**1. Codificar las siguientes interfaces:**

**a. Pila <<Interface>>:**

push( dato ) //apila un dato en el tope de la pila

pop( )//desapila el dato apuntado por el tope de la pila

peek( )//devuelve el dato apuntado por el tope de la pila, pero no lo desapila.

isEmpty( )// verificar si la pila esta o no vacía.

empty( ) //vaciar la pila

**b. Cola <<Interface>>**

offer(dato) // acola un dato en la cola.

poll() // desacola el dato de la cola.

peek() // devuelva el dato próximo a desacolarse, pero no lo desacola.

isEmpty( )// verificar si la cola esta o no vacía.

empty( )//vaciar la cola

**2.** **Codificar las siguientes clases:**

**a. PilaEstática implementando la interfaz Pila**

**b. PilaDinamica implementando la interfaz Pila**

**c. ColaEstatica implementando la interfaz Cola**

**d. ColaDinamica implementando la interfaz Cola**

**e. Lista**

pushBack(dato) //Inserta un elemento al final

popBack() // Retorna y borra el elemento al final

pushFront(dato) // Inserta un elemento al comienzo

popFront() // Retorna y borra el elemento al principio

remove(dato) // Elimina un elemento de un valor determinado.

reverse() //invierte el orden de los elementos en la lista

insertAt(posicion, dato ) // inserta un dato en la posición indicada.

eraseAt(posicion) //Eliminar por posición

isEmpty( )// verificar si la lista está o no vacía.

empty() //vaciar la lista

search(dato)

searchAt(posicion) // buscar por posición

size() // Retorna el numero de elementos en la lista

**f. Desarrollar la class PilaHL como una class derivada (subclase) de la class Lista y que implemente la interfaz Pila.**

**g. Desarrollar la class ColaHL como una class derivada (subclase) de la class Lista y que implemente la interfaz Cola.**

**h. Desarrollar la class PilaCL usando la relación contiene (composición) a la class Lista y que implemente la interfaz Pila.**

**i. Desarrollar la class ColaCL usando la relación contiene (composición) a la class Lista y que implemente la interfaz Cola.**

**j. Extraer conclusiones a partir de comparar las implementaciones de PilaHL vs. PilaCL. Ídem para ColaHL vs. ColaCL.**

**3. Test**

**a. Implemente un método test donde se pueda comprobar el correcto funcionamiento de todas las pilas y colas.**

**b. Apile 1000000 de Integers en una PilaEstatica y en una PilaDinamica y compare los tiempos de respuesta de estas operaciones en ambos diseños.**

**c. Ídem para ColaEstatica y Coladinamica.**

**d. Extraiga conclusiones.**

**e. Ejemplifique el empleo del polimorfismo entre los diferentes tipos de pilas. Ídem para las colas.**

**4. Anexo**

Uso de Calendar para medir rendimiento de algoritmos:

import java.util.GregorianCalendar ;

import java.util.Calendar ;

Calendar tIni = new GregorianCalendar ( ) ;

start code

.

.

end code

Calendar tFin = new GregorianCalendar ( ) ;

long diff = tFin .getTimeInMillis ( ) − tIni.getTimeInMillis ( ) ;

System.out.println ( diff ) ;

**2)**

**j) Conclusión:**

PilaHL vs. PilaCL:

Ambas clases tienen en común su relación con la clase Lista y su implementación de la interfaz Pila.

Sus diferencias residen en:

La PilaHL al heredar Lista, hereda  los métodos de la misma, sumándose a los métodos propios de la clase PilaHL. Estos se implementan dentro de los métodos de PilaHL utilizando un objeto llamador tipo PilaHL. En cambio, la PilaCL contiene un atributo tipo Lista, y lo utiliza como un objeto llamador para implementar los métodos de la clase Lista dentro de las funciones miembro de la clase PilaCL.

Sus tiempos de respuesta, según el Gregorian Calendar, son de:

PilaHL--> 2-3 milisegundos.

PilaCL -->4-5 milisegundos

Teniendo los algoritmos de la clase PilaHL  mejor rendimiento, en comparación con la clase PilaCL.

ColaHL vs. ColaCL:

Sus diferencias y puntos en común son los mismos que antes se mencionan en PilaHL vs. PilaCL, con la única diferencia que en lugar de de implementar la interfaz de Pila, se implementa la de Cola.

Sus tiempos de respuesta, según el Gregorian Calendar,  son de:

ColaHL---> 2-3 milisegundos.

ColaCL--->5-6 milisegundos.

Teniendo los algoritmos de la clase ColaHL  mejor rendimiento, en comparación con la clase ColaCL.

**3)**

**b)**

Tiempo de rendimiento en la operación de apilar 1000000 de Integers en una PilaEstatica y en una  PilaDinamica:

PilaEstatica--->35 milisegundos.

PilaDinamica--->785 milisegundos.

**c)**

Tiempo de rendimiento en la operación de encolar 1000000 de Integers en una ColaEstatica y en una ColaDinamica:

ColaEstatica--->36  milisegundos.

ColaDinamica--->766  milisegundos.

**d) Conclusiones:**

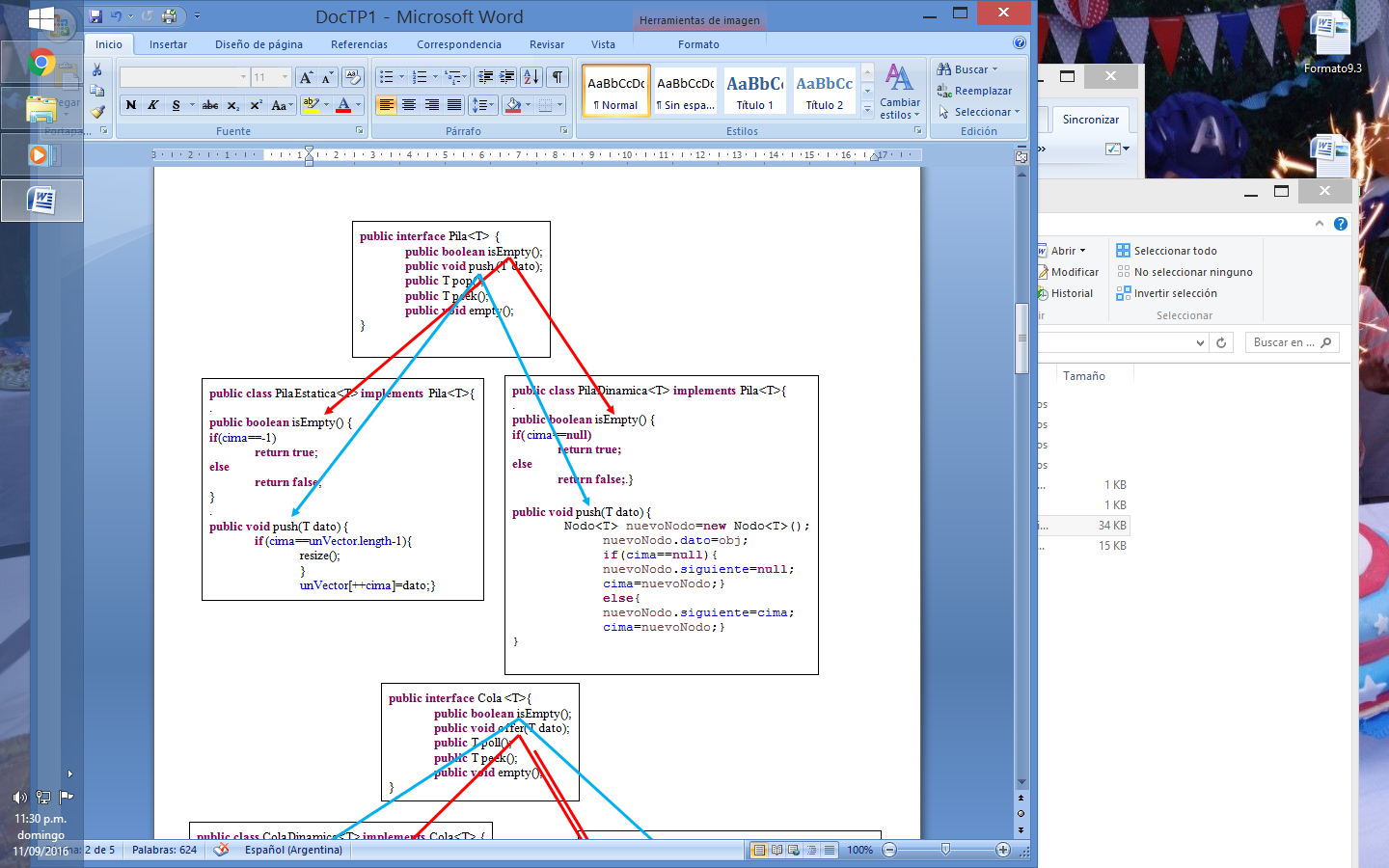
El resultado muestra que tanto PilaEstatica como ColaEstatica tienen un mejor rendimiento de algoritmos al realizar la operación de apilar y encolar, según corresponda, en comparación con PilaDinamica y ColaDinamica.

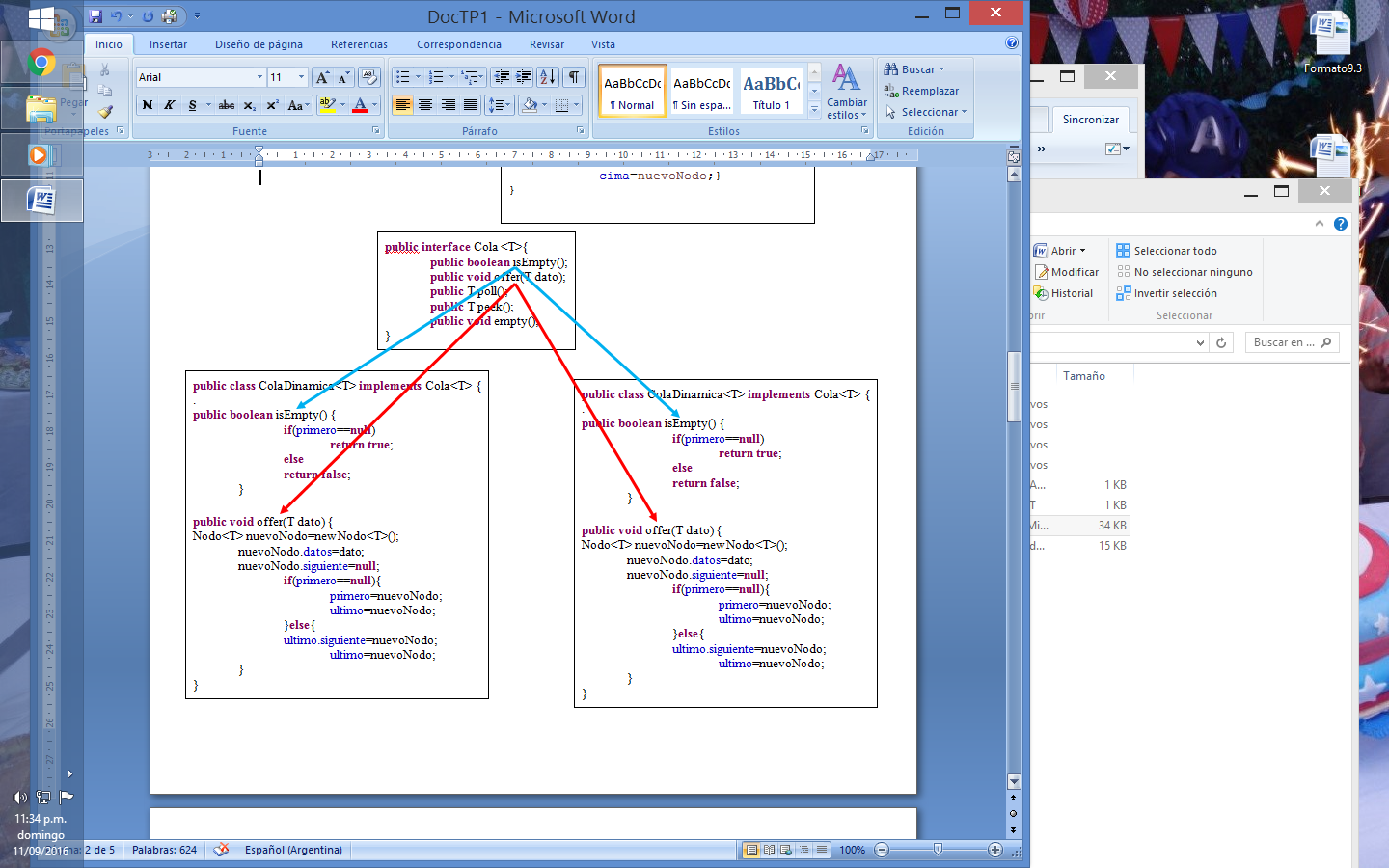
**e)**

Las clases PilaEstatica, PilaDinamica, PilaHL y PilaCL implementan la interfaz Pila, por lo tanto estas clases están obligadas a utilizar los mismos métodos (los que están en la interfaz) como es el caso de isEmpty(), push(T dato), entre otras.

Lo mismo es para las clases ColaEstatica, ColaDinamica, ColaHL y ColaCL, ya que implementan la interfaz Cola, utilizando métodos como peek(), empty(), entre otras.

Ejemplos:

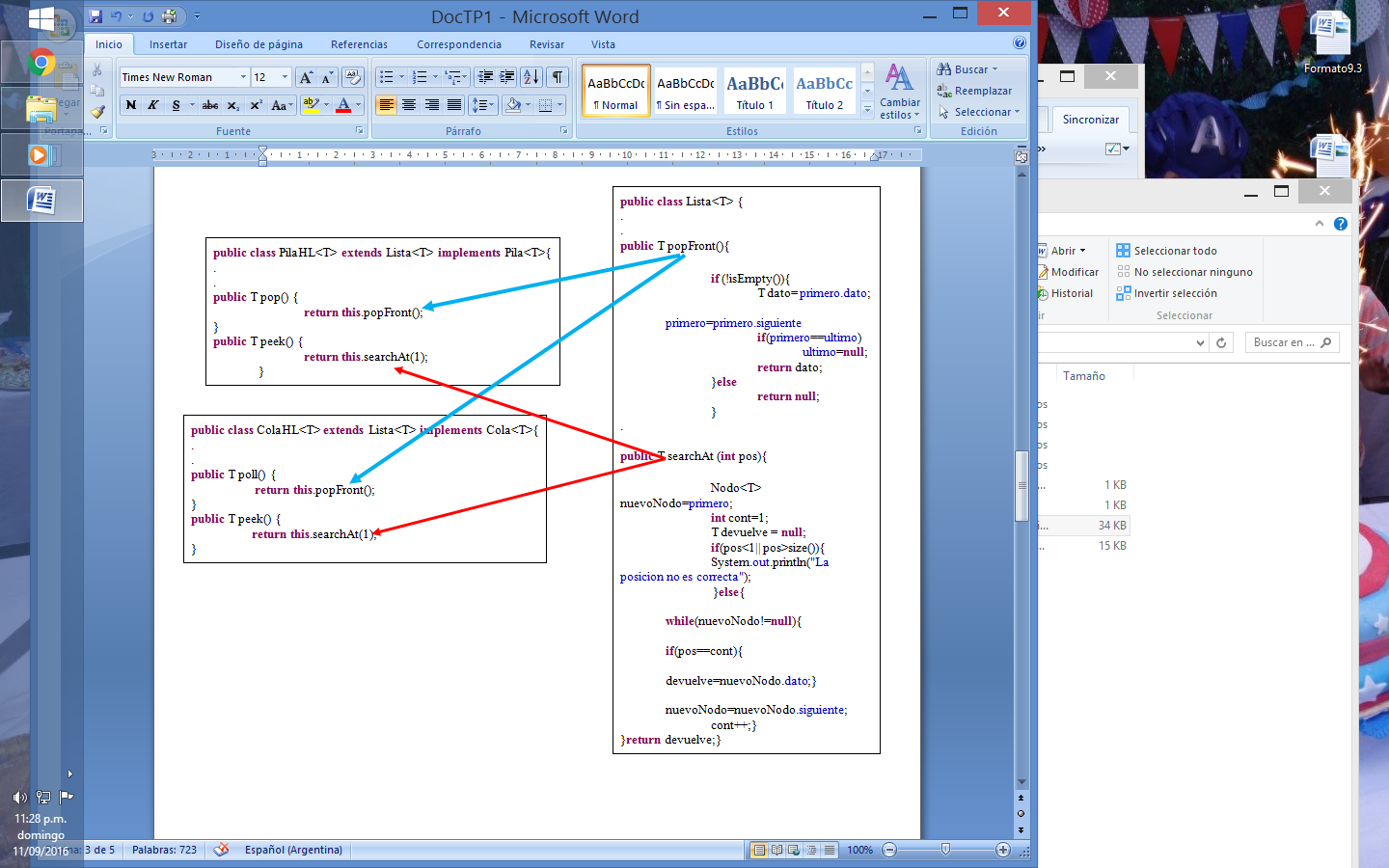




Además de implementar la interfaz Pila o Cola (según corresponda), las clases PilaHL, PilaCL, ColaHL y ColaCL guardan relación con la clase Lista:

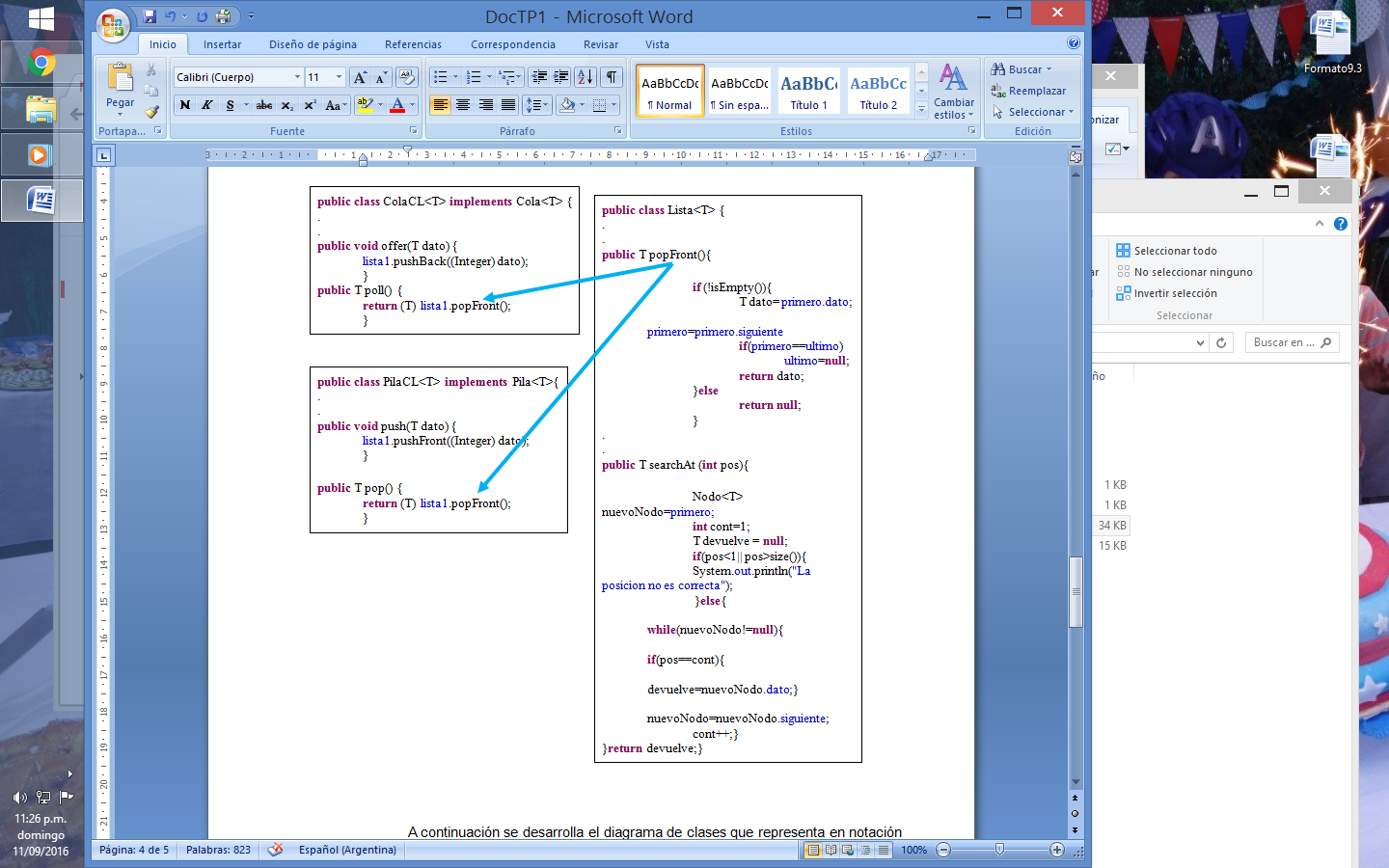
* PilaHL y ColaHL heredan Lista, es decir, heredan sus métodos y los utilizan dentro de la clase junto a sus métodos, como pueden ser popFront(), searchAt(), entre otros.

Ejemplos:



* Tanto PilaCL como ColaCL contienen Lista, por lo tanto, implementan sus atributos, los cuales utilizan en los métodos de sus respectivas  clases.

     Ejemplos:



A continuación se desarrolla el diagrama de clases que representa en notación UML las relaciones entre las clases que integran el package **TP1POO**:

