Al comparar las implementaciones de PilaHL y PilaCL se denotan, a simple vista, varias diferencias. Para empezar, PilaHL sobrescribe dos métodos menos con respecto a PilaCL ya que los métodos isEmpty() y empty() se encuentran implementados en la clase Lista la cual, PilaHL hereda. Por otro lado, no es necesario instanciar una clase Lista como ocurre en PilaCL. Al ser PilaHL una subclase de Lista, esta posee los mismos atributos y métodos por lo que no es necesario la siguiente línea de código:

Lista pila = **new** Lista(); la cual PilaCL contiene ya que, de otra forma, no podría acceder a los atributos primerNodo y ultimoNodo.

Otro aspecto para tener en cuenta es que las implementaciones de los métodos push(), pop() y peek() en PilaHL llaman a los métodos heredados pushFront(), popFront() y getPrimerNodo().getDatos() (respectivamente).

@Override

**public** **void** push(Object dato) {

pushFront(dato);

}

@Override

**public** Object pop() **throws** ListaVaciaException {

**return** popFront();

}

@Override

**public** Object peek() {

**return** getPrimerNodo().getDatos();

}

En cambio, en PilaCL deben usarlos a través de la instancia de Lista mencionada anteriormente.

@Override

**public** **void** push(Object dato) {

listaP.pushFront(dato);

}

@Override

**public** Object pop() **throws** ListaVaciaException {

**return** listaP.popFront();

}

@Override

**public** Object peek() {

**return** listaP.getPrimerNodo();

}

El análisis es análogo para ColaHL y ColaCL. Mientras ColaHL sobrescribe menos métodos debido a la herencia con respecto a Lista, ColaCL puede utilizar los métodos isEmpty() y empty() implementados en Lista gracias a que ColaCL la contiene. Para los métodos offer(), poll() y peek() en ColaHL se llaman a los métodos heredados pushBack(), popFront() y getPrimerNodo().getDatos() (respectivamente).

@Override

**public** **void** offer(Object dato) {

pushBack(dato);

}

@Override

**public** Object poll() **throws** ColaVaciaException {

**return** popFront();

}

@Override

**public** Object peek() **throws** ColaVaciaException {

**return** getPrimerNodo().getDatos();

}

En cambio, en ColaCL se deben usar estos desde la instancia de Lista.

@Override

**public** **void** offer(Object dato) {

listaC.pushBack(dato);

}

@Override

**public** Object poll() **throws** ColaVaciaException {

**return** listaC.popFront();

}

@Override

**public** Object peek() **throws** RuntimeException {

**return** listaC.getPrimerNodo().getDatos();

}

En cuanto al rendimiento de los algoritmos, podemos ver dos constantes tanto para el tiempo de ejecución de las pilas y de las colas.

La primera constante es que el tiempo de respuesta de las implementaciones estáticas es significativamente menor que el de las dinámicas. Mientras, por ejemplo, una prueba dio un tiempo de 13 milisegundos para la pila estática, el mismo dio 50 milisegundos para la dinámica lo cual es una diferencia de 37 milisegundos. Diciéndolo de otra forma, se podría correr la prueba estática casi unas 4 veces más para recién igualar el otro tiempo de respuesta.

La otra constante es que los tiempos de respuesta de ambas implementaciones se mantienen estables yendo de los 13 a 17 ms para la estática y 48 a 50 ms para la dinámica.

Mientras que los tiempos de cola estática se mantienen estables entre los 12 y 14 milisegundos, cola dinámica tiene variaciones entre los 74 y 80 milisegundos.

Como resultado, podemos atribuir estos resultados a las implementaciones. Las estáticas si bien aplican resize() cada vez que el vector se queda sin espacio, este método se invoca significativamente menos veces que las instancias de la clase nodo que realizan las implementaciones dinámicas. En estas pruebas, al apilar y añadir a la cola 1000000 Integers, las implementaciones dinámicas tuvieron que realizar 1000000 de instancias de la clase Nodo mientras que las estáticas invocaron a resize() unas 20 veces. De ahí el tiempo de respuesta menor.

Para los ejemplos del empleo del polimorfismo, desarrollamos una clase EjemplosPolimorfismo donde están los ejemplos de pilas y colas.