Introducción a Queue

¡Comencemos!

¿Qué aprenderás?

- Aplicar el uso de Queue para resolver problemas cotidianos dentro del mundo de la programación.
- Crear distintas implementaciones de Queue para ocupar métodos como agregar, eliminar y recorrer datos.

Introducción

Debemos partir definiendo que Queue es una palabra proveniente del inglés que significa "cola" o "colocar en la lista". Una "cola" es una colección ideal para contener elementos antes del procesamiento. Es una estructura lineal que sigue un orden particular en el que se realizan las operaciones. Un buen ejemplo es una cola para un local de comida rápida, donde el primero en pagar es el primero en servirse.

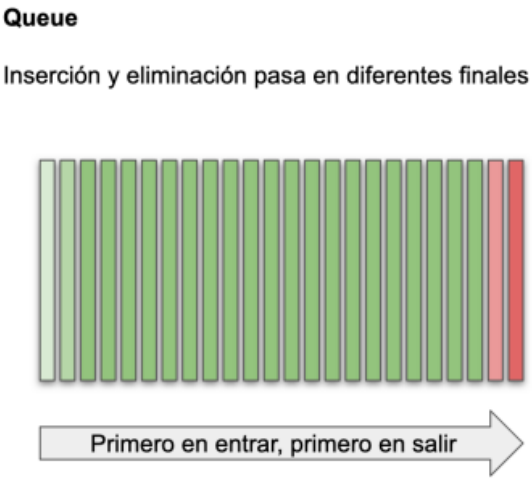

Imagen 1. Funcionamiento de un encolamiento.
Fuente: Desafío Latam

¡Vamos con todo!

Colección Queue

La colección Queue se utiliza para mantener los elementos a punto de ser procesados y proporciona varias operaciones como la inserción, eliminación, etc. Es una lista ordenada de objetos con su uso limitado para insertar elementos al final de la lista y eliminar elementos desde el principio de lista, es decir, sigue el principio "Primero en entrar, primero en salir" FIFO (First In First Out).

Al ser una interfaz, la cola necesita una clase concreta para la declaración y las clases más comunes son PriorityQueue y LinkedList en Java. Como recomendación cabe señalar que ambas implementaciones no son seguras para subprocesos y que PriorityBlockingQueue es una implementación alternativa si se necesitara una implementación segura para subprocesos.

Las colas típicamente, pero no necesariamente, ordenan elementos de manera FIFO (primero en entrar, primero en salir). Entre las excepciones se encuentran las colas de prioridad, que ordenan los elementos de acuerdo con sus valores.

Ejercicio guiado: Métodos de acceso posicional

Para generar un ejemplo, seguiremos usando la geografía como referencia, pero esta vez con enfoque a los continentes.

- **Paso 1**: Para crear una LinkedList<>() del tipo Queue hay que importar su implementación desde "util.java.LinkedList" en la parte superior de la clase y luego instanciarla como se muestra en la siguiente imagen.

```
import java.util.LinkedList;

       Queue continentes = new LinkedList<>();
```

- **Paso 2**: Para agregar elementos a encolar, podemos utilizar el método add(). Este método se usa para agregar elementos a la cola, más específicamente, al final de la cola. Se usa LinkedList para este caso pero dependerá de la prioridad según sea el caso de implementación de PriorityQueue.

```
  Queue continentes = new LinkedList<>();
  continentes.add("África");
  continentes.add("América");
  continentes.add("Europa");
  continentes.add("Oceanía");
  continentes.add("Asia");
  continentes.add("Antártica");
--------------------------
Impresión en pantalla:
[África, América, Europa, Oceanía, Asia, Antártica]
```

- **Paso 3**: Para poner un elemento específico del encolamiento continentes, podemos usar el método remove(). Este método elimina y devuelve el encabezado de la cola. Lanza NoSuchElementException cuando la cola está vacía.

```
System.out.println(continentes.remove("Antártica"));
System.out.println(continentes);
--------------------------
Impresión en pantalla:
[África, América, Europa, Oceanía, Asia]
```

- **Paso 4**: Para eliminar un encabezado, es decir, el primero de la lista se elimina, podemos usar el método poll(). Este método elimina y devuelve el encabezado de la cola. Devuelve nulo si la cola está vacía.

```
System.out.println(continentes.poll());
System.out.println(continentes);
--------------------------
Impresión en pantalla:
[América, Europa, Oceanía, Asia]
```

- **Paso 5**: Para obtener el encabezado de la cola sin eliminarlo podemos usar el método peek(). Este método se utiliza para ver el encabezado de la cola sin eliminarlo, devuelve nulo si la cola está vacía.

```
System.out.println("peek : " + continentes.peek());
System.out.println(continentes);
---------------------------
Impresión en pantalla:
peek: [América]
```

- **Paso 6**: Para encontrar un elemento, al igual que peek se puede hacer sin eliminar el objeto con el método element(): Este método es similar a peek(), pero lanza NoSuchElementException cuando la cola está vacía.

```
System.out.println("element: "+continentes.element());
System.out.println(continentes);
---------------------------
Impresión en pantalla:
element: [América]
```

- **Paso 7**: Para encontrar el tamaño de un encolado, podemos utilizar el método size(). Este método devuelve el número de elementos en la cola.

```
System.out.println(continentes.size());
---------------------------
Impresión en pantalla:
[4]
```