TECNOLÓGICO NACIONAL DE MÉXICO SEDE ORIZABA

PERIODO: ENERO- JUNIO 2023

CLAVE DE CLASE: 3a3A

CARRERA: ING. INFORMÁTICA

HORARIO DE CLASE: 15:00 A 16:00

MATERIA: ESTRUCTURA DE DATOS

EQUIPO: 8

TEMA:

REPORTE DE PRACTICAS TERCERA UNIDAD

(**ESTRUCTURALES LINEALES**)

FACILITADORA:

MARIA JACINTA MARTINEZ CASTILLO

ASPIRANTES:

ALBERTO YERIEL GÓMEZ LÓPEZ (**21010189**)

GAEL OCTAVIO MORALES VALDEOLIVAR (**21010203**)

AXEL REYES GUEVARA (**21010213**)

INTRODUCCION

Una Pila es una clase especial de lista en la cual todas las inserciones y borrados tienen lugar en un extremo denominado extremo, cabeza o tope. Otro nombre para las pilas son listas FIFO (último en entrar, primero en salir) o listas pushdown (empujadas hacia abajo).

El modelo intuitivo de una pila es un conjunto de objetos apilados de forma que al añadir un objeto se coloca encima del ultimo añadido y para quitar un objeto del montón hay que quitar antes los que están por encima de él. Un tipo de dato abstracto.

Una Cola es otro tipo especial de lista en el cual los elementos se insertan por un extremo (el posterior) y se suprimen por el otro (el anterior o frente). Las colas se conocen también como listas FIFO (primero en entrar, primero en salir).

Las operaciones para las colas son análogas a las de las pilas. Las diferencias sustanciales consisten en que las inserciones se hacen al final de la lista, y no al principio, y en que la terminología tradicional para colas y listas no es la misma.

COMPETENCIA ESPECÍFICA

Comprende y aplica estructuras de datos lineales para solución de problemas

MARCO TEORICO

3.0 Clases genéricas, arreglos dinámicos (**memoria dinámica**)

Las clases genéricas proporcionan los medios para describir el concepto cualquier otra clase en forma independiente de su tipo. Así podemos crear instancias de objetos con tipos específicos de la clase genérica**.**

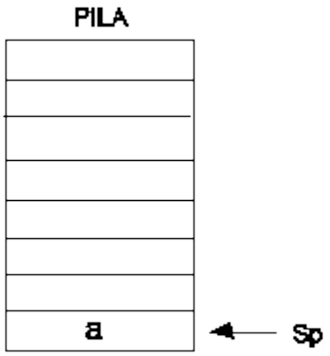
La memoria dinámica que se almacena en el heap es aquella que se utiliza para almacenar datos que se crean en el medio de la ejecución de un programa.

3.1 Pilas

La pila es una secuencia de elementos del mismo tipo en la que el acceso a la misma se realiza por un único lugar denominado cima.

3.1.1 Representación en memoria

Debemos definir el tamaño máximo de la pila, además de un apuntador al último elemento insertado en la pila el cual denominaremos SP. La representación gráfica de una pila es la siguiente:



3.1.2 Operaciones básicas

Las operaciones básicas son push (que introduce un elemento en la pila), pop (que saca un elemento de la pila), peek (consulta el primer elemento de la cima de la pila), isEmpty (que comprueba si la pila está vacía) y search (que busca un determinado elemento dentro de la pila y devuelve su posición dentro de ella).

3.1.3 Aplicaciones

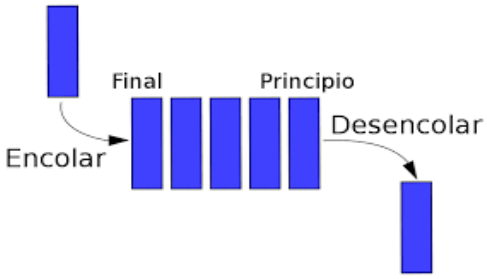
- Llamadas a subprogramas
- Paso de programas recursivos e iterativos
- Equilibrado de símbolos
- Tratamientos de expresiones aritméticas
- Ordenación rápida

3.2 Colas

Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista.

3.2.1 Representación en memoria

Donde debemos definir el tamaño de la cola y dos apuntadores**,** uno para accesar el primer elemento de la lista y otro que guarde el último.



3.2.2 Operaciones básicas

Las operaciones básicas son push (que introduce un elemento en la pila), pop (que saca un elemento de la pila), peek (consulta el primer elemento de la cima de la pila), isEmpty (que comprueba si la pila está vacía).

3.2.3 Tipos de colas: simples, circulares y bicolas

Colas simples: Se inserta por un sitio y se saca por otro, en el caso de la cola simple se inserta por el final y se saca por el principio.

Colas circulares: En las colas circulares se considera que después del último elemento se accede de nuevo al primero.

La bicola o doble cola: Es un tipo de cola especial que permiten la inserción y eliminación de elementos de ambos extremos de la cola.

3.2.4 Aplicaciones

Sistemas informáticos, transportes y operaciones de investigación (entre otros), donde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento.

3.3 Listas

Las listas son un tipo de colección que hereda de la interface collection, son una estructura de datos que respeta el orden en el cual fueron agregados los elementos, también permiten registros repetidos.

3.3.1 Operaciones básicas

- **EsVacia** Averiguar si la lista está vacía.
- **Insertar** Añade un elemento al principio de la lista.
- **Primero** Obtener el valor del primer elemento de la lista, también llamado cabeza.
- **Resto** Devuelve el trozo de lista resultado de eliminar el primer elemento de la lista.
- **Borrar** Borra el primer elemento de la lista.

3.3.2 Tipos de listas: simplemente enlazadas, doblemente enlazadas y circulares

Listas simplemente enlazadas: Es una estructura de datos en la que cada elemento apunta al siguiente. De este modo, teniendo la referencia del principio de la lista podemos acceder a todos los elementos de la misma.

Listas doblemente enlazadas: Una estructura de datos que consiste en un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados *enlaces*, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información.

Listas circulares: Es una lista lineal en la que el último nodo a punta al primero. Las listas circulares evitan excepciones en las operaciones que se realicen sobre ellas. No existen casos especiales, cada nodo siempre tiene uno anterior y uno siguiente.

3.3.3 Aplicaciones: Programa-Proyecto menú completo para el uso y manejo de pilas y colas con sus respectivas implementaciones.

https://github.com/AlbertG8/Datos

MATERIAL Y EQUIPO

- Computadora con Windows 7 en adelante
- IDE utilizado (IntelliJ IDEA 2023.1 (Comunity Edition) OpenJDK 1.17 UTF-8)
- Memorias USB

DESARROLLO DE LA PRÁCTICA

Para poder realizar los programas de pilas y colas con sus respectivos métodos que son el (isEmpty, push, pop y peek) para esto primero creamos una clase que se llame ya sea PilaTDA o ColaTDA y pondremos sus respectivos métodos como los de la siguiente imagen

```
1  package ColaA;
2
3  public interface ColaTDA <T> {
4
5       public boolean isEmpty();
6
7           public void push(Object dato);
8           public T pop();
9           public T peek();
.0
.1  }
.2
```

Posteriormente a esto crearemos la clase que queramos realizar en este caso es ColaA y en el public class importaremos el interface y pondremos lo siguiente

```
1  package ColaA;
2
3  import Tools.toolsList;;
4
5  public class ColaA  <T> implements ColaTDA {
6
7          private T cola[];
8          private byte u;
9
10         public ColaA(int max){
11             cola =(T[]) (new Object[max]);
12             u=-1;
13         }
```

Aquí declaramos las variables globales que son de tipo genérica cola[ ] y de tipo byte que es u, se crea un constructor parametrizado el cual recibirá el tamaño máximo de la cola posteriormente se inicializa la variable u que nos indicara la cantidad de elementos almacenados dentro de la cola.

Posteriormente se crearan sus métodos los cuales ya fueron mencionados y son los que se pueden encontrar en la interface los cuales son los siguientes

```
18         @Override
19         public boolean isEmpty() {
20             return (u==-1);
21         }
22
23
24
25         public boolean isSpace(){
26             return(u<cola.length-1);
27         }
28
29         @Override
```

```
36          }
37
38          @Override
39          public T pop() {
40              T dato = cola[0];
41
42              for (int j=0; j<= u; j++){
43                  cola[j]= cola[j+1];
44              }
45              u--;
46
47              return dato;
48          }
49
50          @Override
51          public T peek() {
52              return (T) cola[0];
53          }
54
55          public String toString(){
56              return toString(u);
57          }
58
59          public String toString(int i){
60              return(i>=0)?"["+i+"]==>" + "" +cola[i]+""+toString(i-1):"";
61          }
62      }
```

Y esos son cada uno de sus métodos que deben contener la clase para poder funcionar adecuadamente y con forme al TDA.

RESULTADOS

Para poder ejecutar el código debemos que tener un menú con las opciones de nuestro TDA, este menú es uno de los que ya hemos posteriormente utilizado en otras clases y es el siguiente solo recibe un ligero cambio que a continuación se mencionara

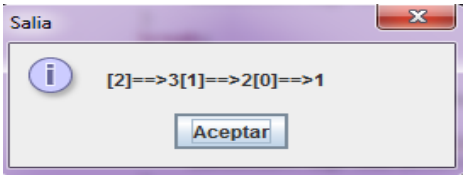
```
1  package ColaA;
2
3  import Tools.toolsList;
4  import ColaA.ColaA;
5  import javax.swing.*;
6
7  public class Test {
8
9      public static void main(String[] args) {
10             String menu = "Push,Pop,Peek,Free,Salir";
11             menu3(menu);
12         }
13         public static String boton(String menu) {
14             String valores[]=menu.split(",");
15             int n;
16             n = JOptionPane.showOptionDialog(null," SELECCIONA DANDO CLICK ", " M E N U",
17                     JOptionPane.NO_OPTION,
18                     JOptionPane.QUESTION_MESSAGE,null,
19                     valores,valores[0]);
20             return ( valores[n]);
21         }
22
23         public static void menu3(String menu)
24         {
25             ColaA<Integer> cola = new ColaA((byte) 10);
26
27             String sel="";
28             do {
29                 sel=boton(menu);
30                 switch(sel){
```

En la línea 25 se llama al programa de ColaA para así poder hacer de una las opciones del menú las cuales quedan así

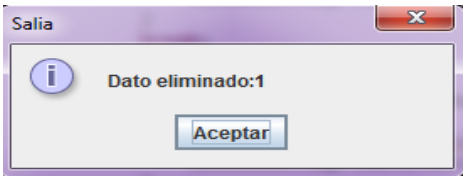
```
27             String sel="";
28             do {
29                 sel=boton(menu);
30                 switch(sel){
31                     case "Push":
32                         cola.push(toolsList.leerInt("Ingresa un valor"));
33                         toolsList.imprimePantalla(cola.toString());
34                         break;
35                     case "Pop":
36                         if(cola.isEmpty())
37                             toolsList.imprimeErrorMsg("Pila Vacia");
38                         else{
39                             toolsList.imprimePantalla("Dato eliminado:"+cola.pop());
40                         }
41                         break;
42                     case "Peek":
43                         if(cola.isEmpty())
44                             toolsList.imprimeErrorMsg("Pila Vacia");
45                         else {
46                             toolsList.imprimePantalla("Dato en el tope de la cola==>" + "" + co
47                         }
48                         break;
49                     case "Free":
50                         if(cola.isEmpty())
51                             toolsList.imprimeErrorMsg("Pila Vacia");
52                         else{
53                             cola = new ColaA((byte)10);
54                             toolsList.imprimePantalla("Pila vaciada");
55                         }
56                         break;
55                                 }
56                                 break;
57                             case "Salir":;  break;
58                         }
59                     }while(!sel.equalsIgnoreCase("Salir"));
60             }
61
62         }
```

Una vez el programa fue ejecutado saldría quedaría así

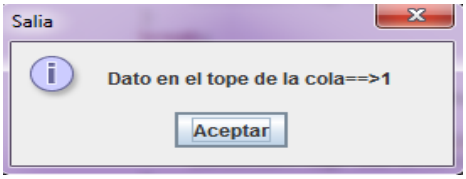
Si agregamos datos con el push quedaría así



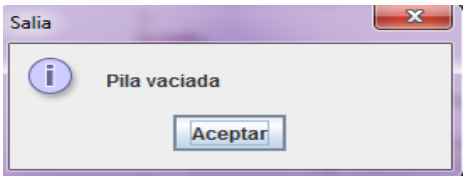
Si eliminamos un dato con el pop quedaría así



Si usamos el peek nos sale asi



Si usamos el free nos aparece este mensaje y para comprobar si esta vacia la cola usaremos de vuelta el pop y nos va adecir que esta vacia la cola



Y eso es todo lo que hace el programa cuando lo ejecutamos.

CONCLUSION

En conclusión nosotros los alumnos pudimos interactuar bien con los programas y saber cómo es que funciona cada uno de los métodos de pilas y colas los cuales son (Push, Pop, Peek, IsEmpty) y como es que se implementan y cómo interactúan con el código, en lo particular nos gustó en la forma de como la maestra nos enseñó ya que pudimos ver bien cómo funcionan y como lo fuimos desarrollando todos al punto que terminábamos los códigos y ver cómo funcionaban.

BIBLIOGRAFIAS

Resumen, P. (2019, 20 noviembre). *Clases genéricas en Java*. PC Resumen. https://www.pcresumen.com/menu-software/35-lenguajes-de programacion/java/84-clases-genericas-en-java

Merino, A. (s. f.). *3. El heap y la memoria dinámica*. https://www.it.uc3m.es/pbasanta/asng/course_notes/ch06s03.html#:~:text=La%20memoria%20din%C3%A1mica%20que%20se,los%20datos%20de%20un%20programa.

Representación en memoria de una pila. (2013b, enero 9). Tareas Universitarias. https://tareasuniversitarias.com/representacion-en-memoria-de-una-pila.html

Cuenca, M. S. Y. J. L. (s. f.). *La estructura de datos pila en java. Clase Stack del api java. Ejemplo simple y ejercicios resueltos (CU00923C)*. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=608:la-estructura-de-datos-pila-en-java-clase-stack-del-api-java-ejemplo-simple-y-ejercicios-resueltos-cu00923c&catid=58&Itemid=180#:~:text=Las%20operaciones%20b%C3%A1sicas%20son%20push,su%20posici%C3%B3n%20dentro%20de%20ella).

DSTool: Herramienta para la programación con estructuras de datos. (s. f.). http://www.hci.uniovi.es/Products/DSTool/colas/colas-operaciones.html

ACTIVIDAD COMPLEMENTARIA

Hacer un cuadro comparativo sobre las diferencias en pilas y colas

Pilas	Colas
Una Pila tiene un modelo LIFO (last-in / first-out), el cual quiere decir que el último elemento en entrar a la Pila será el primero en salir.	Una cola tiene un modelo FIFO (first-in / first-out), el cual quiere decir que el primer elemento en entrar, será el primer elemento en salir.
Una pila los elementos se agregan y se eliminan en el mismo extremo.	Una cola los elementos se agregan de un extremo de la cola llamado "final", y se eliminan del otro extremo de la cola llamado "frente".