ESCUELA DE INGENIERÍA INFORMÁTICA

GUÍA DE EJERCICIOS

TECNOLOGÍA APLICADA

UIA

Guía práctica del curso
“Estructuras de datos y algoritmos”: Listas

Elaborada por:

Olda Bustillos Ortega

Daniel Álvarez Garro

Universidad Internacional de las Américas
Escuela de Ingeniería Informática

San José, Costa Rica

Máster Olda Bustillos Ortega, Directora de la Escuela de Ingeniería Informática de la UIA.

Sinopsis de la autora

Profesional en tecnologías de información con formación académica combinada a una experiencia práctica de más de 25 años en gestión de procesos empresariales, gerencia de proyectos, auditoría de sistemas de información, consultora en procesos de implementación tecnológica, desarrollo de *software* y análisis de procesos organizacionales para la innovación de T.I.

- Bachiller en Ingeniería de Sistemas.
- Licenciada en Administración de Empresas con énfasis en Sistemas de Información.
- Licenciada en Administración de Empresas con énfasis en Gerencia.
- Máster en Auditoría de Tecnología Informática.
- Máster en Administración de Proyectos Informáticos.
- Estudiante del Programa de Doctorado en Ingeniería Informática con énfasis en Inteligencia Artificial (actualmente).
- Certificada en Administración Estratégica de *Harvard Extension School*.
- Certificada en Innovación de enseñanza y aprendizaje Costa Rica STEM 2.0, *Laspau affiliated with Harvard University*.
- Docente universitaria con más de 25 años de experiencia.
- Líder en el proceso de acreditación al Sistema Nacional de Acreditación de Enseñanza Superior (SINAES) de las carreras de la Escuela de Ingeniería Informática de la UIA.
- Miembro del *Institute of Electrical and Electronics Engineers* (IEEE) IEEE-*Computer Society*.

Licenciado Daniel Álvarez Garro

Sinopsis del autor

Licenciado en Ingeniería Informática con énfasis en Calidad del *Software*, con más de 15 años de experiencia en aspectos relacionados al Desarrollo de Software para entidades financieras. Desarrollador *senior* de experiencia de usuario y docente de la Universidad Internacional de las Américas.

EDITORIAL

La presente guía práctica para la Escuela de Ingeniería Informática surge en el año 2021, como respuesta a la necesidad de contar con material formal, de utilidad práctica y con ejemplos de casos reales. Estos contenidos de los diferentes cursos impartidos en las carreras de Ingeniería en Informática, Ingeniería en Sistemas de Información e Ingeniería de *Software* son adecuados a sus contenidos.

El objetivo de estas guías es orientar al estudiante a establecer una solución a los problemas que se plantean utilizando un análisis de la situación o problema, un planteamiento de un diseño, un desarrollo y realizando las pruebas necesarias para garantizar los resultados obtenidos.

Las guías están clasificadas en cinco áreas de conocimiento: “Tecnología aplicada”, “Asuntos organizacionales”, “Infraestructura”, “Metodologías y tecnologías de *software*” e “Interdisciplinarios”.

En la sección “Tecnología aplicada” se incluyen las tendencias tecnológicas actuales y su aplicación en la solución de un problema o servicio. Ejemplos de estas tecnologías son las bases de datos, los *frameworks* para desarrollo, lenguajes de programación, arquitecturas, entre otros. Adicionalmente, en esta área se pretende desarrollar fundamentos de esa tecnología: los conceptos y habilidades básicas para comprender los principales conceptos de la computación e informática, como por ejemplo los conceptos de autómatas, sistemas numéricos, algoritmos y diagramas de flujo, entre otros.

En “Asuntos organizacionales” se incluyen aspectos administrativos, como por ejemplo la administración de los recursos para un proyecto informático, los tiempos de duración, los costos, qué tipo de sistemas empresariales son los adecuados para la organización, la planificación de estrategias para el uso de tecnologías y velar por los resultados que los procesos generen.

El área “Infraestructura” contiene todo lo relacionado a la arquitectura y organización de los computadores, los sistemas operativos, las redes de telecomunicaciones. Asimismo, incluye elementos de seguridad informática e infraestructura virtual.

En “Metodologías y tecnologías de *software*” se abarca lo concerniente al análisis y diseño de las soluciones tecnológicas. Para esto se utilizan elementos de especificación del *software*, procesos de ingeniería, aspectos relacionados con la calidad de los sistemas, la verificación y validación de los resultados, y la implementación y el mantenimiento de las aplicaciones.

La sección “Interdisciplinarios” aborda aspectos que no son necesariamente técnicos, pero que ayudan a la formación integral del estudiante y a construir habilidades necesarias del profesional en tecnologías de información, como lo son otro idioma, elementos matemáticos, de comunicación, de metodologías de investigación, de presupuestos y estadísticas. El dominio del idioma inglés es importante para la interacción con personal de países que utilizan esa lengua universal. Las habilidades de comunicación para entrevistar a los usuarios, elaborar y exponer reportes técnicos de los proyectos resulta muy importante.

TABLA DE CONTENIDO

TABLA DE FIGURAS

CURSO: ESTRUCTURAS DE DATOS Y ALGORITMOS

a) Generalidades

La presente guía, titulada *Guía práctica para el desarrollo de ejercicios del curso "Estructuras de datos y algoritmos"*, forma parte de la serie de guías correspondientes al área de Tecnología aplicada. El curso es impartido para las carreras de Ingeniería en Informática, Ingeniería en Sistemas de Información e Ingeniería de *Software*.

b) Competencias por desarrollar

Este cuaderno de prácticas tiene el propósito de desarrollar en el estudiante las siguientes competencias:

1) Utilizar de forma eficiente los tipos de datos y estructuras de datos más adecuados para la resolución de problemas.
2) Tener facilidad para realizar procesos de abstracción, análisis y síntesis para la implementación correcta de algoritmos.
3) Diseñar algoritmos que tengan un coste temporal y espacial que se ajusten de manera correcta a la cantidad de datos por procesar.

c) Contenidos temáticos

Este cuaderno de prácticas tratará sobre el tipo abstracto de dato Lista, sus diferentes tipos y generalidades, además de una serie de ejercicios para el estudiante.

d) Resultados de aprendizaje

1. Diferenciar las estructuras de datos.
2. Diferenciar los tipos de listas.
3. Desarrollar operaciones con listas enlazadas (creación, borrado, inserción, listado de nodos).
4. Programar la inserción y la eliminación de datos con los comportamientos específicos de listas.

ESTRUCTURAS DE DATOS

Existen estructuras de datos dinámicas y estáticas, lineales y no lineales. En esta guía se estará abordando la estructura Lista Enlazada; no obstante, existen otros tipos de estructuras, como se podrá ver en la figura 1.

Las estructuras de datos estáticas corresponden a los *Arrays*, conocidos también como arreglos unidimensionales o vectores, y las matrices o tablas, conocidas también como arreglos bidimendionales. La diferencia entre las estructuras de datos estáticas y las estructuras de datos dinámicas es que las estructuras estáticas establecen su tamaño en memoria durante la compilación y permanecen inalterables durante la ejecución del programa, mientras que las estructuras dinámicas crecen y se contraen a medida que se ejecuta el programa (Aguilar, 2000). Es importante que quede clara esta diferencia, ya que con la lista no hay que preocuparse por indicar un tamaño para almacenar los elementos que se tienen, lo que da mucha ventaja a la hora de resolver distintos problemas de la vida real en los que no se tiene una idea de cuántos elementos se van a tener en un principio.

A continuación, se ven los distintos tipos de estructuras de datos que existen. La primera figura que se observa, con el nombre de árbol, es una estructura no lineal y dinámica, al igual que la última figura, con el nombre de grafo, esto significa que no se sigue una secuencia lógica como en el resto de estructuras, las cuales siguen un patrón de izquierda a derecha o de arriba hacia abajo. Las figuras con nombre: pila, cola y lista son estructuras lineales dinámicas, mientras que el arreglo es una estructura lineal pero estática (se necesita especificar el tamaño que va tener esta estructura); finalmente, se tiene la figura con el nombre de matriz, la cual es una estructura estática y lineal, pero que a diferencia del arreglo es bidimensional.

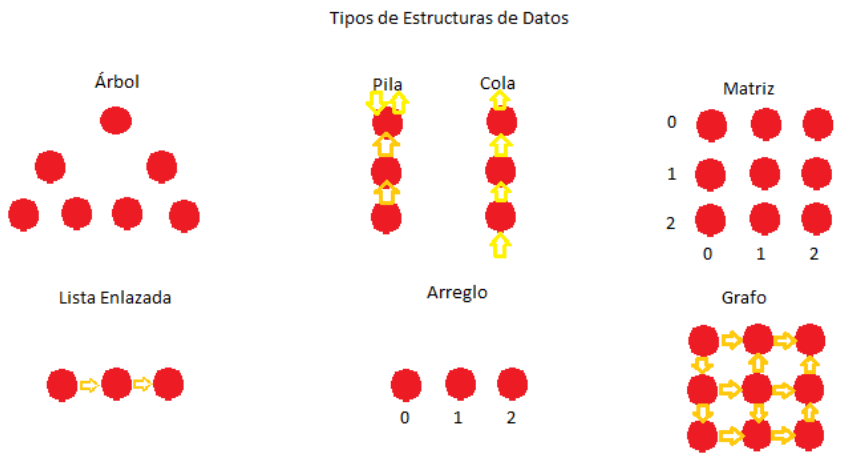


Figura No 1. Tipos de estructuras de datos lineales y no lineales, estáticos y dinámicos
Fuente: Elaboración propia, 2021.

Para el caso de las listas enlazadas, es posible trabajar con varias modalidades, entre las que se pueden mencionar las listas enlazadas simples, listas doblemente enlazadas o las listas circulares (con o sin doble enlace), cada una con una funcionalidad diferente a otra según el problema que se esté resolviendo. Dentro de las listas simplemente enlazadas tenemos dos casos en particular, las pilas y las colas. De acuerdo con Sznajdleder (2017), las pilas son estructuras lineales, pero deben respetar la restricción de que el último elemento que entra debe ser el primero en salir; por otra parte, las colas presentan la restricción de que el primer elemento en entrar debe ser el primer elemento en salir (pp. 144-145).

Esta guía práctica desarrolla la estructura de datos "lista enlazada" (ligada o encadenada), la cual suele encontrarse como "Linked-List" o "Arrays Dinámicos" a la hora de utilizar las librerías que ya contienen cada lenguaje de programación. Aguilar (2000) menciona que una lista enlazada es: "una colección de elementos denominados 'nodos' dispuestos uno a continuación de otro, cada uno de ellos conectado al siguiente elemento por un 'enlace' o 'puntero'" (p. 369).

Las listas enlazadas son una estructura lineal muy similar a los vectores, pero con la distinción de que nunca hay que especificar el tamaño que la estructura va a tener, por lo que podrá almacenar una cantidad infinita de elementos.

Como se puede observar en la figura 2, se tiene un nodo el cual contiene un elemento con valor “MSP”, el cual corresponde, por ejemplo, al código de un aeropuerto, y la flecha es el puntero o referencia al siguiente nodo en la lista, el cual también sería otro código de otro aeropuerto.

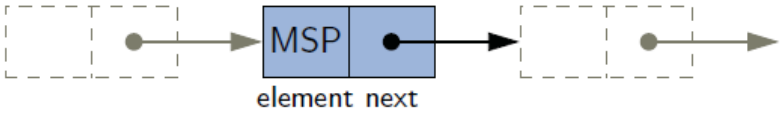


Figura No. 2. Ejemplo de una instancia de un nodo que forma parte de una lista enlazada simple.
Fuente: GoodRich, Tamassia y GoldWasser, 2014.

A continuación, se puede obervar la programación en lenguaje JAVA de la clase “nodo”, la cual es requisito fundamental para la creación de una “lista enlazada”. Cabe aclarar que este nodo es para una lista simplemente enlazada. De acuerdo con la modalidad que se esté trabajando: lista simple, lista doblemente enlazada, lista circular o lista circular doblemente enlazada, la clase nodo puede variar.

```
public class Nodo<T> {

    public T data;
    public Nodo siguiente;

}
```

Figura No. 3. Programación en Lenguaje Java
Fuente: Elaboración propia, 2021.

También, es importante notar en la clase Nodo de la figura 3, que estamos usando el concepto de clase genérica en Java, lo que, de acuerdo con Oracle(2021), permite que la clase pueda ser instanciada usando diferentes tipos de datos sin cambiar la programación, lo cual proporciona robustez al código fuente.

Es importante tener en cuenta que siempre que se ocupe manipular una lista, se necesita tener referenciado el primer nodo en la lista, el cual se llamará “principio” o “inicio”, a diferencia de un vector, donde simplemente se llama cualquier elemento en función de su

ubicación, que va de 0 a N-1. GoodRich *et al.* (2014) mencionan que, en caso de no hacer referencia explícita a la cabeza de la lista, no se podría localizar el primer nodo o ninguno de la lista enlazada.

Como se observa en la figura 4, se tiene una lista enlazada de códigos de aeropuerto, en la cual, para poder recorrer la lista, se necesita tener referenciada la cabeza (*head*) o principio de la lista, la cual corresponde al nodo "LAX", y el último nodo tiene una referencia a *null* que marca el final de la lista, en este caso "BOS":

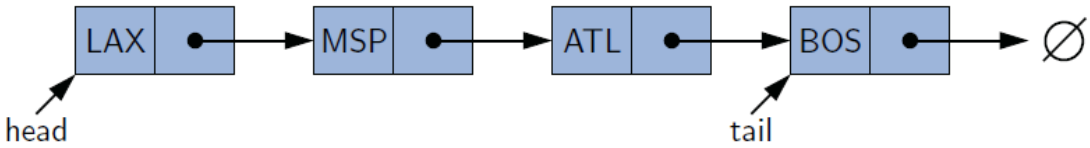


Figura No. 4. Ejemplo de lista enlazada simple especificando el principio y final de la lista.
Fuente: GoodRich *et al.*, 2014.

En la figura 5 se puede observar la programación en lenguaje JAVA de la lista enlazada simple, la cual indica el principio de la lista:

```
public class ListasSimples<T, K> implements Listas<T, K> {

    private Nodo<T, K> Head;

    ListasSimples() {
        Head = null;
    }
```

Figura No. 5. Programación en Lenguaje Java
Fuente: Elaboración propia, 2021.

La estructura de datos lista se puede dividir en cuatro categorías, las cuales son:

1) Listas enlazadadas simples: cada nodo (elemento) contiene un único enlace que conecta ese al nodo siguiente o sucesor. La lista es eficiente en recorridos directos ("adelante").

2) Listas doblemente enlazadas: como se muestra en la figura 6, cada nodo contiene dos enlaces, uno a su predecesor y el otro a su sucesor. La lista es eficiente tanto en recorrido directo (“adelante”) como en recorrido inverso (“atrás”).
3) Lista circular enlazada simple: como se muestra en la figura 7, una lista enlazada simple con el detalle de que su último elemento (cola) se enlaza al primer elemento (cabeza), de tal modo que la lista puede ser recorrida de modo circular o en “anillo”.
4) Lista circular doblemente enlazada: como se muestra en la figura 8, este tipo particular de lista es el más completo y funcional de todos. Es una lista doblemente enlazada en la que el último elemento se enlaza al primer elemento y viceversa. Esta lista se puede recorrer de modo circular (en anillo) tanto en dirección directa (adelante) como inversa (atrás).

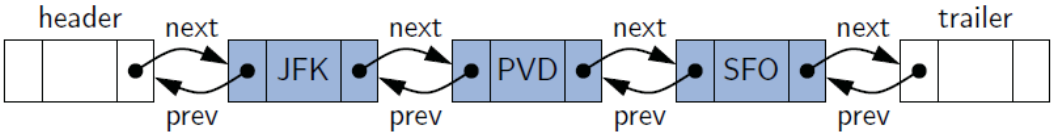


Figura No. 6. Ejemplo de una lista doblemente enlazada.
Fuente: GoodRich *et al.*, 2014.

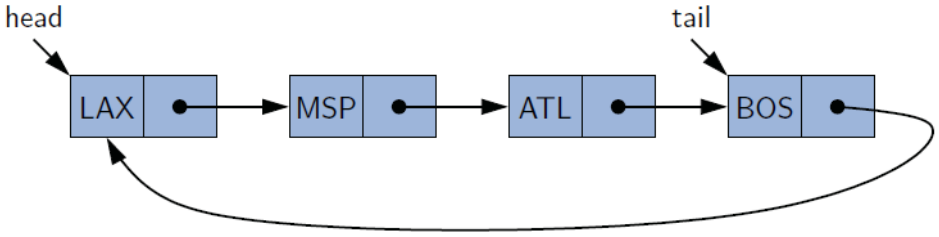


Figura No. 7. Ejemplo de una lista circular enlazada simple.
Fuente: GoodRich *et al.*, 2014.

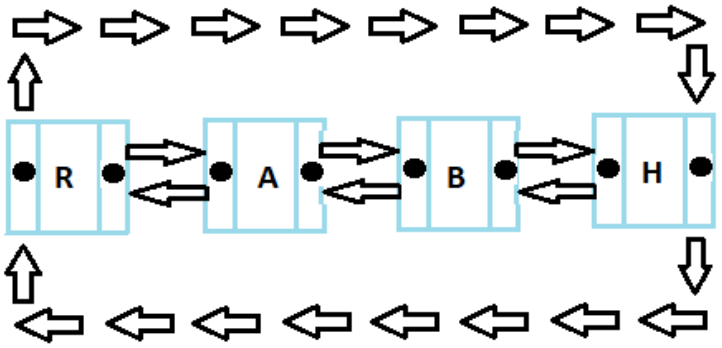


Figura No. 8. Ejemplo de una lista circular doblemente enlazada.
Fuente: Elaboración propia, 2021.

En las listas simplemente enlazadas tenemos dos tipos especiales que son muy utilizadas en la programación y solución de problemas, las Pilas y las Colas.

Los métodos primitivos que tiene una estructura lista, independientemente de sus cuatro modalidades, incluyen: “insertar” (al principio, al medio, al final o de manera ordenada), “borrar” (al principio, al medio, al final o por un valor), “vacía” (devuelve *true* o *false* según esté vacía o no), “reiniciar” (remueve todo el contenido de la lista), “existe” (devuelve *true* o *false* si un elemento existe o no en la lista) y “tamaño” (devuelve la cantidad de nodos que tiene la lista).

Si se emplea JAVA como lenguaje para implementar la lista, se tiene que, al ser un tipo de datos abstracto, suele implementar una interfase en la cual se indique el comportamiento que va a tener cualquiera de las cuatro modalidades de lista enlazada que se necesite, y de esta manera cada estructura implementará el código necesario para funcionar de manera correcta. El uso de la interfase se ejemplifica en la figura 9, y el polimorfismo se ejemplifica en las figuras 10 y 11.

```java
public interface Listas<T, K> {

    public void imprimir();

    public T buscar(T data);

    public boolean borrar_inicio();
    public boolean borrar_medio(T data);
    public boolean borrar_final();

    public boolean insertar_inicio(T nombre, K edad);
    public boolean insertar_medio(T nombre, K edad, T prev);
    public boolean insertar_final(T nombre, K edad);
```

Figura No. 9. Ejemplo de interfase de la estructura de datos "lista enlazada", la cual puede ser implementada por cualquiera de las cuatro modalidades.
Fuente: Elaboración propia, 2021.

```java
public boolean insertar_inicio(T nombre, K edad) {
    Nodo<T, K> nuevo = new Nodo<T, K>();
    nuevo.nombre = nombre;
    nuevo.edad = edad;
    nuevo.siguiente = null;
    try {
        if (Head != null){
            nuevo.siguiente = Head;
            Head = nuevo;
        }
        else
            Head = nuevo;
        return true;
    }
    catch (Exception ex) {
        return false;
    }
}
```

Figura No. 10. Implementación del método insertar en una lista enlazada simple.
Fuente: Elaboración propia, 2021.

```
public boolean insertar_inicio(T data){
    NodoDoble<T> nuevo = new NodoDoble<T>();
    nuevo.data = data;
    nuevo.siguiente = nuevo.anterior = null;

    try {
        if (Head != null){
            nuevo.siguiente = Head;
            Head.anterior = nuevo;
            Head = nuevo;
        }
        else
            Head = nuevo;
        return true;
    }
    catch (Exception ex) {
        return false;
    }
}
```

Figura No. 11. Implementación del método insertar en una lista circular doblemente enlazada.
Fuente: Elaboración propia, 2021.

Como se puede observar en las figuras 10 y 11, el método “insertar_inicio” ha sido sobreescrito en las clases que implementan la interfase “Lista”, y cada una de las listas implementa el método a su manera, según corresponda.

LISTA DE EJERCICIOS PARA SOLUCIONAR

MANEJO DE RESTAURANTES

El restaurante *Le petit monde* tiene un serio problema, porque la anfitriona del restaurante nunca sabe quién llegó primero y quién después, para ordenar la fila de espera por una mesa en el restaurante, lo que ocasiona que muchos comensales se enojen y se retiren del restaurante.

Por esta razón, el chef ejecutivo ha contratado a la empresa a la cual usted pertenece para el desarrollo de la aplicación que solvente la situación y ha definido los siguientes requerimientos:

1. La aplicación debe administrar los comensales que se encuentren en espera por una mesa.
2. El primer comensal que esté esperando por una mesa debe ser el primero en ser acomodado, apenas se libere una mesa.
3. Los clientes deben respetar la fila y la espera, por lo tanto, el último cliente que llegue debe ser acomodado al final de la fila.
4. La única excepción a la regla es por cumplimiento de la ley 7600. Los clientes de tercera edad, mujeres embarazadas y los que presenten alguna discapacidad deben ser siempre los primeros en la fila, independientemente de quiénes estén esperando o cuánto hayan esperado.
5. Una vez que haya una mesa libre, la anfitriona debe revisar en la aplicación quién es el siguiente cliente en espera, y una vez que el comensal se encuentre en una mesa, quitarlo de la fila.
6. Además de las funcionalidades anteriores, el chef y la anfitriona deben poder consultar cuántos comensales hay en espera en un momento determinado.

Para cumplir con las especificaciones anteriores, se formó un equipo de desarrollo, liderado por un líder técnico, y a usted se le ha asignado el desarrollo de las siguientes clases en Java:

i. Clase Nodo
    a. Debe contener los siguientes atributos:
        i. Nombre
        ii. Cantidad de comensales
        iii. ¿Es cliente preferencial? (Sí o No)
        iv. Coordenada de mesa asignado (dos números separados por “;”)
        v. ¿Requiere silla para bebés? (Sí o No)

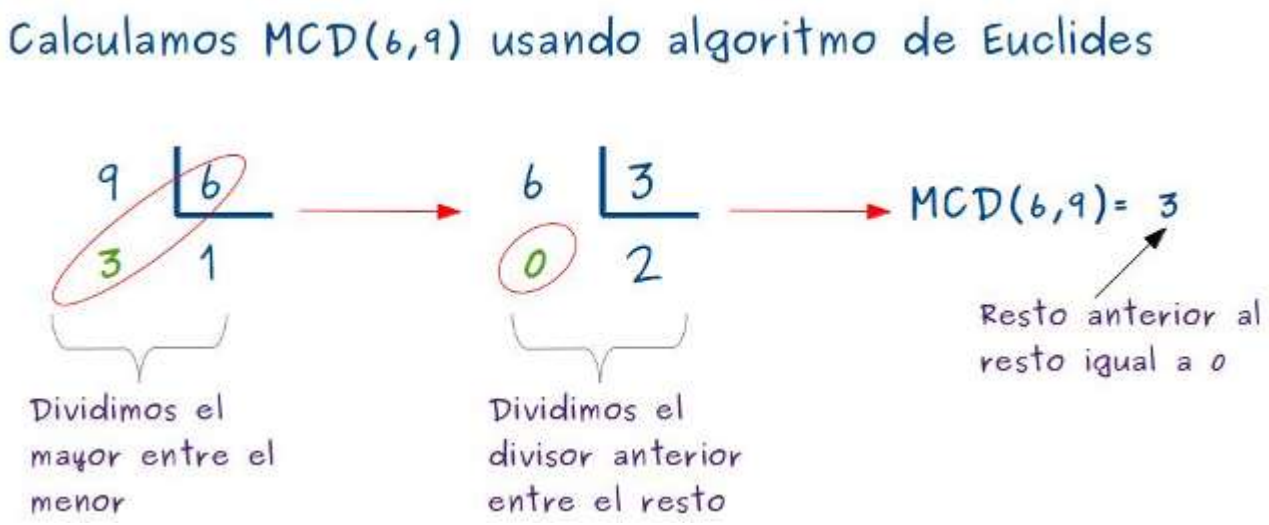
ii. Clase Fila
    a. Debe tener una instancia “private” de la clase Nodo denominada FILA.
    b. Debe tener métodos para las siguientes funcionalidades:
        i. Agregar un nuevo comensal al final de la fila.
        ii. Agregar un cliente preferencial al inicio de la fila.
        iii. Determinar si el cliente trae un niño que requiera una silla para bebés.
        iv. Borrar un cliente cualquiera (al principio, al final o al medio), por si algún cliente no quiere esperar y decide irse.
        v. Borrar un cliente una vez que ya se haya sentado, para esto se ha de eliminar el siguiente en la fila.
        vi. Mostrar la cantidad de clientes en la fila.
        vii. Mostrar el nombre del cliente que sigue en la fila.
        viii. En el restaurante solamente hay 20 mesas disponibles. Para asignar la mesa a un cliente, se debe mantener una matriz de 5 filas por 4 columnas, que almacene el nombre del cliente asignado. Las coordenadas (fila, columna) en donde se guarde el nombre del cliente son las coordenadas que se almacenan como un dato en la clase Nodo.
        ix. Se debe tener una validación que la fila de personas no supere las 100 personas en espera por una mesa, y también que no se sobrepase de las 20 mesas que se indican en el punto anterior.

Al desarrollar las clases anteriores, debe tomar en cuenta que esto es parte de un equipo de desarrollo. Este equipo decidió utilizar estructuras dinámicas de memoria, por lo tanto, debe incluir el manejo dinámico en las clases y realizar las validaciones necesarias para evitar el ingreso incorrecto de datos por parte del usuario.

MÁXIMO COMÚN DIVISOR

Debe elaborar un programa en Java con las siguientes consideraciones:

1. Debe implementar una lista simplemente enlazada que apoye a estudiantes de primaria en matemáticas, la cual debe almacenar dos números enteros y el máximo común divisor de estos.
2. La lista debe admitir las siguientes operaciones
    a. Ingreso de dos números (enteros positivos).
    b. Borrado de los dos números y su máximo común divisor.
3. Utilice dos botones de opción “Ingresar” y “Borrar”.
4. Solo debe usar un botón de comando denominado “Aceptar”. Cuando este botón es presionado, activa el evento clic, que debe validar cuál de las acciones seleccionó el usuario en los botones de opción. Esto debe ser manejado con una clase que implemente la interfaz del evento del botón.
5. Cuando el usuario seleccione que desea ingresar un par de números, este debe ser el último nodo que se ingrese en la lista.
6. Una vez que un nodo haya sido ingresado, se debe calcular cuál es el máximo común divisor de los dos números, el cual es el mayor número entero que los divide sin dejar resto, tal y como lo muestra la siguiente imagen:



7. La obtención del máximo común divisor se debe realizar con un algoritmo recursivo.
8. Cuando el usuario decida borrar un par de números, se debe buscar estos números en la lista, y si existen, eliminarlos; por el contrario, indicarle al usuario que ese par de números no están almacenados.
9. El ingreso y borrado de los números se realiza a través de los mismos cuadros de texto.
10. La disposición de los objetos gráficos queda a decisión del estudiante.

MANEJO DE PILAS

Realice la programación necesaria en una clase de Java, para solventar el siguiente enunciado:

"Escribir un programa que, haciendo uso de una Pila, procese cada uno de los caracteres de una expresión que viene dada en una línea. La finalidad es verificar el equilibrio de paréntesis, llaves y corchetes. Por ejemplo, la siguiente expresión tiene un número de paréntesis equilibrado: ((a + b) * 5) – 7. A esta otra expresión le falta un corchete: 2 * [(a + b) / 2.5 + x – 7 * y"

En el desarrollo y la solución del enunciado anterior se calificará:

i. Programación del TAD correspondiente a una pila.
ii. La operación Incluir un elemento a la pila.
iii. La operación Eliminar un elemento de la pila.
iv. Verificación del equilibrio de la expresión mediante la estructura pila.

MANEJO DE DIFERENTES TIPOS DE LISTAS

A continuación, se le presenta una serie de problemas para los cuales debe realizar la programación necesaria en Java con tal de solventar cada uno de ellos:

1. Escribir un programa para obtener una lista doblemente enlazada con los caracteres de una cadena leída desde el teclado. Cada nodo de la lista tendrá un carácter.

2. Escribir un programa para trasladar los valores almacenados en una pila a otra. Se deben realizar los traslados de los datos respetando los movimientos definidos para una estructura LIFO. Se deben imprimir en pantalla la pila original y luego la pila con los valores trasladados.

3. Escribir un programa en el que se generen 100 números aleatorios en el rango de-25 a +25 y se guarden en una cola, implementando las operaciones y respetando la estructura FIFO. Una vez completada la inclusión de los números, el usuario puede solicitar que se cree otra cola, pero solo con los números negativos de la primera cola.

MANEJO DE BIBLIOTECAS

Usted ha sido contratado por el Sistema Nacional de Bibliotecas para elaborar un programa que apoye la labor de la restauradora de libros de la institución.

La restauradora trabaja todos los días, seleccionando el primer libro que haya llegado a la oficina, lo restaura y lo envía a una sucursal. Luego continúa con el siguiente libro, y así hasta llegar al último libro que haya llegado ese día.

Por cada libro que la restauradora trabaja, ella necesita ingresar en el programa la siguiente información:

- Nombre del libro
- Nombre del autor
- Año de publicación
- Identificador
- Editorial

Por lo tanto, se le solicita a usted como programador realizar el siguiente desarrollo:

1. Desarrollar un TAD para una lista simplemente enlazada que almacene la información que necesita la restauradora.
2. Desarrollar un método que incluya los libros que se reciben durante el día y que deben ser incluidos al final de los pendientes de la restauradora.
3. Desarrollar un método que permita eliminar los libros que ya han sido restaurados y que deben ser trasladados a las sucursales.
4. Desarrollar un método recursivo que permita obtener el identificador de cada libro restaurado.

¿Cómo generar el identificador de un libro?

La restauradora de libros ha generado su propia manera de elaborar un identificador que le permita saber cuáles libros ha restaurado o no.

Para esto, utiliza la siguiente función:

$$\rho(x) = \begin{cases} 1, & x = 0 \\ x \cdot \rho(x - 1), & x > 0 \end{cases}$$

La “x” de la función es el año de publicación del libro. Si la restauradora lo desconoce, le asigna un 0.

Por lo tanto, por cada libro almacenado, la restauradora solamente ingresa el año de publicación y el programa automáticamente genera el identificador, a partir de la función que detalló la restauradora.

REFERENCIAS

Aguilar, L. J. (2000). *Programación en C++ Algoritmos, Estructuras de Datos y Objetos.* Madrid: McGrawHill.

Deitel, H. M. y Deitel, P. J. (2008). *C++ Cómo Programar*. (6ª ed.). México: Pearson Education.

GoodRich, M. T., Tamassia, R. y GoldWasser, M. (2014). *Data Structures & Algorithms.* Addison Wesley.

Oracle. (23 de octubre de 2021). Java Documentation. *Oracle*: https://docs.oracle.com/javase/tutorial/java/generics/why.html

Sznajdleder, P. A. (2017). *Programación Orientada a Objetos y Estructura de datos a Fondo. Implementación de Algoritmos en Java.* Ciudad Autónoma de Buenos Aires: Alfaomega.