Programación Avanzada - UNLAM
Trabajo Practico N° 2

Ejercicio 3-d

Los resultados de las pruebas obtenidos fueron los siguientes

- Cola Estática (ms): 28
- Cola Dinámica (ms): 515
- Pila Dinámica (ms): 30
- Pila Estática (ms): 24

La primera conclusión que se saca es que en ambos casos la implementación estática es más rápida que la dinámica. La causa de esto es que siempre en la forma estática el espacio de memoria este reservado desde que se declara el objeto, entonces por cada iteración se tiene que guardar el dato en su posición correspondiente. Mientras que, en la dinámica por cada iteración se debe primero crear el nodo, guardar el dato y luego enlazarlo con el resto de los nodos.

Sin embargo, en la implementación dinámica el tope de datos que puede guardar lo da el espacio de memoria disponible en el sistema, pero en la forma estática solo se puede guardar la cantidad de datos para los cuales fue reservado el espacio de memoria. Entonces para nuestro caso si quisiéramos guardar 1 000 001 datos la implementación estática no guardaría el último dato.

Ejercicio 3-e

Ejemplos de Polimorfismo

1. Para Pila

```
public static void main(String[] args) {
                String stringPrueba = "Programacion Avanzada";
                Pila pila = new PilaDinamica(); (1)
                pila.push(stringPrueba); (2)
                pila = new PilaEstatica(100); (3)
                pila.push(stringPrueba); (4)
        }
```

- En (1), se realizó una asigación polimórfica haciendo que una variable de tipo Pila (la cual es una interfaz) haga referencia a una pila de tipo dinámica.
- Seguidamente en (2), se invoca el método push de Pila que inserta un elemento en la misma. A través de enlazado dinámico, el compilador sabe en tiempo de ejecución cuál es el método que debe ser invocado.
- Luego, en (3), a la referencia pila se le asigna un nuevo espacio de memoria, esta vez de PilaEstatica, dejando la pila anteriormente creada sin referencias (será quitada de meroria por el recolector de basura).

- Nuevamente se invoca el método push, esta vez correspondiente a la implementación de PilaEstatica.

2. Para Cola

```
public static void main(String[] args) {
        String stringPrueba = "Programacion avanzada";
        Cola cola = new ColaCL();
        cola.offer(stringPrueba);
        System.out.println(cola.peek());
        cola = new ColaHL();
        cola.offer(stringPrueba);
        System.out.println(cola.peek());
}
```

Las explicaciones para este fragmento de código son muy similares a los ejemplos de pila. En este caso, se hacen asignaciones polimórficas entre las clases ColaHL y ColaCL, se inserta un nodo en cada cola y luego se muestra por pantalla cada uno de ellos.

Ejercicio 2-j

En la comparación de PilaCL vs PillaHL y ColaCL vs ColaHL se pueden realizar las mismas observaciones.

Para las clases CL (ContieneLista), el contenido del código es un poco más extenso que la otra implementación debido al hecho de que dentro de ellas es necesario declarar, inicializar y manejar (usar) una lista. La responsabilidad de la clase Lista recae en estas clases, siendo quienes “saben” cómo debe ser manejada.

En cambio, en las clases HL (HeredaLista), el comportamiento de lista es herdado, haciendo que no sea necesario manejarla internamente en la clase. El código es reducido ya que sólo deben ser sobrecargados los métodos abstractos de las interfaces de pila y cola respectivamente. Estas clases no “hacen uso” de la clase Lista sino que añaden atributos y funcionalidades a la misma.