PILAS Y COLAS

Maria C. Torres Madroñero
Profesora asociada
Departamento de Ciencias de la Computación y la Información

Objetivos
- Implementar las estructuras de datos STACK and QUEUE empleando una lista simple
- Implementar soluciones algorítmicas empleando las pilas y colas

Recursos requeridos
- PC
- IDE para JAVA o Python – el estudiante deberá seleccionar un lenguaje de programación para el desarrollo de las prácticas de laboratorio

Actividades preliminares al laboratorio
- Lectura de la guía

Marco teórico

PILAS

Una pila o STACK es una colección de objetos que son insertados o eliminados de acuerdo con el principio "ultimo en entrar – primero en salir". En ingles este principio se denomina LIFO (last-in first out), es decir, el ultimo objeto insertado en la pila es el único elemento al cual se tiene acceso. Similar a una pila de libros: usted puede remover el libro de la parte superior, de forma similar, solo puede agregar un libro en el tope de la pila.

Pila de libros

Una pila o STACK solo se permite dos operaciones para modificar la colección:
• PUSH: operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
• POP: operación de sacar o remover el objeto en el tope de la pila

Implementación de pilas con lista simple

El diagrama de clase resume los principales elementos para la implementación de una pila usando una lista simple.

Stack
- data: List
+ Stack() + size(): int + isEmpty(): Boolean + push(Object e) + pop(): Object + top(): Object

Para esto mantendremos un solo atributo, correspondiente a la lista simple, donde la cabecera de la lista (head) corresponderá al tope de la pila y cada nodo almacenará los datos de la pila. Por tanto, al tope de la pila podremos acceder empleando el método First() de nuestra lista simple. Cada vez que realicemos una operación PUSH estaremos agregando un elemento al tope, es decir, utilizaremos el método addFirst(). Cuando realicemos una operación POP invocaremos el método removeFirst(). Note que tanto addFirst(), removeFirst() y First() son métodos de nuestra clase lista simple que tienen una complejidad computacional de O(1), por tanto, todas las operaciones de nuestra pila también tendrán una complejidad O(1). En detalle los métodos a incluir son:
• Se incluye un constructor vacío
• Los métodos para determinar el tamaño (size) y si la pila esta vacía (isEmpty)

• Las operaciones de push y pop
• Y la operación de top que retorna, pero no eliminar el elemento u objeto en el tope de la pila

COLAS

Una cola o QUEUE es una colección de objetos que son insertados o eliminados de acuerdo con el principio "primero en entrar – primero en salir". En ingles este principio se denomina FIFO (first-in first out). Es decir, el objeto que lleva más tiempo en la cola es el único elemento al cual se tiene acceso. Similar a una cola de personas en una taquilla: usted puede atender a la persona que está de primero en la cola; y una persona solo puede ingresar al final de la cola.

Cola de personas

Una cola o QUEUE solo permite dos operaciones (acceso restringido tal como la pila) para modificar la colección:

- ENQUEUE: operación de insertar o encolar un nuevo objeto a la cola; este objeto se agregará al final de la colección
- DEQUEUE: operación de desencolar o remover un objeto de la cola; este corresponde al elemento que lleva más tiempo en la colección.

Implementación de colas con lista simple

El diagrama de clase resume los elementos que componen la clase QUEUE implementada con una lista simple.

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object

Solo se mantiene un atributo correspondiente a la lista simple, donde la cabecera (head) corresponderá al objeto que lleva más tiempo en la cola (first) y donde la cola (tail) corresponderá al objeto que mas recientemente entro a la colección. Por tanto, al primer elemento de la cola lo podremos acceder con el método first() de nuestra lista simple. Para implementar la operación ENQUEUE deberemos utilizar el método addLast() que se encarga de colocar un nuevo dato en la cola de la lista. Por su parte, la operación DEQUEUE se realizara con el método removeFirst(). Nuevamente, todas las operaciones tienen una complejidad computacional de O(1). En detalle los métodos a incluir son:

• Se incluye un constructor vacío
• Los métodos para determinar el tamaño (size) y si la pila esta vacía (isEmpty)
• Las operaciones de enqueue y dequeue
• Y la operación first() que retorna pero no eliminar el elemento u objeto al principio de la cola

Actividades

Problema 1 Implementación clase Stack

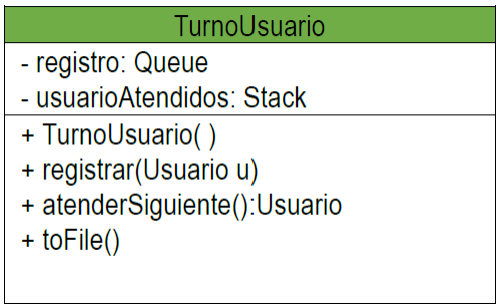
En esta primera etapa del laboratorio, implemente las clases Stack (pila implementada con lista simple), para ello haga uso de las clases nodo simple (Node) y lista simple (List) implementadas en el laboratorio 4. Para probar sus clases cree una pila con los valores enteros 2-4-6-8-10, imprima en pantalla la pila haciendo uso del método pop() de forma sucesiva.

Problema 2 Implementación clase Queue

En esta segunda etapa del laboratorio, implemente las clases Queue (cola implementada con lista simple), para ello haga uso de las clases nodo simple (Node) y lista simple (List) implementadas en el laboratorio 4. Para probar sus clases cree una cola con los valores enteros 2-4-6-8-10, imprima en pantalla la pila haciendo uso del método pop() de forma sucesiva.

Problema 3 Uso de las clases Stack y Queue

Ahora vamos a implementar un sistema de atención de usuarios para un banco. Para ello vamos a emplear la clase usuario que tenemos previamente programada, empleando solo los atributos nombre y id. El usuario ingresa al banco y pide su turno registrando su nombre y id. El sistema debe insertar este usuario en una cola. Cuando el asesor bancario este listo para atender un usuario, este debe extraerse de la cola e insertarse a una pila. El diagrama de clases para el sistema se presenta a continuación:



Descripción atributos

- registro: corresponde a una cola implementada con una lista simple, la cual almacena usuarios de acuerdo con su orden de llegada
- usuarioAtendidos: corresponde a una pila implementada con una lista simple, la cual almacena los usuarios que ya fueron atendidos

Descripción de métodos

- TurnoUsuario(): constructor que inicializa registro como un Queue y usuarioAtendido como un Stack
- registrar(Usuario u): inserta un nuevo usuario a la cola registro
- atenderSiguiente(): retorna el usuario que se encuentra de primero en la cola, lo elimina del registro y lo inserta en la pila usuarioAtendido
- toFile(): almacena los usuarios que quedaron en la cola en un archivo de texto usuariospendientes.txt y los que fueron atendidos en un archivo de texto usuariosatendidos.txt. Tip: haga uso de los getNombre() y getId() de la clase Usuario para guardar la información en el archivo de texto. El método toString puede causar errores ya que no vamos usar los demás atributos de la clase usuario.

Pruebas sugeridas para la presentación de la implementación:

- Ingrese 5 usuarios en la cola e invoque el método toFile()
- Invoque el método atenderSiguiente() dos veces y vuelva a llamar el método toFile()

Instrucciones de entrega

- La solución de los problemas debe desarrollarse en JAVA o Python. Los estudiantes tendrán la libertad de seleccionar el lenguaje de programación y plataforma para presentar la solución de los problemas.
- La solución debe emplear librerías nativas y se invita a los estudiantes a no usar código descargado de internet. Los laboratorios están diseñados para practicar los fundamentos teóricos; entre más código escriba el estudiante más fácil será su comprensión de los temas de clase.
- La solución se puede presentar en grupos de hasta 3 estudiantes.

- La solución de los problemas debe entregarse y sustentarse en el aula de clase o en la hora de asesoría a estudiantes. No se reciben soluciones por correo electrónico.