Trabajo Práctico Nº 12
Tema: COLAS (FIFO)

1. Mostrar que es lo que se escribe por cada uno de los siguientes segmentos de código, dado que C es un objeto de la clase Cola de enteros, P es un objeto de la clase Pila de enteros y X, Y y Z son variables enteras:

a)
```
Cola C=new Cola();
X=0;
Y=1;
C.insertar(X);
C.insertar(Y);
Y=C.borrar();
Z=Y + 5;
while(!C.estaVacia())
   {
   Z=C.borrar();
   System.out.println(Z);
   }
```

b)
```
Pila P=new Pila();
Cola C=new Cola();
X=0;
Y=1;
Z=X + Y;
while(Z < 10)
   {
   if((Z % 2) == 0)
      P.meter(Z);
   else
      C.insertar(Z);
   X=Y;
   Y=Z;
   Z=X + Y;
   }
System.out.println("La Pila contiene:");
while(!P.estaVacia())
   {
   Z=P.sacar();
   System.out.println(Z);
   }
System.out.println("La Cola contiene:");
while(!C.estaVacia())
   {
   Z=C.borrar();
   System.out.println(Z);
   }
```

2. En los ejercicios a, b, c, d, e y f mostrar el resultado de las operaciones dadas sobre las colas, usando la última implementación vista. Si se presenta desbordamiento positivo o negativo, marcar el correspondiente lugar; caso contrario mostrar los cambios en la cola.

a)

elementos	[1]	[2]	[3]	[4]	[5]	
	'V'	'W'	'X'	'Y'	'Z'	frente = 5
						ultimo = 4

C.insertar('J'); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

elementos	[1]	[2]	[3]	[4]	[5]	
						frente = …
						ultimo = …

b)

elementos	[1]	[2]	[3]	[4]	[5]	
	'V'	'W'	'X'	'Y'	'Z'	frente = 4
						ultimo = 5

C.insertar('K'); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

elementos	[1]	[2]	[3]	[4]	[5]	
						frente = …
						ultimo = …

c)

	[1]	[2]	[3]	[4]	[5]	
elementos	'V'	'W'	'X'	'Y'	'Z'	frente = 1 ultimo = 4

C.insertar('J'); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

	[1]	[2]	[3]	[4]	[5]	
elementos						frente = … ultimo = …

d)

	[1]	[2]	[3]	[4]	[5]	
elementos	'V'	'W'	'X'	'Y'	'Z'	frente = 3 ultimo = 3

C.borrar(); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

	[1]	[2]	[3]	[4]	[5]	
elementos						frente = … ultimo = …

e)

	[1]	[2]	[3]	[4]	[5]	
elementos	'V'	'W'	'X'	'Y'	'Z'	frente = 5 ultimo = 3

C.borrar(); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

	[1]	[2]	[3]	[4]	[5]	
elementos						frente = … ultimo = …

f)

	[1]	[2]	[3]	[4]	[5]	
elementos	'V'	'W'	'X'	'Y'	'Z'	frente = 2 ultimo = 1

C.borrar(); ¿Desbordamiento?..... ¿Desbordamiento Negativo? .....

	[1]	[2]	[3]	[4]	[5]	
elementos						frente = … ultimo = …

3. Escriba la clase Cola cuya declaración de atributos y métodos correspondan a la implementación Frente Fijo y Final Movible (tener en cuenta que al borrar debe correr todos los elementos).

4. En base a la implementación Frente y Final movible, escribir un método contar que devuelva la cantidad de elementos en la cola. Nota: no olvidar que si bien la cola se implementa como un arreglo, no debería manejarse como tal.

5. Agregar un método en la clase Cola que elimine los elementos repetidos de la cola.

6. La realización de una cola mediante un arreglo circular sacrifica un elemento del arreglo; esto se puede evitar añadiendo un nuevo atributo a la representación: *vacio* de tipo boolean. Implementar dicho cambio en la clase Cola.

7. Escribir un método que tenga como argumentos dos colas del mismo tipo y devuelva verdadero o falso si las dos colas son o no idénticas, respectivamente. Nota: no olvidar que si bien la cola se implementa como un arreglo, no debería manejarse como tal.

8. Se tiene una pila de enteros positivos. Con las operaciones básicas de pilas y colas, escribir un programa que coloque todos los elementos pares de la pila en la cola.

9. Escribir un programa que lea una cadena de caracteres, metiendo cada caracter en una pila a medida que se lee y añadiéndolo simultáneamente a una cola. Cuando se encuentre el final de la cadena, utilice las operaciones básicas de pilas y colas para determinar si la cadena es un palíndromo.

10. Un estacionamiento de autos contiene una sola línea, la cual tiene capacidad para 10 autos. Los autos llegan al extremo sur y salen por el norte, si un cliente llega a retirar su auto que no está muy hacia el norte, todos los autos al norte de su auto son retirados, el auto del cliente sale, y los otros autos son colocados de nuevo en el mismo orden en el que estaban originalmente. Cada vez que sale un auto, todos los autos que están al sur son movidos hacia adelante de tal manera que en todo momento todos los espacios vacíos están en la parte sur del estacionamiento.
Escriba un programa que procese un grupo de entradas. Cada entrada contiene una "e" para llegada o "s" para salida y una placa con el número de patente. Se asume que los autos llegan y salen en el orden especificado en la entrada. El programa debe imprimir un mensaje que los autos llegan y salen en el orden especificado en la entrada. El programa debe imprimir un mensaje cada vez que llega un auto, el mensaje debe especificar si hay o no espacio para el auto en el estacionamiento. Si no hay espacio para el auto, el auto espera hasta que haya uno, cuando queda espacio libre, debe imprimirse otro mensaje. Cuando sale un auto, el mensaje debe incluir el número de veces que fue movido el auto para permitir que otros autos salieran.

11. Una variación de la clase Cola es la estructura Bicola. Una bicola es un conjunto ordinal de elementos en el cual se pueden añadir o quitar elementos de cualquier extremo de la misma. Es, en realidad una *cola bidireccional*. Los dos extremos de una bicola se llaman *izquierdo* y *derecho*, respectivamente. Las operaciones básicas que definen una bicola son:

```
Bicola();                 //constructor que inicializa una bicola sin elementos
esVacia();                //devuelve verdadero si la bicola no tiene elementos
insertarIzquierda(x); //añade un elemento por extremo izquierdo
insertarDerecha(x);   //añade un elemento por extremo derecho
eliminarIzquierda();  //devuelve el elemento izquierdo y lo retira de la bicola
eliminarDerecha();    //devuelve el elemento derecho y lo retira de la bicola
```

Para representar una bicola se puede elegir una representación estática, con arreglos, en la cual se mantienen los elementos de la bicola con un arreglo circular y dos variables índice del extremo izquierdo y derecho, respectivamente.
a) Considere una bicola de caracteres representada por un arreglo circular. El arreglo consta de 9 posiciones. Los extremos actuales y el contenido de la bicola: Izquierdo:5, Derecha:7, Bicola:A, C, E. Escribir los extremos y la bicola según se vayan realizando las siguientes operaciones (en papel):

- Se añaden los elementos F, J a la derecha de la bicola.
- Se añaden los elementos R, W, V a la izquierda de la bicola.
- Se añade el elemento M a la derecha de la bicola.
- Se eliminan dos elementos a la izquierda.
- Se añaden los elementos K, L a la derecha de la bicola.
- Se añade el elemento S a la izquierda de la bicola.

b) Implementar la clase Bicola con los métodos básicos anteriormente detallados. Probar la clase.
c) A la estructura bicola se le pueden imponer restricciones respecto al tipo de entrada o al tipo de salida. Una bicola con restricción de entrada es aquella que sólo permite inserciones por uno de los dos extremos, pero permite la eliminación por los dos extremos. Una bicola con restricciones de salida es aquella que permite inserciones por los dos extremos, pero sólo permite retirar elementos por un extremo. Definir e implementar las operaciones para esta estructura.

12. Usar la siguiente información para los ejercicios a, b y c. Un sistema operativo particular almacena los trabajos en unas colas de espera para ejecutarlos de acuerdo con el siguiente esquema:
- Los usuarios del sistema que tengan prioridades relativas basadas en sus números de identificación:

Usuarios 0-99	mayor prioridad (presidente, vicepresidente y gerentes de la empresa)
Usuarios 100-199	siguiente más alta prioridad (subgerentes y asesores)
Usuarios 200-299	siguiente más alta prioridad (jefes de departamentos y secretarios)
.	
.	
.	
Usuarios 800-899	siguiente a la más baja (programadores)
Usuarios 900-999	más baja (trabajos que se ejecutaran solo los Sábados a las 3 de la madrugada).

- Dentro de cada grupo de prioridad los trabajos se ejecutan en el orden en que llegan al sistema.
- Si hay un trabajo de prioridad más alta, ejecutarlo antes que otro trabajo, si no, si hay un trabajo de prioridad siguiente a la más alta, ejecutarlo antes de cualquier trabajo de prioridad más baja y así sucesivamente. Esto es, un trabajo de prioridad más baja se ejecutará solo cuando no haya trabajos de prioridad más alta en espera.
- El sistema tiene un arreglo de cola FIFO para almacenar las colas de los distintos niveles de prioridad ColaTrabajos[10].

Para realizar los siguientes ejercicios, puede llamar a cualquiera de las operaciones de cola especificadas en el paquete de cola FIFO.
a) Escribir un método *añadirTrabajo* que reciba un número de identificación del usuario y un token (representando el trabajo que hay que ejecutar) y añada el token a la cola adecuada para el nivel de prioridad de dicho usuario.
b) Escribir un método *obtenerSiguienteTrabajo* que devuelva el token del trabajo que este en la cola de mayor prioridad para ejecución (El token debe ser quitado de la cola).
c) Hay que apagar el sistema para mantenimiento. Todos los trabajos que están esperando para ejecutarse se quitaran de las colas de trabajo. Sin embargo, este es un sistema muy amistoso por lo que notifica a los usuarios que sus trabajos serán destruidos y que deben por lo tanto, enviar los trabajos más adelante. El método *Notificar*, con *Token* e *IdMensaje* como entrada, realizará esta notificación. Escribir un método *limpiarTrabajos* que mande el mensaje a todos los usuarios con trabajos en las colas (llamar al método *Notificar* para hacer esto). Por supuesto, se envían primero los mensajes a los usuarios de mayor prioridad.