráficamente cual es el algoritmo o proceso para **eliminar** un elemento al **inicio** de una lista. Hacer un ejemplo en Java.

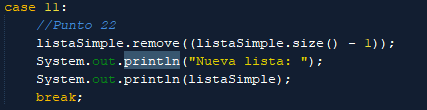
R//



En este punto usamos el metodo .remove() el cual elimina el objeto que se encuentra en un índice dado. En este caso le indicamos que queremos eliminar el elemento en el índice 0.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **eliminar** un elemento al **final** de una lista. Hacer un ejemplo en Java.

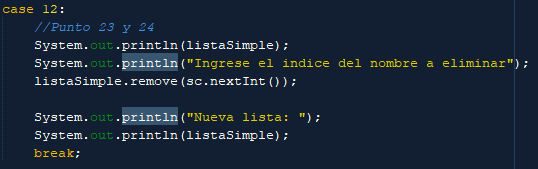
R//



En este punto usamos el método .remove() el cual elimina el objeto que se encuentra en un índice dado. En este caso le indicamos que queremos eliminar el elemento en el índice obtenido de la resta del tamaño de la lista menos 1.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **eliminar** un elemento al en cualquier parte entre el **inicio y el final** de una lista. Hacer un ejemplo en Java.

R//



En este método pedimos por teclado el índice en el que queremos eliminar el nombre.

Luego eliminamos el índice usando el método .remove().

1. Explique textual y gráficamente cual es el algoritmo o proceso para **eliminar** un elemento al **en cualquier parte de la lista.** Hacer un ejemplo en Java.

R//

Texto

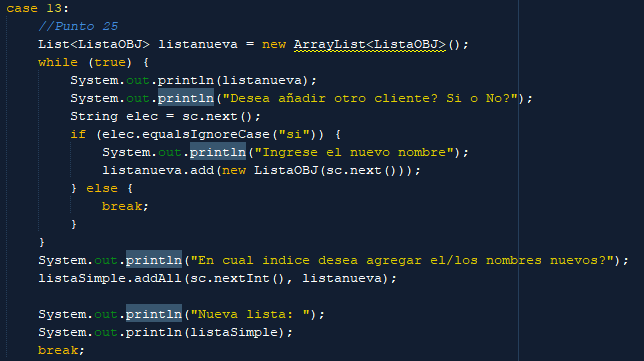
Descripción generada automáticamente

En este método pedimos por teclado el índice en el que queremos eliminar el nombre.

Luego eliminamos el índice usando el método .remove().

1. Explique textual y gráficamente cual es el algoritmo o proceso para **agregar** **varios** elementos al **en cualquier parte de la lista,** es decir, **agregar una sublista en otra lista.** Hacer un ejemplo en Java.

R//



En este punto creamos una lista nueva de tipo ListaOBJ en donde vamos a juntar los nuevos nombres. Luego preguntamos por teclado por medio de un bucle si quiere o no quiere agregar mas nombres a la lista nueva.

Una vez se hayan agregado los nombres a la nueva lista, usamos el metodo .addAll() y pedimos el índice en donde quieren ingresar los nuevos nombres en la lista original.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **eliminar** **varios** elementos al **en cualquier parte de la lista, e**s decir, **eliminar una sublista en otra lista.** Hacer un ejemplo en Java.

R//

Texto

Descripción generada automáticamente

Para este punto pedimos el índice inicial y el índice final que queremos eliminar.

Luego creamos una sublista usando el metodo .subList(), en donde indicamos que queremos hacer una sublista que inicia en el índice 1 y el índice 2.

Luego le pedimos que elimine los elementos de la sublista que se encuentren en la lista original, esto lo hacemos por medio del método .removeAll.

1. Explique textual y gráficamente cual es el algoritmo o proceso para **todos los elementos** de una Lista, es decir, vaciar una lista. Hacer un ejemplo en Java.

R//

Texto

Descripción generada automáticamente

Para este punto usamos el metodo .clear() el cual elimina todos los elementos presentes en la lista.

1. Explique textual y gráficamente al menos uno de los algoritmos o procesos utilizados para o**rdenar una lista de forma ascendente .**Hacer un ejemplo en Java.

R//

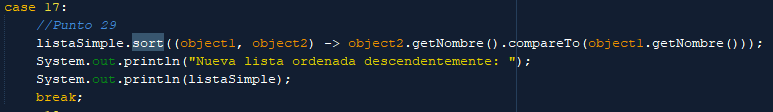
Texto, Sitio web

Descripción generada automáticamente con confianza media

Para este punto usamos el metodo .sort() el cual nos ordena la lista, pero, si queremos ordenarla de manera ascendente debemos crear una comparación en donde comparamos el nombre del primer objeto con el nombre del segundo objeto. De esta manera la lista es ordenada alfabéticamente de manera ascendente.

1. Explique textual y gráficamente al menos uno de los algoritmos o procesos utilizados para **ordenar una lista de forma descendente.** Hacer un ejemplo en Java.

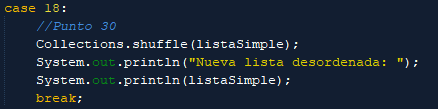
R//



Para este punto usamos el metodo .sort() el cual nos ordena la lista, pero, si queremos ordenarla de manera descendente debemos crear una comparación en donde comparamos el nombre del segundo objeto con el nombre del primer objeto. De esta manera la lista es ordenada alfabéticamente descendentemente.

1. Explique textual y gráficamente al menos uno de los algoritmos o procesos utilizados para ordenar una lista de forma aleatoria, es decir, **desordenar una lista.** Hacer un ejemplo en Java.

R//



Para este punto usamos la clase Collections de java para usar el método .shuffle(), el cual desordenara la lista deseada.

**BIBLIOGRAFIA**

<https://www.journaldev.com/11444/java-list#:~:text=Java%20List%20is%20an%20interface,search%20elements%20in%20the%20list>.

<https://sites.google.com/a/espe.edu.ec/programacion-ii/home/listas-enlazadas>

<https://www.arquitecturajava.com/utilizando-java-8-predicate/>

<https://www.geeksforgeeks.org/arraylist-sublist-method-in-java-with-examples/>

<https://www.geeksforgeeks.org/collections-sort-java-examples/>

<https://devs4j.com/2017/11/23/listas-doblemente-ligadas-en-java/>

<https://www.javatpoint.com/java-list>

<https://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/Cola.svg/440px-Cola.svg.png>

<https://blog.martincruz.me/2012/10/pilas-en-c.html>

<http://codigolibre.weebly.com/blog/listas-simples-en-java>

<https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/ldoble.html>

<http://codigolibre.weebly.com/blog/listas-circulares-simples-en-java>

<http://coodigob.blogspot.com/2013/03/listas-enlazadas-circulares-dobles.html>

<https://spa.myservername.com/doubly-linked-list-java-implementation-code-examples>