|  |
| --- |
| Programación III |
| Trabajo Práctico Nº2 |
| Mediciones del tiempo de ejecución |
|  |
|  |
|  |

|  |
| --- |
|  |

|  |
| --- |
| GRUPO 11 Turno Noche |
| Ramírez Emiliano |
| Ingni Sebastián |
| Rodríguez Matías |

Se han hecho pruebas del tiempo de ejecución para los métodos *Offer()* y *Poll()* usando TDA de cola; implementado con un vector estático (incluye método *resize()* para aumentar el tamaño), heredando de Lista, y conteniendo una lista. Los datos ingresados eran objetos *Integer* y se midió el tiempo de ejecución al encolar 1000000 de elementos y el tiempo de ejecución al desencolar la misma cantidad. Los resultados de las mediciones fueron:

|  |  |  |
| --- | --- | --- |
| Método | Implementación | Tiempo (mseg.) |
| Offer() | Cola Estática (ColaE) | 525 |
|  | Cola Conteniendo un Lista (ColaCL) | 629 |
|  | Cola heredando de Lista (ColaHL) | 739 |
| Poll() | Cola Estática (ColaE) | 6 |
|  | Cola Conteniendo un Lista (ColaCL) | 24 |
|  | Cola heredando de Lista (ColaHL) | 21 |

Del mismo modo, se midió el tiempo de ejecución de los métodos *Push()* y *Pop()*, usando TDA de pila; también implementado con un vector estático (incluye método *resize()* para aumentar el tamaño), heredando de Lista, y conteniendo una lista. Los objetos que se apilaron y desapilaron eran *Integer*. También se midió el tiempo de ejecución al apilar 1000000 de objetos, y luego desapilar esa cantidad. Los resultados obtenidos se ven en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Método | Implementación | Tiempo (mseg.) |
| Push() | Pila Estática (PilaE) | 386 |
|  | Pila Conteniendo un Lista (PilaCL) | 691 |
|  | Pila heredando de Lista (PilaHL) | 1210 |
| Pop() | Pila Estática (PilaE) | 18 |
|  | Pila Conteniendo un Lista (PilaCL) | 20 |
|  | Pila heredando de Lista (PilaHL) | 21 |

Cabe aclarar que al aumentar la cantidad de objetos a medir a 10000000 se capturaba la siguiente excepción:

*“Exception in thread “main” java.lang.OutOfMemoryError: Java heap space”*

Esto se debe a que se superó la cantidad de memoria disponible para la máquina virtual de Java. Una posible solución es aumentar esta memoria asignada cambiando el valor del parámetro *–Xmx*.

La manera en que se han hecho las mediciones es utilizando la biblioteca *java.util.Calendar*, de la siguiente manera:

Calendar ini = Calendar.getInstance();

For(int i=0; i < 1000000; i++){

*(Codigo a medir)…*

}

Calendar fin = Calendar.getInstance();

System.out.println( fin.getTimeInMillis() – ini.getTimeInMillis() );

Conclusión:

Al interpretar los tiempos de ejecución para cada TDA y para cada implementación, podemos ver que las estructuras del tipo estático tienen un tiempo de ejecución mucho menor que implementaciones utilizando memoria dinámica.

Además, implementando vectores estáticos se hace un mejor uso de la memoria, ya que al utilizar *nodos* para listas dinámicas, se debe reservar memoria para el dato en sí, y una cantidad extra para el puntero al siguiente nodo de la lista.-