

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE TRABAJO PRÁCTICO

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Estructuras de datos y algoritmos</i>				
TÍTULO DEL TRABAJO:	<i>HEAPS</i>				
NÚMERO DE TRABAJO:	<i>3</i>	AÑO LECTIVO:	<i>2023A</i>	NRO. SEMESTRE:	<i>III</i>
FECHA DE PRESENTACIÓN	<i>17/06/2023</i>	HORA DE PRESENTACIÓN			
INTEGRANTE (s) <i>Carrasco Choque Arles Melvin</i> <i>Chara Condori Jean Carlo</i>				NOTA (0-20)	
DOCENTE(s): <i>Teoría : Mg. Karim Guevara</i> <i>Laboratorio: Edith Cano</i>					

INTRODUCCIÓN
<p>PROPÓSITO:</p> <p>El propósito fue resolver los ejercicios planteados con los conocimientos que adquirimos en el laboratorio y teoría, estos dan un margen del progreso que tenemos hasta el momento. Por ejemplo para el desarrollo de este trabajo utilizamos la estructura heap y colas de prioridad</p> <p>METODOLOGÍA:</p> <p>Para resolver los ejercicios cada uno repaso primero la teoría y retroalimentando con ejercicios anteriores para tener mas practica.</p> <p>RESULTADOS:</p> <p>Se ha reforzado los conocimientos en estructuras de datos como lo son los heaps y colas prioritarias con la práctica.</p>
MARCO CONCEPTUAL
<p>Se ha utilizado la estructura heap, la cual es una estructura de datos que se utiliza para almacenar y administrar dinámicamente un conjunto de elementos. También conocida como "montículo binario". También se ha utilizado las colas prioritarias, la cual es una estructura de datos que permite almacenar elementos con una prioridad determinada y acceder a ellos en función de su prioridad.</p>
SOLUCIONES Y PRUEBAS

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

LINK DEL REPOSITORIO: https://github.com/JeanChara/eda_practica_03

1.EJERCICIO 5: Construya una cola de prioridad que utilice un heap como estructura de datos. Para esto realice lo siguiente:

- Implemente el TAD Heap genérico que este almacenado sobre un ArrayList con las operaciones de inserción y eliminación. Este TAD debe de ser un heap maximo.
- Implemente la clase PriorityQueueHeap generica que utilice como estructura de datos el heap desarrollado en el punto anterior. Esta clase debe tener las operaciones de una cola tales como:
 - a. Enqueue (x, p) : inserta un elemento a la cola 'x' de prioridad 'p' a la cola. Como la cola esta sobre un heap, este deberá ser insertado en el heap-max y reubicado de acuerdo a su prioridad.
 - b. Dequeue() : elimina el elemento de la mayor prioridad y lo devuelve. Nuevamente como la cola está sobre un heap-max, el elemento que debe ser eliminado es la raíz, por tanto, deberá sustituir este elemento por algún otro de modo que se cumpla las propiedades del heap-max.
 - c. Front() : solo devuelve el elemento de mayor prioridad.
 - d. Back(): sólo devuelve el elemento de menor prioridad. NOTA: tenga cuidado en no romper el encapsulamiento en el acceso a los atributos de las clases correspondientes.

Resolución:

Clase Heap:

```
import java.util.*;
import myExceptions.ExceptionNotFound;

public class Heap <T extends Comparable <T>> {
    ArrayList<T> heapList;
    public Heap (){
        heapList = new ArrayList<T>();
    }
}
```



Se implemento excepciones para los casos se desee remover algo y no existan objetos genericos. Se crea el ArrayList<T> heapList, el cual almacenara a los objetos genericos.

```
public void insert(T item) {
    heapList.add(item);
    heapifyUp(heapList.size()-1);
}
public T peek(){
    return heapList.get(0);
}
public T remove() throws ExceptionNotFound{
    if (isEmpty()) {
        throw new ExceptionNotFound("No hay elementos en heap");
    }
    T datoAuxMayor = peek();
    int indexFinal = heapList.size()-1;
    heapList.set(0,heapList.get(indexFinal)); //intercambio
    heapList.remove(indexFinal);
    heapifyDown(0);

    return datoAuxMayor;
}
```

```
30 private void heapifyUp(int index){
31     int indexPadre = (index-1)/2; // formula
32
33     // recorremos y comparamos los heap siempre y cuando halla mas de 1
34     while(index > 0 && heapList.get(index).compareTo(heapList.get(indexPadre)) > 0){ // heap maximo. si es mayor, intercambiamos con el padre
35         intercambio(index,indexPadre);
36         index = indexPadre; // cambiamos de nodo a comparar
37         indexPadre = (index-1)/2; // re calculamos el nuevo index del padre
38     }
39
40 }
41 private void heapifyDown(int index){
42     int indexIzquierdo = 2 * index + 1;
43     int indexDerecho = 2 * index + 2;
44     int mayorIndex = index;
45
46     // TAD Heap maximo
47     if (indexIzquierdo < heapList.size() && heapList.get(indexIzquierdo).compareTo(heapList.get(mayorIndex)) > 0) { // vemos si nuestro nodo izquierdo es mayor a
48         mayorIndex = indexIzquierdo; // si es asi, reemplazamos el index
49     }
50
51     if (indexDerecho < heapList.size() && heapList.get(indexDerecho).compareTo(heapList.get(mayorIndex)) > 0) {
52         mayorIndex = indexDerecho;
53     }
54
55     if (mayorIndex != index) {
56         intercambio(index, mayorIndex); // intercambiamos lugares
57         heapifyDown(mayorIndex); // hasta que ninguno sea mayor que el elemento o que sea una hoja
58     }
59 }
```

Se crea el metodo insert, el cual añadira el dato generico a nuestro heapList, y llamara a la funcion heapifyUp, la cual en caso haya mas de un elemento, comenzara a ordenar los nodos hasta llegar a ser un heap maximo. Tenemos la funcion peek() en el cual retornara nuestro elemento cima. Asimismo, la

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

funcion remove(), tiene una precondition en la cual si no hay elementos nos retorna un error. En caso existan elementos se intercambia el menor elemento con el mayor y se elimina el mismo. Luego llama a la funcion HeapifyDown para ordenar nuestro nuevo heap, puesto que se ha eliminado un elemento.

```

private void intercambio(int i, int j){
    T aux = heapList.get(i);
    heapList.set(i, heapList.get(j));
    heapList.set(j, aux)
}

public boolean isEmpty(){
    return heapList.isEmpty();
}

```

Tenemos 2 funciones auxiliares las cuales serán de utilidad para nuestras funciones heapifyUp, HeapifyDown. El método intercambio intercambiara 2 elementos del heapList. La función isEmpty nos retorna true o false si nuestro heapList se encuentra vacío.


```
public T getMax() throws ExceptionNotFound {
    if (isEmpty()) {
        throw new ExceptionNotFound("No hay elementos en heap");
    }
    return heapList.get(0);
}

public T getMin() throws ExceptionNotFound {
    if (isEmpty()) {
        throw new ExceptionNotFound("No hay elementos en heap");
    }
    T min = heapList.get(0);
    for (int i = 1; i < heapList.size(); i++) {
        if (heapList.get(i).compareTo(min) < 0) {
            min = heapList.get(i);
        }
    }
    return min;
}

public int getSize() {
    return heapList.size();
}
```

```
86     ArrayList<T> getArray() {
87         return heapList;
88     }
89     public T get(int index) {
90         if (index < 0 || index >= heapList.size()) {
91             throw new IndexOutOfBoundsException("Index is out of bounds");
92         }
93
94         return heapList.get(index);
95     }
96
97     public boolean isEmpty(){
98         return heapList.isEmpty();
99     }
100
101
102 }
```

Tenemos los gets los cuales retornaran los valores maximos y minimos de nuestro heapList. un get para obtener el tamaño de nuestro heapList. El metodo getArray el cual retorna nuestro heapList y finalmente el metodo “get”, el cual recibe un indice y retorna el elemento con dicho indice de nuestro heapList.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 6</p>

También se creó la otra clase QueuePriorityHeap, esta tendría que poner el árbol en función de la prioridad y los números, no pudo ser implementada correctamente, porque lo que por ahora tendría la misma función del heap



```

1  public class PriorityQueueHeap<T extends Comparable<T>> {
2      private Heap<T> heap;
3
4      public PriorityQueueHeap() {
5          heap = new Heap<>();
6      }
7
8  public void enqueue(T item,int prioridad) {
9      heap.insert(item);
10     //llama al metodo de la clase heap
11
12 }
13 public T dequeue() {
14     if (isEmpty()) {
15         throw new IllegalStateException("La cola de prioridad está vacía.");
16     }
17
18     return heap.remove();
19 }
20 public T front() {
21     if (isEmpty()) {
22         throw new IllegalStateException("La cola de prioridad está vacía.");
23     }
24     return heap.getMax();
25 }
26 public T back() {
27     if (isEmpty()) {
28         throw new IllegalStateException("La cola de prioridad está vacía.");
29     }
30
31     return heap.getMin();
32 }

```

En conclusión la clase Heap lo que va a ser es recibir a los valores y ordenarlo de mayor a menor en cada momento que se agregue, o sea que si ingresamos un nuevo valor este se va a poner en la cabeza y va a ir recorriendo para ver en qué posición se queda, como es una cola al momento de remover lo va a ser con la cabeza ya que en teoría es el elemento con un valor mayor.

Prueba en clase test:

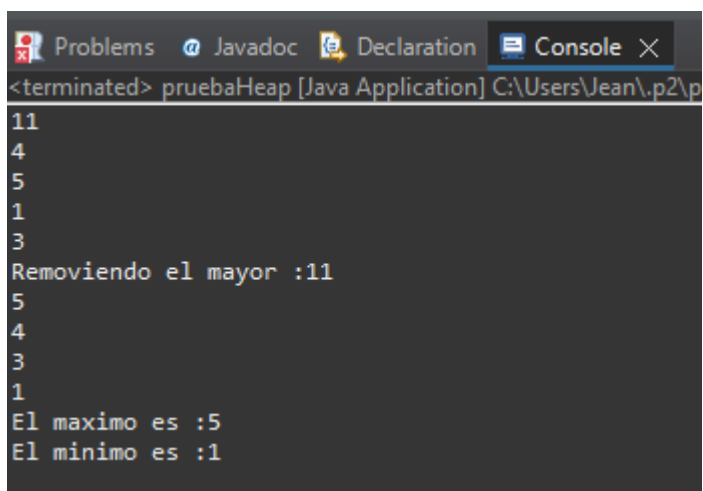
	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 7</p>

```

1  public class test {
2
3  public static void main(String[] args) throws ExceptionNoFound {
4      Heap<Integer> maxHeap = new Heap<>();
5      maxHeap.insert(4);
6      maxHeap.insert(11);
7      maxHeap.insert(5);
8      maxHeap.insert(1);
9      maxHeap.insert(3);
10
11     for(int i=0; i < maxHeap.getArray().size(); i++) {
12         System.out.println(maxHeap.getArray().get(i));
13     }
14     System.out.println("Removiendo el mayor :"+maxHeap.remove());
15
16     for(int i=0; i < maxHeap.getArray().size(); i++) {
17         System.out.println(maxHeap.getArray().get(i));
18     }
19     System.out.println("El maximo es :"+maxHeap.getMax());
20     System.out.println("El minimo es :"+maxHeap.getMin());
21
22 }
23 }
24

```

Ejecución:



```

Problems  Javadoc  Declaration  Console  X
<terminated> pruebaHeap [Java Application] C:\Users\Jean\.p2\p
11
4
5
1
3
Removiendo el mayor :11
5
4
3
1
El maximo es :5
El minimo es :1

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

LECCIONES APRENDIDAS Y CONCLUSIONES

Se ha aplicado los conceptos vistos en teoría (colas prioritarias) y la investigación acerca de la estructura Heap reforzando los conocimientos con la práctica. Se ha visto la importancia de utilizar estructuras de datos adecuadas para manejar elementos con prioridades. El uso de un heap o montículo nos ha permitido acceder eficientemente a los elementos con la mayor o menor prioridad según fue necesario.

REFERENCIAS Y BIBLIOGRAFÍA

<https://www.geeksforgeeks.org/heap-data-structure/>
<https://www.geeksforgeeks.org/priority-queue-class-in-java/>