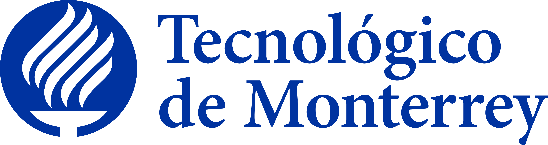
**TC1018 – Estructura de datos**

**Laboratorio 4. Stack y Queue**

**Parte I – Implementar la estructura Stack**

Implementa en java la estructura **Stack**, la cual deberá ser genérica <T>y dinámica. Recuerda que un Stack o Pila es una estructura LIFO (Last In First Out). Los métodos que debe tener son:

* Stack() – Construye un Stack vacío
* T peek() – Devuelve el elemento que se encuentra en la cima del Stack, sin removerlo. Si el Stack está vacío lanzaremos la excepción EmptyStackException ( )
* T pop() – Devuelve y remueve del Stack el elemento que se encuentra en la cima del Stack. Si el Stack está vacío lanzaremos la excepción EmptyStackException ( )
* void push(T t) – Ingresa el elemento t en el Stack.
* boolean isEmpty() – Nos devuelve True si el Stack está vacío.

**NOTA**: Para esta implementación, no heredarás de ninguna clase anterior. Utilizarás una clase anidada para los nodos del Stack ¿Qué tipo de nodo crees que sea más adecuado, de enlace simple o doble? Justifica tu elección, determinando la complejidad que tendrán los métodos de tu clase Stack.

**Parte II – Implementar la estructura Queue**

Implementa en java la estructura **Queue**, la cual deberá ser genérica <T> y utilizará un arreglo para construirla. ***El arreglo deberá tratarse como si fuera circular***, manteniendo referencia a la posición donde se encuentra el primer y último elemento del Queue. Los métodos que debe contener son los siguientes:

* Queue() – Construye un queue vacío, con una capacidad inicial default.
* Queue(int capacity) –Construye un queue vacío de la capacidad indicada.
* boolean isEmpty() – Devuelve verdadero si la fila está vacía
* T dequeue() – Devuelve y saca de la fila al elemento que se encuentra al inicio de la fila.
* void enqueue(T t) – Ingresa el elemento t al final de la fila. Si la fila está llena, duplica su capacidad e ingresa el elemento t.
* T front() – Nos devuelve al elemento que se encuentra al inicio de la fila, sin removerlo.
* T rear() – Nos devuelve quien es el elemento que se encuentra al final de la fila, sin removerlo.
* private void duplica() – Este método privado crea un nuevo arreglo con el doble la la capacidad del arreglo inicial, pasa los elementos que actualmente están en el queue y actualiza las variables de instancia.