PRÁCTICA 2:

TAD Cola

Grupo de Prácticas: 02

Alumnos: Juan Blanco Martín y Adrián Herrero Artola

Turno de Laboratorio: 1L

Fecha de Entrega: 18/ 05 / 2020

Índice

1.- Análisis lógico

1.1.- Especificación Lógica del TAD Cola (LEG con Último)

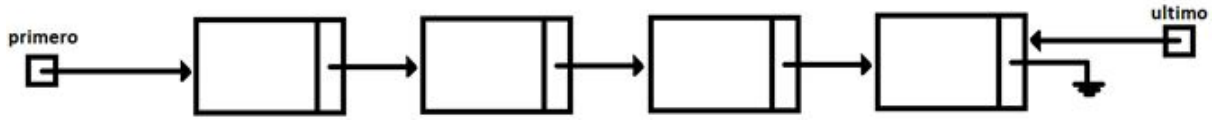
1.1.1.- Definición:

La cola es representada por una lista formada por una serie de nodos (cada nodo representa 1 único elemento (dato), genérico) enlazados entre sí por un puntero que hace referencia al siguiente nodo de la lista (o a null, si no hay más nodos, es decir, si nos encontramos en el final de la lista).

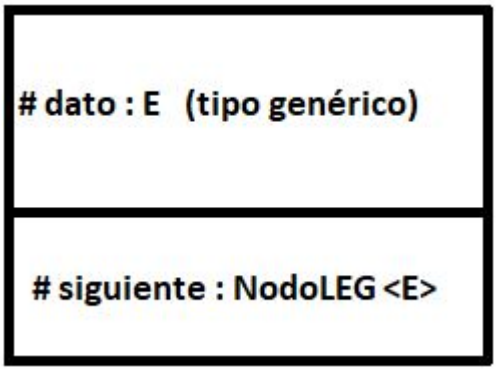
Justificación de la Lista Enlazada escogida para la implementación del TAD Cola:

Para implementar el **TAD Cola** como una lista enlazada hemos pensado en una **LEG con Último**, (a la que llamamos LECola) ya que una cola tiene frente y fin, y para acceder a estos nodos, la LEG con Último es la que mejor se ajusta porque consta del puntero al primer nodo de la lista y del puntero al último nodo de la lista. Además, mediante el método **insertarEnFin** podemos ***encolar*** un trabajo a la cola, ya que lo inserta en el fin de la cola.

Dibujo del TAD Cola (LEG (con Último) <E>):



NodoLEG <E> :



1.1.2.- Elementos :

Son nodos constituidos por un objeto de tipo genérico (tipo de dato que guarda el nodo, en este caso, introduciremos objetos de tipo Trabajo, por lo que el dato será una instancia de la clase Trabajo) y un puntero que referencia al siguiente elemento (nodo) de la lista.

1.1.3.- Tipo de organización:

Organización **lineal**, debido a que la relación entre elementos es de 1:1.

1.1.4.- Dominio de los elementos del TAD:

Cada elemento del TAD (es decir, cada nodo) es un objeto de tipo genérico.
La cola tendrá de *0* a *n* elementos (Dominio: **[0 - n]**), ya que la cola no tiene límite de elementos y también puede encontrarse vacía (caso en el que tendría 0 elementos).

1.1.5.- Operaciones Básicas:

Nombre: **contarElemCola**
Descripción: Método recursivo que devuelve el número de elementos que están almacenados en la cola.
Datos de entrada:
La referencia al nodo que queremos evaluar para ser contabilizado. En la primera invocación del método, esta referencia será la del primer nodo de la cola, y en posteriores invocaciones (en caso de que se den) será el siguiente al actual que estamos evaluando en tiempo de ejecución.
Datos de Salida: 0, en caso de que nos encontremos en el caso base, que es cuando invocamos al método con la referencia al siguiente del último, la cual apunta a null; 1 + el resultado de una nueva invocación al método con la referencia al siguiente del actual, en el caso general o recursivo.
Precondiciones: Ninguna. Si la cola está vacía, nos encontramos con que la referencia al primero de la cola es null y por tanto, devuelve 0.
Postcondiciones: Que en cada llamada al método se devuelva correctamente el entero que indica que nos encontramos en el caso base (entero = 0) o en el caso recursivo (entero = 1 + resultado de nueva invocación al método) para que, tras la fase de plegado, el entero que se devuelva a la primera invocación del método sea el número de elementos (nodos) de la cola.

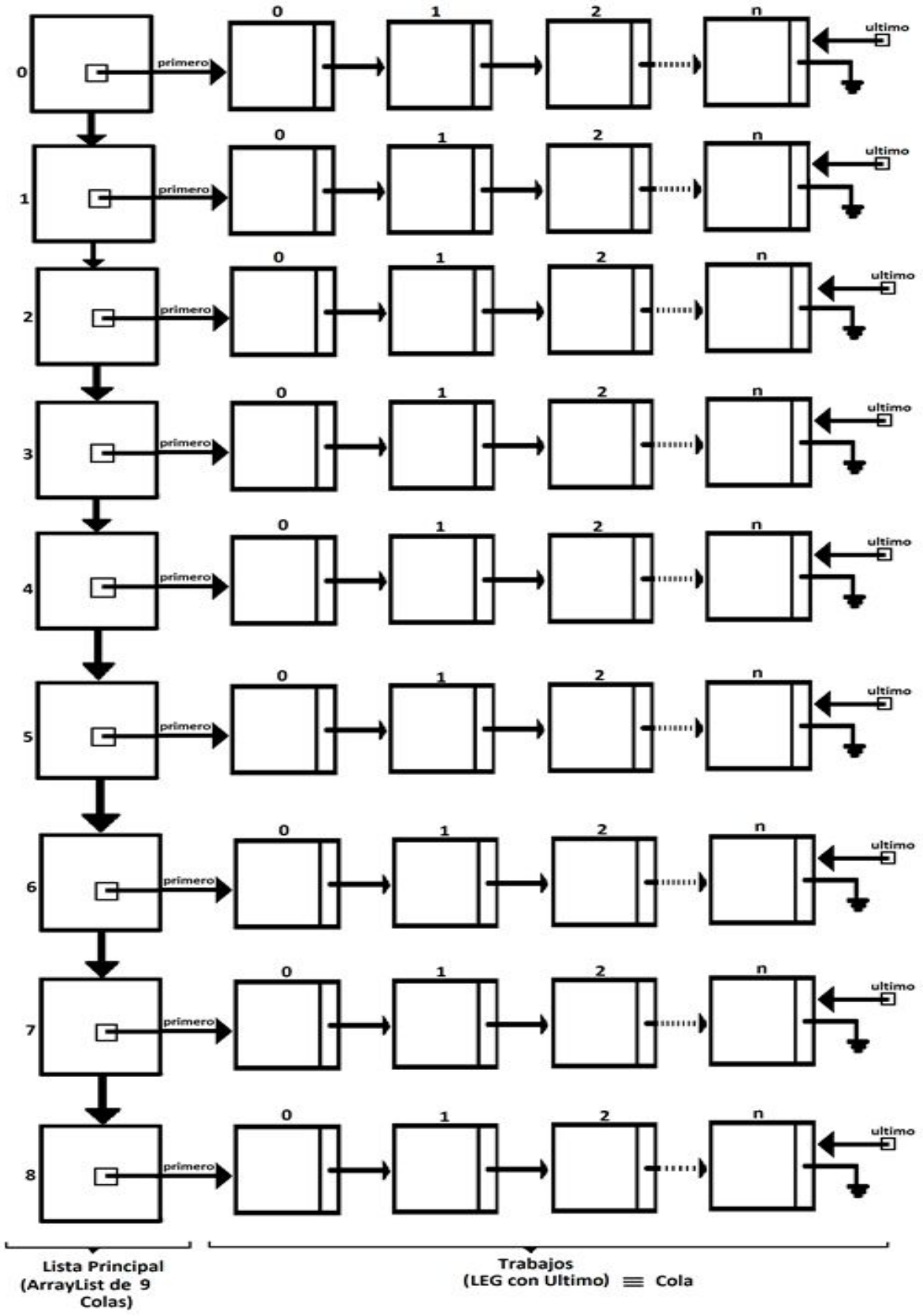
Nombre: **encolar**
Descripción: Inserta un nuevo elemento (nodo) al final de la cola.
Datos de entrada: Un objeto de tipo genérico (una instancia de tipo genérico).
Datos de Salida: Ninguno.
Precondiciones: Ninguna.
Postcondiciones: Que se haya insertado el elemento (nodo) al final de la cola.

Nombre: **desencolar**
Descripción: Saca de la cola el primer elemento (nodo) de ésta, en caso de que al menos haya un elemento en la cola, devolviendo el dato que contiene dicho nodo.
Datos de entrada: Ninguno.
Datos de Salida: El dato de tipo genérico contenido en el primer nodo de la cola (es decir, del nodo que desencolamos).
Precondiciones: Que la cola no esté vacía.
Postcondiciones: Que devuelva correctamente el dato de tipo genérico del primer nodo que se encontrara en la cola, y que además se elimine dicho nodo de la cola, pasando a ser el primer nodo de esta el siguiente al nodo que desencolamos.

Nombre: **primero**
Descripción: Devuelve el dato contenido en el nodo que se encuentra al frente de la cola.
Datos de entrada: Ninguno.
Datos de Salida: El dato del primer nodo de la cola.
Precondiciones: Que la cola no esté vacía.
Postcondiciones: Que devuelva correctamente el dato contenido en el nodo que se encuentra al frente de la cola.

Nombre: **esVacia**
Descripción: Comprueba si la cola está vacía o no.
Datos de entrada: Ninguno.
Datos de Salida: Un booleano que indica si la lista está vacía o no.
Precondiciones: Ninguna.
Postcondiciones: Devolver el booleano que indique correctamente si la lista está vacía o no.

1.2.- Diseño de la Estructura Principal



1.3.- Observaciones

1.3.1.- Estructura Principal:

La estructura principal que hemos escogido es un ArrayList de 9 Colas de Trabajos. También pensamos en haberlo hecho con un array de colas, o incluso creando una Lista Enlazada Genérica de Colas, pero como no había problema en elegir una u otra estructura, optamos por la que incluía el ArrayList.

1.3.2.- Unidades del peso (tamaño) de los trabajos:

En el enunciado de la práctica no quedaba muy claro si el peso de un trabajo debía ser almacenado y mostrado en Bytes o KiloBytes. Como no suponía ningún problema escoger una de las 2 unidades, o incluso escoger 1 unidad para almacenar y hacer la conversión en la visualización, hemos optado por escoger el KiloByte como unidad general para almacenar y mostrar los pesos.

1.3.3.- Métodos que implementan las opciones del Menú Principal:

Una de las restricciones de la práctica es que los distintos métodos que implementen las opciones del menú principal utilicen exclusivamente las operaciones básicas del TAD Cola diseñado (incluída la operación que cuenta el número de elementos de la Cola).
Nosotros, a parte de las operaciones básicas del TAD Cola diseñado, hemos utilizado métodos para asegurarnos que el usuario introduce por teclado correctamente el formato de las distintas variables (especialmente los integer y long), métodos para visualizar menús (el menú principal y el menú que surge al pedir al usuario la introducción de datos por teclado) y métodos que utilizan únicamente métodos básicos del TAD Cola (por ejemplo, el método **esVacia** del paquete OpcionesMenuPrincipal, utiliza el metodo esVacia de la clase LECola para evaluar todas las colas y determinar si la impresora esVacia completamente, o el método **buscarColaPrioritaria**, que utiliza también el método esVacia de la clase LECola para determinar cuál es la primera cola de la impresora que no está vacía, y que por tanto es la cola prioritaria. También hay un método **mostrarMasPesado**, que muestra, correctamente indexado, el mensaje que visualiza qué trabajo a la espera de ser impreso es el que más tamaño ocupa (en KiloBytes, por supuesto)). Ninguno de estos métodos utiliza otros métodos que no estén implementados en el TAD Cola.

2.- Implementación

2.1.- Código

```java
package librerias.estructurasDeDatos.lineales;

import librerias.estructurasDeDatos.modelos.Cola;

/** Clase que se encarga de la lista de nodos LEG. Representa una Lista Enlazada Generica con Ultimo, lo cual equivale a una Cola
*
* @author Adrian Herrero Artola y Juan Blanco Martin
 * @param <E> Objeto de tipo generico
 */
public class LECola<E> implements Cola<E>{

    /**
     *Puntero al primer nodo de la LEG con Ultimo (es decir, de la cola)
     */
    protected NodoLEG<E> primero;

    /**
     *Puntero al ultimo nodo de la LEG con Ultimo (es decir, de la cola)
     */
    protected NodoLEG<E> ultimo; // Referencia al primer y al ultimo nodo de la lista que representa la cola

    /**
     *Constructor que no recibe ningun parametro y que inicializa a null tanto el puntero al primero como el puntero al ultimo de la lista
     */
    public LECola(){
        this.primero = null;
        this.ultimo = null;
    }
```

```java
    /** Metodo que nos indica cual es el primer nodo de la lista
     * @return primero
     */
    public NodoLEG<E> getPrimero() {
        return primero;
    }

    /** Metodo que actualiza el valor del puntero primero
     * @param primero
     */
    public void setPrimero(NodoLEG<E> primero) {
        this.primero = primero;
    }

    /** Metodo que nos indica cual es el ultimo nodo de la lista
     * @return ultimo
     */
    public NodoLEG<E> getUltimo() {
        return ultimo;
    }

    /** Metodo que actualiza el valor del puntero ultimo
     * @param ultimo
     */
    public void setUltimo(NodoLEG<E> ultimo) {
        this.ultimo = ultimo;
    }

    /** Metodo recursivo que nos devuelve el numero de elementos que estan almacenados en la cola.
     *
     * @param actual Es el nodo que evaluaremos en la invocacion del metodo en la que nos encontremos. Distinguimos casos en funcion de si se trata del ultimo nodo o no
     * @return 0 si el nodo actual es null, 1 + lo que resulte la llamada recursiva si el nodo actual no es null
     */
    @Override
    public int contarElemCola(NodoLEG<E> actual){
```

```
        if (actual == null) { return 0; } //Hemos llegado al siguiente
elemento del que es el ultimo de la cola, es decir,al siguiente del que
se encuentra en el fin de la cola
        else { return 1 + contarElemCola(actual.siguiente); }
    }

    /** Metodo que permite insertar un nodo en el final de la lista (es
decir, al final de la cola)
     * @param x (objeto generico)
     */
    @Override
    public void encolar(E x){
        NodoLEG<E> nuevo = new NodoLEG(x);

        if(primero != null){ ultimo.siguiente = nuevo; }

        else{ primero = nuevo; }

        ultimo = nuevo;
    }

    /** Metodo que permite eliminar el primer nodo de la lista (es
decir, el nodo que se encuentra al frente de la cola)
     * @return frente Es el dato que ha desencolado
     */
    @Override
    public E desencolar(){
        E frente = primero();

        primero = primero.siguiente;

        return frente;
    }

    /** Metodo que devuelve el dato que se encuentra en el primer nodo
de la lista (es decir, el que se encuentra al frente de la cola)
     * @return retorno Es el dato contenido en el primer nodo de la
cola
     */
    @Override
```

```
    public E primero(){
        E retorno;

        if(primero == null){ retorno = null; }
        else{ retorno = primero.dato; }

        return retorno;
    }

    /** Metodo que devuelve un booleano indicando si la cola se
encuentra vacia
     * @return Si el primer dato del primer nodo de la cola es null, lo
cual indicaria que la cola esta vacia y devolveria true. En caso
contrario, la cola no estaria vacia y devolveria por tanto false
     */
    @Override
    public boolean esVacia(){
        return primero() == null;
    }

}
```

```
package librerias.estructurasDeDatos.lineales;

/**Clase que crea los nodos LEG y contiene sus metodos
*
* @author Adrian Herrero Artola y Juan Blanco Martin
 * @param <E> Objeto de tipo generico
 */
public class NodoLEG<E>{

    /**
     * Dato de tipo generico que almacena el nodo
     */
    protected E dato;

    /**
     * Puntero al siguiente nodo de la lista
     */
```

```
    protected NodoLEG<E> siguiente;

    /**
     *Constructor que no recibe ningun parametro y que inicializa a
null tanto el dato de tipo generico que almacenara como el puntero al
siguiente nodo de la lista
     */
    public NodoLEG(){
        this.dato = null;
        this.siguiente = null;
    }

    /** Constructor al que solo se le pasa el objeto que va a contener
el nodo
     * @param dato Es el dato de tipo generico que almacenara el nodo
     */
    public NodoLEG(E dato){
        this(dato, null);
    }


    /** Constructor al que se le pasa el objeto que va a contener y el
puntero siguiente
     * @param dato Es el dato de tipo generico que almacenara el nodo
     * @param siguiente Es el puntero al siguiente nodo de la lista
     */
    public NodoLEG(E dato, NodoLEG<E> siguiente){
        this.dato = dato;
        this.siguiente = siguiente;
    }

    /** Metodo que nos devuelve el objeto que almacena el nodo
     * @return dato
     */
    public E getDato() {
        return dato;
    }

    /** Metodo que actualiza el objeto que almacena el nodo
     * @param dato
```

```
     */
    public void setDato(E dato) {
        this.dato = dato;
    }

    /** Metodo que nos indica a donde apunta el puntero siguiente
     * @return siguiente
     */
    public NodoLEG getSiguiente() {
        return siguiente;
    }

    /** Metodo que actualiza el puntero siguiente
     * @param siguiente
     */
    public void setSiguiente(NodoLEG siguiente) {
        this.siguiente = siguiente;
    }
}
```

```
package librerias.estructurasDeDatos.modelos;

import librerias.estructurasDeDatos.lineales.NodoLEG;

/**Interfaz que aloja los metodos de la LECola
 *
 * @author Adrian Herrero Artola y Juan Blanco Martin
 * @param <E> Objeto de tipo generico
 */
public interface Cola<E> {
    /** Metodo recursivo que nos devuelve el numero de elementos que
estan almacenados en la cola.
     *
     * @param actual Es el nodo que evaluaremos en la invocacion del
metodo en la que nos encontremos. Distinguimos casos en funcion de si
se trata del ultimo nodo o no
     * @return 0 si el nodo actual es null, 1 + lo que resulte la
llamada recursiva si el nodo actual no es null
     */
    abstract int contarElemCola(NodoLEG<E> actual);
```

```
    /** Metodo que permite insertar un nodo en el final de la lista (es
decir, al final de la cola)
     * @param x (objeto generico)
     */
    abstract void encolar(E x);
    /** Metodo que permite eliminar el primer nodo de la lista (es
decir, el nodo que se encuentra al frente de la cola)
     * @return frente Es el dato que ha desencolado
     */
    abstract E desencolar();
    /** Metodo que devuelve el objeto que se encuentra en el primer
nodo de la lista (es decir, el que se encuentra al frente de la cola)
     * @return retorno Es el dato contenido en el primer nodo de la
cola
     */
    abstract E primero();
    /** Metodo que devuelve un booleano indicando si la cola se
encuentra vacia
     * @return Si el primer dato del primer nodo de la cola es null, lo
cual indicaria que la cola esta vacia y devolveria true. En caso
contrario, la cola no estaria vacia y devolveria por tanto false
     */
    abstract boolean esVacia();
```

```
package librerias.tiposDeDatos;

/** Clase utilizada para la creacion de objetos de tipo Trabajo
 *
* @author Adrian Herrero Artola y Juan Blanco Martin
 */
public class Trabajo{
    private final int DESPLAZAMIENTO_ID_PRIORIDAD = 100;

    /**
     * Es el id del usuario propietario del trabajo
     */
    protected int ID_Usuario;

    /**
     * Es el titulo del trabajo
```

```
     */
    protected String titulo;

    /**
     * Es el peso del trabajo
     */
    protected long peso;

    /** Constructor al que se le pasa el id de usuario, el titulo del
trabajo y el peso del mismo
     * @param ID_Usuario Es el id del usuario propietario del trabajo
     * @param titulo Es el titulo del trabajo
     * @param peso Es el peso del trabajo
     */
    public Trabajo(int ID_Usuario, String titulo, long peso){
        this.ID_Usuario = ID_Usuario;
        this.titulo = titulo;
        this.peso = peso;
    }

    /** Constructor al que se le pasa un objeto de tipo Trabajo
     * @param t Objeto de tipo Trabajo
     */
    public Trabajo(Trabajo t){
        this.ID_Usuario = t.ID_Usuario;
        this.titulo = t.titulo;
        this.peso = t.peso;
    }

    /** Metodo que indica el ID_Usuario del trabajo
     * @return ID_Usuario Es el ID_Usuario del trabajo
     */
    public int getID_Usuario() {
        return ID_Usuario;
    }

    /** Metodo que actualiza el ID_Usuario del trabajo
     * @param ID_Usuario Es el ID_Usuario del trabajo
     */
    public void setID_Usuario(int ID_Usuario) {
```

```
        this.ID_Usuario = ID_Usuario;
    }

    /** Metodo que indica el titulo del trabajo
     * @return titulo Es el titulo del trabajo
     */
    public String getTitulo() {
        return titulo;
    }

    /** Metodo que actualiza el titulo del trabajo
     * @param titulo Es el titulo del trabajo
     */
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    /** Metodo que indica el peso (tamano) del trabajo
     * @return peso Es el  peso (tamano) del trabajo
     */
    public long getPeso() {
        return peso;
    }

    /** Metodo que actualiza el peso (tamano) del trabajo
     * @param peso Es el peso (tamano) del trabajo
     */
    public void setPeso(long peso) {
        this.peso = peso;
    }

    /** Metodo que calcula la prioridad de un trabajo en funcion del
ID_Usuario que tenga
     * @return la prioridad del trabajo (entero entre 1 y 9, ambos
inclusive)
     */
    public int obtenerPrioridad(){
        return ID_Usuario/DESPLAZAMIENTO_ID_PRIORIDAD;
    }
}
```

```java
package librerias.utilidades;

import librerias.utilidades.MyInput;

/** Clase que aloja los metodos usados para validar formatos en la
entrada de datos por teclado por parte del usuario
 *
* @author Adrian Herrero Artola y Juan Blanco Martin
 */
public class Formatos {

    /** Constructor de la clase que no recibe ni actualiza ningun
parametro, sino que llama al constructor de la superclase
     *
     */
    public Formatos(){
        super();
    }

    /** Metodo que comprueba que el formato de opcion es correcto
     * @return opcion Entero con formato de entero
     */
    public static int opcionConFormatoCorrecto(){
        int opcion = -1;
        try {
            opcion = MyInput.readInt();
            System.out.println();
        }
        catch (NumberFormatException nfe) { System.out.println("\nEl
formato del numero es erroneo"); }

        return opcion;
    }

    /** Metodo que comprueba si el id de usuario introducido es un
entero comprendido en el intervalo [100-999]
     * @return id Es un entero comprendido en el intervalo [100-999]
     */
    public static int idConFormatoCorrecto(){
        int id = -1;
```

```
        try {
            id = MyInput.readInt();
            System.out.println();
            if(id < 100){ System.out.println("Error: no puede haber id
de usuarios menores a 100\n"); }
            if(999 < id){ System.out.println("Error: no puede haber id
de usuarios mayores a 999\n"); }
        }
        catch (NumberFormatException nfe) { System.out.println("\nEl
formato del numero es erroneo"); }

        return id;
    }

    /** Metodo que comprueba si el peso introducido es un elemento tipo
long mayor que 0
     * @return peso Es un tipo long mayor que 0
     */
    public static long pesoConFormatoCorrecto(){
        long peso = 0;
        try {
            peso = MyInput.readLong();
            System.out.println();
            if(peso < 0){ System.out.println("Error: no puede existir
un peso negativo\n"); }
            if(peso == 0){ System.out.println("Error: no puede existir
un peso con tamano 0\n"); }
        }
        catch (NumberFormatException nfe) { System.out.println("\nEl
formato del numero es erroneo"); }

        return peso;
    }

    /** Metodo que comprueba si se responde a una pregunta con S/N
     * @return continuar true si se responde con 'S' (o 's') o false si
se responde con 'N' (o 'n')
     */
    public static boolean continuarProcedimiento(){
        String respuesta;
```

```
        boolean continuar = false;
        boolean respuestaValida;

        do{
            respuesta = MyInput.readString();

            if((respuesta.equalsIgnoreCase("n")) ||
(respuesta.equalsIgnoreCase("N"))){
                respuestaValida = true;
                System.out.println();
            }

            else if((!respuesta.equalsIgnoreCase("s")) &&
(!respuesta.equalsIgnoreCase("S"))){
                System.out.print("Respuesta no valida. Introduce 'S' o
'N': ");
                respuestaValida = false;
            }

            else{
                continuar = true;
                respuestaValida = true;
            }

        }while(!respuestaValida);

        return continuar;
    }

    /** Metodo que comprueba si la prioridad introducida es un entero
comprendido en el intervalo [1-9]
     * @return prioridad Es un entero comprendido en el intervalo [1-9]
     */
    public static int prioridadConFormatoCorrecto(){
        int prioridad = -1;
        try {
            prioridad = MyInput.readInt();
            System.out.println();
            if(prioridad < 1){ System.out.println("Error: no existen
prioridades menores de 1\n"); }
```

```
            else if(9 < prioridad){ System.out.println("Error: no
existen prioridades mayores de 9\n"); }
        }
        catch (NumberFormatException nfe) { System.out.println("\nEl
formato del numero es erroneo"); }

        return prioridad;
    }
}
```

```
package librerias.utilidades;

import librerias.utilidades.MyInput;


/** Clase responsable de la visualizacion del menu principal
 *  y metodos utilizados en la interaccion con el menu
 *
* @author Adrian Herrero Artola y Juan Blanco Martin
 */
public class Menus {

    /** Constructor de la clase que no recibe ni actualiza ningun
parametro, sino que llama al constructor de la superclase
     *
     */
    public Menus(){
        super();
    }

    /** Metodo con la visualizacion del menu principal
     * @param opc -1 por defecto
     * @return El entero con la opcion elegida por el usuario
     */
    public static int menuPrincipal(int opc) {
        System.out.println("      MENU PRINCIPAL");
        System.out.println("1.- Enviar un trabajo a la impresora");
        System.out.println("2.- Imprimir trabajos");
        System.out.println("3.- Mostrar trabajo mas pesado");
```

```
        System.out.println("4.- Mostrar tiempo de espera de un
usuario");
        System.out.println("5.- Informe de trabajos por prioridad");
        System.out.println("6.- Informe de trabajos de una prioridad");
        System.out.println("7.- Reducir espera en una prioridad");
        System.out.println("8.- Reiniciar el Sistema de impresion");
        System.out.println("0.- Salir");
        System.out.print("Elija opcion: ");

        opc = Formatos.opcionConFormatoCorrecto();

        return opc;
    }

    /** Metodo para introducir correctamente el id de Usuario, segun
los requisitos pedidos en el enunciado (que sea un entero entre
[100-999])
     * @return id Es el ID_Usuario
     */
    public static int introducirIdUsuario(){
        int id = -1;
        while((id < 100) || (999 < id)){
            System.out.print("  Introduce el ID de Usuario: ");
            id = Formatos.idConFormatoCorrecto();
        }

        return id;
    }

    /** Metodo para introducir titulo del trabajo
     * @return MyInput.readString() Es la cadena que representa el
titulo del trabajo
     */
    public static String introducirTituloTrabajo(){
        System.out.print("  Introduce el titulo del trabajo a imprimir:
");
        return MyInput.readString();
    }

    /** Metodo para introducir peso del trabajo
```

```
     * @return peso Es una variable tipo long que representa el peso
(tamano) del trabajo
     */
    public static long introducirPesoTrabajo(){
        long peso = -1;

        while(peso <= 0){
            System.out.print("  Introduce el peso del trabajo a
imprimir (en Kb): ");
            peso = Formatos.pesoConFormatoCorrecto();
        }

        return peso;
    }

    /** Metodo para introducir una prioridad, segun los requisitos
pedidos en el enunciado (que sea un entero entre [1-9])
     * @return prioridad Es un entero en el intervalo [1-9]
     */
    public static int introducirPrioridad(){
        int prioridad = -1;
        while((prioridad < 1) || (9 < prioridad)){
            System.out.print("  Introduce una prioridad: ");
            prioridad = Formatos.prioridadConFormatoCorrecto();
        }

        return prioridad;
    }
}
```

```
package librerias.utilidades;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
```

```java
/** La clase MyInput permite la entrada de datos en el programa
*
* @author Adrian Herrero Artola y Juan Blanco Martin
*/
public class MyInput {

    /**
     * Lee un dato de tipo string desde el teclado
     * @return string
     */
    public static String readString() {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in),1);
        String string="";
        try {
        string = br.readLine(); }
        catch (IOException ex) {
        System.out.println(ex); }
        return string; }

    /**
     * Lee un dato tipo int desde el teclado
     * @return
     * @throws NumberFormatException
     */
    public static int readInt()throws NumberFormatException{
        return Integer.parseInt(readString()); }

    /**
     * Lee un dato tipo double desde el teclado
     * @return
     */
    public static double readDouble() {
        return Double.parseDouble(readString()); }

    /**
     * Lee un dato tipo byte desde el teclado
     * @return byte
     */
```

```
    public static byte readByte() {
        return Byte.parseByte(readString()); }

    /**
     * Lee un dato tipo short desde el teclado
     * @return short
     */
    public static short readShort() {
        return Short.parseShort(readString()); }

    /**
     * Lee un dato tipo long desde el teclado
     * @return
     */
    public static long readLong() {
        return Long.parseLong(readString()); }

    /**
     * Lee un dato tipo float desde el teclado
     * @return
     */
    public static float readFloat() {
        return Float.parseFloat(readString()); }

    /**
     * Permite la lectura de ficheros
     * @param nombreFichero
     */
    public static void leeFichero(String nombreFichero){
        File fichero=null;
        FileReader fr=null;
        BufferedReader br=null;

        try{
            fichero=new File(nombreFichero);
            fr = new FileReader(fichero);
            br = new BufferedReader(fr);

            String linea;
```

```
            while ((linea = br.readLine())!= null){
                System.out.println(linea);
            }
        }
        catch (Exception e){ e.printStackTrace(); }

        finally {
            try {
                if (null != fr){
                    fr.close();
                    br.close();
                }
            }
            catch (Exception e1){ e1.printStackTrace(); }
        }
    }

    /**
     * Permite la escritura de ficheros
     * @param nombreFichero
     * @param cadena
     */
    public static void escribeFichero(String nombreFichero, String
cadena){
        BufferedWriter bw = null;
        FileWriter fw = null;

        try{
            File file = new File(nombreFichero);

            //Si el archivo no existe, se crea
            if(!file.exists()){
                file.createNewFile();
            }

            //flag true, indica adjuntar informacion al archivo
            fw = new FileWriter(file.getAbsoluteFile(), true);
            bw = new BufferedWriter(fw);
            bw.write("\n"+cadena);
```

```
            System.out.println("Elemento agregado al archivo " +
nombreFichero);
        }
        catch (IOException e){ e.printStackTrace(); }
        finally{
            try{
                //Cierra las instancias de FileWriter y BufferedWriter
                if(bw != null){
                    bw.close();
                }
                if(fw != null){
                    fw.close();
                }
            }
            catch (IOException ex){ ex.printStackTrace(); }
        }
    }
}
```

```
package programa;

import librerias.utilidades.Formatos;
import static librerias.utilidades.Menus.introducirIdUsuario;
import static librerias.utilidades.Menus.introducirPesoTrabajo;
import static librerias.utilidades.Menus.introducirPrioridad;
import static librerias.utilidades.Menus.introducirTituloTrabajo;
import java.util.ArrayList;
import librerias.estructurasDeDatos.lineales.LECola;
import librerias.estructurasDeDatos.lineales.NodoLEG;
import librerias.tiposDeDatos.Trabajo;

/** Clase que engloba los metodos con los que se ejecutaran las
acciones de las diferentes opciones del menu principal
 *
 * @author Adrian Herrero Artola y Juan Blanco Martin
 */
public class OpcionesMenuPrincipal {

    /** Constructor de la clase que no recibe ni inicializa ningun
parametro, sino que llama al constructor de la superclase
```

```
     *
     */
    public OpcionesMenuPrincipal(){
        super();
    }

    /** Metodo que solicita al usuario la introduccion por teclado de
un ID de usuario (valor entero entre 100-999), el titulo y el peso del
trabajo a imprimir, y envia un trabajo con los datos introducidos a la
impresora
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void enviarTrabajo(ArrayList<LECola<Trabajo>>
impresora){
        //Introduccion de datos del usuario
        int id = introducirIdUsuario();
        String titulo = introducirTituloTrabajo();
System.out.println();
        long peso = introducirPesoTrabajo();
System.out.println();

        Trabajo nuevo = new Trabajo(id, titulo, peso);

        //Calculo de prioridad
        int prioridad = nuevo.obtenerPrioridad(); //Obtienes un entero
del 1 al 9

        LECola<Trabajo> aux = impresora.get((prioridad-1));

        aux.encolar(nuevo);

        //el arrayList tiene 9 posiciones (0-8)
        impresora.set((prioridad-1), aux);

        System.out.println("Informacion:  El trabajo se ha enviado
correctamente a la impresora.\n\n");
    }

    /** Metodo utilizado para imprimir trabajos
```

```
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void imprimirTrabajos(ArrayList<LECola<Trabajo>>
impresora){
        if(!esVacia(impresora)){
            boolean seguirImprimiendo;
            Trabajo impreso;

            int cola = buscarColaPrioritaria(impresora);//Busca la
siguiente cola que contenga trabajos para ser impresos

            do{
                impreso = impresora.get(cola).desencolar();

                System.out.println("Se ha impreso el trabajo
'"+impreso.getTitulo()+"' (Usuario "+impreso.getID_Usuario()+",
Prioridad "+impreso.obtenerPrioridad()+")\n");

                if(!esVacia(impresora)){//Al menos existe 1 cola que
contiene, al menos, 1 trabajo para ser impreso
                    System.out.print("Desea imprimir otro trabajo?
(S/N): ");
                    seguirImprimiendo =
Formatos.continuarProcedimiento();   System.out.println();

                    if(seguirImprimiendo){//Buscamos el siguiente
trabajo a ser impreso
                        if(impresora.get(cola).esVacia()){//Pasamos a
la siguiente cola que contenga trabajos a imprimir
                            cola = buscarColaPrioritaria(impresora);
                        }
                    }
                }
                else{
                    seguirImprimiendo = false;
                    System.out.println("\nNo quedan mas trabajos a la
espera de impresion.\n\n");
                }
            }while(seguirImprimiendo);
```

```
        }
        else{ System.out.println("\nERROR: No hay trabajos a la espera
de impresion.\n       El Sistema se encuentra en estado inicial.\n\n");
}
    }

    /** Metodo que muestra en pantalla un mensaje indicando cual es el
trabajo que tiene mayor tamano (en kilobytes) que se encuentra a la
espera de ser impreso
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void trabajoMasPesado(ArrayList<LECola<Trabajo>>
impresora){
        if(!esVacia(impresora)){
            Trabajo masPesado = new Trabajo(1000, "Predeterminado", 0);
// Fuerzo a que el trabajo predeterminado tenga una prioridad de 10 (al
dividir entre 100 con el metodo obtenerPrioridad) la cual es menor que
cualquier prioridad a la que se vaya a comparar, y tambien le fuerzo a
que tenga un peso 0, mayor que cualquier peso con el que se vaya a
comparar. (Este trabajo nunca se mostrara puesto que la lista no esta
vacia)
            System.out.println("\n\t\t\t\tTRABAJO DE IMPRESION MAS
PESADO\n");
            System.out.println("ID_Usuario\tNivel de
Prioridad\tTitulo\t\t\t\t\tTamano (Kb)");

System.out.println("_____________________________________________________
_____________________________________________");
            for(int cola = 0; cola < impresora.size(); cola++){
                for(NodoLEG<Trabajo> aux =
impresora.get(cola).getPrimero(); aux != null; aux =
aux.getSiguiente()){
                    if(masPesado.getPeso() < aux.getDato().getPeso()){
                        masPesado = new
Trabajo(aux.getDato().getID_Usuario(), aux.getDato().getTitulo(),
aux.getDato().getPeso());
                    }
```

```
                    else if((masPesado.getPeso() ==
aux.getDato().getPeso()) && (masPesado.obtenerPrioridad() >
aux.getDato().obtenerPrioridad())){
                        masPesado = new
Trabajo(aux.getDato().getID_Usuario(), aux.getDato().getTitulo(),
aux.getDato().getPeso());
                    }
                    else if((masPesado.getPeso() ==
aux.getDato().getPeso()) && (masPesado.obtenerPrioridad() ==
aux.getDato().obtenerPrioridad())){
                        masPesado = new
Trabajo(aux.getDato().getID_Usuario(), aux.getDato().getTitulo(),
aux.getDato().getPeso());
                    }
                }
            }
            mostrarMasPesado(masPesado);
        }
        else{ System.out.println("\nERROR: No hay trabajos a la espera
de impresion.\n       El Sistema se encuentra en estado inicial.\n\n");
}
    }

    /** Metodo que solicita un ID_Usuario y muestra en pantalla cuantos
trabajos seran impresos ANTES de que se imprima el primer trabajo que
envio este usuario, incluyendo los trabajos que tienen una prioridad
mayor
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void informarEspera(ArrayList<LECola<Trabajo>>
impresora){
        if(!esVacia(impresora)){
            boolean encontrado = false;
            int cola = 0;
            int trabajosDelante = 0;
            NodoLEG<Trabajo> aux;
            int id = introducirIdUsuario();

            while((!encontrado) && (cola < impresora.size())){
```

```
                if(!impresora.get(cola).esVacia()){
                    aux = impresora.get(cola).getPrimero();
                    while((!encontrado) && (aux != null)){
                        if(id == aux.getDato().getID_Usuario()){
encontrado = true; }
                        else{ trabajosDelante++; }
                        aux = aux.getSiguiente();
                    }
                }
                if(!encontrado){ cola++; }
            }

            if(encontrado){
                if(0 < trabajosDelante){
                    System.out.print("**El usuario "+id+" tiene delante
"+trabajosDelante+" trabajo");
                    if(trabajosDelante == 1){
System.out.println(".\n\n"); }
                    else{ System.out.println("s.\n\n"); }
                }
                else{ System.out.println("**El usuario "+id+" no tiene
trabajos delante.\n  Su primer trabajo es el prioritario para ser
impreso.\n\n"); }
            }
            else{ System.out.println("**No existen trabajos pendientes
del usuario "+id+".\n\n"); }
        }
        else{ System.out.println("\nERROR: No hay trabajos a la espera
de impresion.\n       El Sistema se encuentra en estado inicial.\n\n");
}
    }

    /** Metodo que muestra en pantalla un listado indicando cuantos
trabajos se encuentran a la espera de impresion en cada una de las
distintas prioridades existentes
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void
informeTodasPrioridades(ArrayList<LECola<Trabajo>> impresora){
```

```
        if(!esVacia(impresora)){
            int totalPendientes = 0;
            int totalCola;
            System.out.println("\n\tINFORME DE TRABAJOS POR
PRIORIDAD\n");
            System.out.println("Nivel de Prioridad\t\tTrabajos en
espera");

System.out.println("_____________________________________________________
");
            for(int cola = 0; cola < impresora.size(); cola++){
                totalCola =
impresora.get(cola).contarElemCola(impresora.get(cola).getPrimero());
                System.out.println("
"+(cola+1)+"\t\t\t\t"+totalCola);
                totalPendientes += totalCola;
            }

System.out.println("-----------------------------------------------------
");
            System.out.println("   Numero total de trabajos pendientes:
"+ totalPendientes);

System.out.println("-----------------------------------------------------
");
            System.out.println("\n");
        }
        else{ System.out.println("\nERROR: No hay trabajos a la espera
de impresion.\n       El Sistema se encuentra en estado inicial.\n\n");
}
    }

    /** Metodo que solicita al usuario una prioridad (1..9) y muestra
en pantalla un listado indicando los trabajos con esa prioridad que se
encuentran a la espera de impresion
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void informeUnaPrioridad(ArrayList<LECola<Trabajo>>
impresora){
```

```
        int prioridad = introducirPrioridad();

        if(!impresora.get(prioridad-1).esVacia()){
            System.out.println("\n\tINFORME DE TRABAJOS con Prioridad
"+prioridad+"\n");
            System.out.println("ID Usuario\t\tTitulo");

System.out.println("_____________________________________________________
___________");
            for(NodoLEG<Trabajo> aux =
impresora.get(prioridad-1).getPrimero(); aux != null; aux =
aux.getSiguiente()){
                    System.out.println("
"+aux.getDato().getID_Usuario()+"\t\t\t"+aux.getDato().getTitulo());
            }
            System.out.println("\n");
        }
        else{ System.out.println("\nERROR: No hay trabajos con esa
prioridad a la espera de impresion.\n\n"); }
    }

    /** Metodo que solicita una prioridad (1..9) y elimina algunos de
los trabajos que actualmente se encuentran en espera con esa prioridad,
mostrando un mensaje en que indica que trabajo se ha eliminado de la
cola, con cada trabajo eliminado de la misma
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void reducirEspera(ArrayList<LECola<Trabajo>>
impresora){
        int prioridad = introducirPrioridad();

        if(!impresora.get(prioridad-1).esVacia()){
            int distancia = 2;
            int cont = 1;
            int numElems =
impresora.get(prioridad-1).contarElemCola(impresora.get(prioridad-1).ge
tPrimero());

            Trabajo t;
```

```
            LECola<Trabajo> colaAux; // Creamos una cola auxiliar

            if(numElems == 1){
                System.out.println("No se ha reducido la espera en la
prioridad "+prioridad+", ya que solo hay 1 trabajo en esta cola.");
            }

            else{
                while(numElems >= distancia){
                    colaAux = new LECola();
                    for(int i = 0; i < numElems; i++){
                        // Desencolamos un trabajo
                        t = impresora.get(prioridad-1).desencolar();

                        if((cont%distancia) == 1){ // Si el trabajo es
victima, se imprime
                            System.out.println("Se ha eliminado el
trabajo <"+t.getID_Usuario()+"><"+t.getTitulo()+"><"+t.getPeso()+"> de
la cola");
                        }

                        else{ // En caso contrario, se encola en la
cola auxiliar
                            colaAux.encolar(t);
                        }

                        cont++;
                    }
                    cont = 1;
                    distancia++;
                    impresora.set(prioridad-1, colaAux); //
Reemplazamos el puntero al primero de la cola vigente (en este momento
vacia), por el puntero al primer elemento de la cola auxiliar, que es
la que contiene los trabajos que no han sido impresos
                    numElems =
impresora.get(prioridad-1).contarElemCola(impresora.get(prioridad-1).ge
tPrimero()); // Calculamos el numero de elementos de la cola, ahora
actualizada
                }
            }
```

```
        }

        else{ System.out.println("\nERROR: No hay trabajos con esa
prioridad a la espera de impresion.\n"); }

        System.out.println("\n");
    }

    /** Metodo utilizado para reiniciar la impresora. Las ordenes de
impresion de todos los trabajos que se encuentran a la espera de ser
impresos se perderan, pero antes, el sistema de gestion de impresion
muestra un mensaje en pantalla indicando que usuarios/trabajos estan
afectados
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     */
    public static void reiniciarSistema(ArrayList<LECola<Trabajo>>
impresora){
        if(!esVacia(impresora)){
            System.out.println("\n\tLISTADO DE TRABAJOS DE IMPRESION
ABORTADOS\n");
            System.out.println("ID Usuario\t\tTitulo");

System.out.println("_____________________________________________________
__________");
            for(int cola = 0; cola < impresora.size(); cola++){
                for(NodoLEG<Trabajo> aux =
impresora.get(cola).getPrimero(); aux != null; aux =
aux.getSiguiente()){
                    aux.setDato(impresora.get(cola).desencolar());
                    System.out.println("
"+aux.getDato().getID_Usuario()+"\t\t\t"+aux.getDato().getTitulo());
                }
            }
            System.out.println("\n");
        }
        else{ System.out.println("\nERROR: No hay trabajos a la espera
de impresion.\n       El Sistema se encuentra en estado inicial.\n\n");
}
    }
```

```
    /** Metodo utilizado para verificar si las 9 colas que forman la
impresora estan todas vacias, o no
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     * @return Si el numero de colas vacias coincide con el tamano de
la impresora (9, ya que son 9 colas). En caso de coincidir, significa
que la impresora esta vacia (sus 9 colas estan vacias) y devolvera
true. Devolvera false en caso contrario
     */
    public static boolean esVacia(ArrayList<LECola<Trabajo>>
impresora){
        int numColasVacias = 0;
        for(int i = 0; i < impresora.size(); i++){
            if(impresora.get(i).esVacia()){ numColasVacias++; }
        }

        return numColasVacias == impresora.size();
    }

    /** Metodo que busca la cola con mayor prioridad de la impresora
que cuente con trabajos a la espera de impresion
     *
     * @param impresora Es el ArrayList de 9 colas de trabajos
     * @return El indice de la cola con mayor prioridad de la impresora
que cuente con trabajos a la espera de impresion
     */
    public static int buscarColaPrioritaria(ArrayList<LECola<Trabajo>>
impresora){
        int prioritaria = 0;

        if(!esVacia(impresora)){
while(impresora.get(prioritaria).esVacia()){ prioritaria++; } }
//Hallamos la primera cola que tenga trabajos para imprimir

        return prioritaria;
    }

    /** Metodo utilizado para mostrar por pantalla correctamente
indexado el trabajo mas pesado a la espera de ser impreso
```

```
     *
     * @param trabajo Es el trabajo del cual se mosrtaran sus datos por
pantalla
     */
    public static void mostrarMasPesado(Trabajo trabajo){
        if(trabajo.getTitulo().length() < 8){ System.out.println("
"+trabajo.getID_Usuario()+"\t\t\t"+trabajo.obtenerPrioridad()+"\t\t"+tr
abajo.getTitulo()+"\t\t\t\t\t\t"+trabajo.getPeso()+"\n\n"); }
        else if(trabajo.getTitulo().length() < 16){
System.out.println("
"+trabajo.getID_Usuario()+"\t\t\t"+trabajo.obtenerPrioridad()+"\t\t"+tr
abajo.getTitulo()+"\t\t\t\t\t"+trabajo.getPeso()+"\n\n"); }
        else if(trabajo.getTitulo().length() < 24){
System.out.println("
"+trabajo.getID_Usuario()+"\t\t\t"+trabajo.obtenerPrioridad()+"\t\t"+tr
abajo.getTitulo()+"\t\t\t\t"+trabajo.getPeso()+"\n\n"); }
        else if(trabajo.getTitulo().length() < 32){
System.out.println("
"+trabajo.getID_Usuario()+"\t\t\t"+trabajo.obtenerPrioridad()+"\t\t"+tr
abajo.getTitulo()+"\t\t\t"+trabajo.getPeso()+"\n\n"); }
        else{ System.out.println("
"+trabajo.getID_Usuario()+"\t\t\t"+trabajo.obtenerPrioridad()+"\t\t"+tr
abajo.getTitulo()+"\t\t"+trabajo.getPeso()+"\n\n"); }
    }
}
```

```
package programa;

import librerias.utilidades.Menus;
import java.util.ArrayList;
import librerias.estructurasDeDatos.lineales.LECola;
import librerias.tiposDeDatos.Trabajo;
import static programa.OpcionesMenuPrincipal.*;

/**
 *Clase principal. El metodo que arranca el programa se encuentra en
esta clase
 * @author Adrian Herrero Artola y Juan Blanco Martin
 */
public class Principal {
```

```
    private static final int TAM_IMPRESORA = 9;
    /**Metodo principal que inicia la ejecucion del programa
     * @param args the command line arguments
     */
    public static void main(String[] args){
        menuPrincipal();
    }


    /**Metodo que invoca los metodos utilizados en cada opcion del menu principal, en funcion de la opcion que introduzca el usuario por teclado
     */
    public static void menuPrincipal(){
        int opc = -1;

        ArrayList<LECola<Trabajo>> impresora = new ArrayList();

        iniciarImpresora(impresora); // Introducimos las 9 colas en la impresora

        do {
            opc = Menus.menuPrincipal(opc);

            switch(opc){
                case 0:{ System.out.println("\n\n\n*** Gracias por utilizar la aplicación PrintManagement ***\n\n\n"); } break;

                case 1:{//Enviar un trabajo a la impresora
                    enviarTrabajo(impresora);
                } break;

                case 2:{//Imprimir trabajos
                    imprimirTrabajos(impresora);
                } break;

                case 3:{//Mostrar trabajo mas pesado
                    trabajoMasPesado(impresora);
                } break;
```

```
                case 4:{//Mostrar tiempo de espera de un usuario
                    informarEspera(impresora);
                } break;

                case 5:{//Informe de trabajos por prioridad
                    informeTodasPrioridades(impresora);
                } break;

                case 6:{//Informe de trabajos de una prioridad
                    informeUnaPrioridad(impresora);
                } break;

                case 7:{//Reducir espera en una prioridad
                    reducirEspera(impresora);
                } break;

                case 8:{//Reiniciar el Sistema de impresion
                    reiniciarSistema(impresora);
                } break;

                default:{ System.out.println("Error: Introduce un
numero que corresponda a alguna de las posibles opciones del Menu
Principal\n"); }
            }
        }while(opc !=0);
    }

    /**Metodo que incluye las 9 colas de trabajos en la impresora
     * @param impresora Es el ArrayList de colas de trabajos
     */
    public static void iniciarImpresora(ArrayList<LECola<Trabajo>>
impresora){
        for(int i = 0; i < TAM_IMPRESORA; i++) impresora.add(i, new
LECola());
    }
}
```

2.2.- Pruebas de ejecución

Aspecto del menú principal tras arrancar el programa:

```
      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 0
```

Introducir una opción con formato incorrecto:

```
0.- Salir
Elija opcion: h

El formato del numero es erroneo
Error: Introduce un numero que corresponda a alguna de las posibles opciones del Menu Principal

      MENU PRINCIPAL
```

Introducir una opción que sea un entero fuera del rango [0-8]:

```
0.- Salir
Elija opcion: -1

Error: Introduce un numero que corresponda a alguna de las posibles opciones del Menu Principal

      MENU PRINCIPAL

0.- Salir
Elija opcion: 9

Error: Introduce un numero que corresponda a alguna de las posibles opciones del Menu Principal

      MENU PRINCIPAL
```

Intentar ejecutar cualquiera de las opciones 2 a 8 sin que haya ningún trabajo a la espera de impresión:

```
0.- Salir
Elija opcion: 2


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


       MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 3


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


       MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 4


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


       MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

```
0.- Salir
Elija opcion: 5


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


      MENU PRINCIPAL

 8.- Reiniciar el Sistema de impresion
 0.- Salir
 Elija opcion: 6

   Introduce una prioridad: 1


 ERROR: No hay trabajos con esa prioridad a la espera de impresion.


       MENU PRINCIPAL

0.- Salir
Elija opcion: 7

  Introduce una prioridad: 4


ERROR: No hay trabajos con esa prioridad a la espera de impresion.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 8


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


      MENU PRINCIPAL
```

Salir del programa:

```
0.- Salir
Elija opcion: 0



*** Gracias por utilizar la aplicación PrintManagement ***


BUILD SUCCESSFUL (total time: 24 minutes 12 seconds)
```

Opción 1:

```
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 99

Error: no puede haber id de usuarios menores a 100

  Introduce el ID de Usuario: 1000

Error: no puede haber id de usuarios mayores a 999

  Introduce el ID de Usuario: m

El formato del numero es erroneo
  Introduce el ID de Usuario: 285

  Introduce el titulo del trabajo a imprimir: Diagramas UML

  Introduce el peso del trabajo a imprimir (en Kb): 0

Error: no puede existir un peso con tamano 0

  Introduce el peso del trabajo a imprimir (en Kb): -1

Error: no puede existir un peso negativo

  Introduce el peso del trabajo a imprimir (en Kb): 40500


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
```

Opción 2 (tras haber introducido varios trabajos con la opción 1):

```
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 183

  Introduce el titulo del trabajo a imprimir: Enunciado de la practica 1 de PED

  Introduce el peso del trabajo a imprimir (en Kb): 20000


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 187

  Introduce el titulo del trabajo a imprimir: Enunciado de la practica 3 de MP

  Introduce el peso del trabajo a imprimir (en Kb): 30000


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

```
0.- Salir
Elija opcion: 2

Se ha impreso el trabajo 'Enunciado de la practica 1 de PED' (Usuario 183, Prioridad 1)

Desea imprimir otro trabajo? (S/N): b
Respuesta no valida. Introduce 'S' o 'N': 4
Respuesta no valida. Introduce 'S' o 'N': N


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 2

Se ha impreso el trabajo 'Enunciado de la practica 3 de MP' (Usuario 187, Prioridad 1)

Desea imprimir otro trabajo? (S/N): S

Se ha impreso el trabajo 'Diagramas UML' (Usuario 285, Prioridad 2)


No quedan mas trabajos a la espera de impresion.

      MENU PRINCIPAL
0.- Salir
Elija opcion: 2


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


      MENU PRINCIPAL
```

Opción 3:
(Para comprobar todos los posibles casos que se pueden dar con esta opción, introducimos varios trabajos con la opción 1 del menú principal)

```
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 789

  Introduce el titulo del trabajo a imprimir: Libro Codigo Limpio

  Introduce el peso del trabajo a imprimir (en Kb): 79207


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 285

  Introduce el titulo del trabajo a imprimir: Diagramas UML

  Introduce el peso del trabajo a imprimir (en Kb): 15000


Informacion:  El trabajo se ha enviado correctamente a la impresora.
```

```
Elija opcion: 1

  Introduce el ID de Usuario: 183

  Introduce el titulo del trabajo a imprimir: Enunciado de la practica 1

  Introduce el peso del trabajo a imprimir (en Kb): 70000


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 376

  Introduce el titulo del trabajo a imprimir: Diseno de Bases de Datos Re

  Introduce el peso del trabajo a imprimir (en Kb): 12000


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
```

```
Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 187

  Introduce el titulo del trabajo a imprimir: Enunciado de la practica 3

  Introduce el peso del trabajo a imprimir (en Kb): 20000


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
```

```
Elija opcion: 1

  Introduce el ID de Usuario: 201

  Introduce el titulo del trabajo a imprimir: Practica Final DIU

  Introduce el peso del trabajo a imprimir (en Kb): 50400


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 520

  Introduce el titulo del trabajo a imprimir: Practica 1 FTI

  Introduce el peso del trabajo a imprimir (en Kb): 50400


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
0.- Salir
Elija opcion: 1

  Introduce el ID de Usuario: 520

  Introduce el titulo del trabajo a imprimir: Practica 2 FTI

  Introduce el peso del trabajo a imprimir (en Kb): 50400


Informacion:  El trabajo se ha enviado correctamente a la impresora.


      MENU PRINCIPAL
```

```
0.- Salir
Elija opcion: 3

                                TRABAJO DE IMPRESION MAS PESADO

ID_Usuario      Nivel de Prioridad      Titulo                                  Tamano (Kb)
-------------------------------------------------------------------------------------------
   789                  7              Libro Codigo Limpio                     79207

      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

Ejecutamos la opción 8:

```
0.- Salir
Elija opcion: 8

        LISTADO DE TRABAJOS DE IMPRESION ABORTADOS

ID Usuario              Titulo
-------------------------------------------------------------------
   183                  Enunciado de la practica 1 de PED
   187                  Enunciado de la practica 3 de MP
   285                  Diagramas UML
   201                  Practica Final DIU
   376                  Diseno de Bases de Datos Relacionales
   520                  Practica 1 FTI
   520                  Practica 2 FTI
   789                  Libro Codigo Limpio


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 8


ERROR: No hay trabajos a la espera de impresion.
       El Sistema se encuentra en estado inicial.


      MENU PRINCIPAL
```

Ahora, añadimos de nuevo todos los trabajos menos el que fue el más pesado en la anterior ejecución (Libro Codigo Limpio, del usuario 789 (prioridad 7), con 79207 KB). Son las mismas capturas de antes, quitando la de este trabajo. Ahora, el trabajo más pesado sería:

```
0.- Salir
Elija opcion: 3

                        TRABAJO DE IMPRESION MAS PESADO

ID_Usuario      Nivel de Prioridad      Titulo                                  Tamano (Kb)
-------------------------------------------------------------------------------------------
   183                  1              Enunciado de la practica 1 de PED       70000

      MENU PRINCIPAL
```

Desencolamos precisamente este trabajo, puesto que además es el más prioritario en espera de ser impreso de toda la impresora:

```
0.- Salir
Elija opcion: 2

Se ha impreso el trabajo 'Enunciado de la practica 1 de PED' (Usuario 183, Prioridad 1)

Desea imprimir otro trabajo? (S/N): n

      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 3

                        TRABAJO DE IMPRESION MAS PESADO

ID_Usuario      Nivel de Prioridad      Titulo                                  Tamano (Kb)
-------------------------------------------------------------------------------------------
   201                  2              Practica Final DIU                      50400

      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

Había 3 trabajos con el mismo peso (50400):
-Practica Final DIU, con prioridad 2
-Practica 1 FTI, con prioridad 5
-Practica 2 FTI, con prioridad 5 (de estos 2 últimos, la Practica 1 fue encolada antes que la 2)
Desencolamos trabajos hasta desencolar la Practica Final de DIU, y ejecutamos de nuevo la opción 3:

```
0.- Salir
Elija opcion: 2

Se ha impreso el trabajo 'Enunciado de la practica 3 de MP' (Usuario 107, Prioridad 1)

Desea imprimir otro trabajo? (S/N): s

Se ha impreso el trabajo 'Diagramas UML' (Usuario 285, Prioridad 2)

Desea imprimir otro trabajo? (S/N): s

Se ha impreso el trabajo 'Practica Final DIU' (Usuario 201, Prioridad 2)

Desea imprimir otro trabajo? (S/N): n


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 3

                              TRABAJO DE IMPRESION MAS PESADO

ID_Usuario      Nivel de Prioridad      Titulo                                                  Tamano (Kb)
-------------------------------------------------------------------------------------------------------
   520                  5              Practica 2 FTI                                          50400

      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

Volvemos a meter los trabajos anteriores en el mismo orden y de la misma forma que antes (mismas capturas) para seguir ejecutando el resto de opciones del menú principal.

Opción 4:

```
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 4

  Introduce el ID de Usuario: 999

**No existen trabajos pendientes del usuario 999.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 4

  Introduce el ID de Usuario: 789

**El usuario 789 tiene delante 7 trabajos.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

```
Elija opcion: 4

  Introduce el ID de Usuario: 520

**El usuario 520 tiene delante 5 trabajos.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 4

  Introduce el ID de Usuario: 187

**El usuario 187 tiene delante 1 trabajo.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 4

  Introduce el ID de Usuario: 183

**El usuario 183 no tiene trabajos delante.
  Su primer trabajo es el prioritario para ser impreso.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

Opción 5:

```
0.- Salir
Elija opcion: 5


        INFORME DE TRABAJOS POR PRIORIDAD

Nivel de Prioridad              Trabajos en espera
_____________________________________________________
        1                                 2
        2                                 2
        3                                 1
        4                                 0
        5                                 2
        6                                 0
        7                                 1
        8                                 0
        9                                 0
-----------------------------------------------------
   Numero total de trabajos pendientes: 8
-----------------------------------------------------


      MENU PRINCIPAL
1 - Enviar un trabajo a la impresora
```

Opción 6:
Por ejemplo:

```
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 1


        INFORME DE TRABAJOS con Prioridad 1

ID Usuario              Titulo
_____________________________________________________________
   183                  Enunciado de la practica 1 de PED
   187                  Enunciado de la practica 3 de MP


     MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 2


        INFORME DE TRABAJOS con Prioridad 2

ID Usuario              Titulo
_____________________________________________________________
   285                  Diagramas UML
   201                  Practica Final DIU


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
```

```
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 3

        INFORME DE TRABAJOS con Prioridad 3

ID Usuario              Titulo
-------------------------------------------------------------------------
   376                  Diseno de Bases de Datos Relacionales

      MENU PRINCIPAL
0.- Salir
Elija opcion: 6

  Introduce una prioridad: k

El formato del numero es erroneo
  Introduce una prioridad: 0

Error: no existen prioridades menores de 1

  Introduce una prioridad: 10

Error: no existen prioridades mayores de 9

  Introduce una prioridad: 9


ERROR: No hay trabajos con esa prioridad a la espera de impresion.

      MENU PRINCIPAL
```

Opción 7:

Con los trabajos que teníamos, por ejemplo:

```
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 5

Se ha eliminado el trabajo <520><Practica 1 FTI><50400> de la cola


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

Si ejecutamos la opción 6 para esta cola:

```
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 5


        INFORME DE TRABAJOS con Prioridad 5

ID Usuario              Titulo
_____________________________________________________________
   520                  Practica 2 FTI


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
```

A continuación, reiniciamos el sistema y metemos trabajos prueba con nombres más visuales para comprobar esta opción 7 del menú:

0.- Salir
Elija opcion: 6

Introduce una prioridad: 1

INFORME DE TRABAJOS con Prioridad 1

ID Usuario	Título
101	1
102	2
103	3
104	4
105	5
106	6
107	7
108	8
109	9
110	10
111	11
112	12
113	13
114	14
115	15
116	16
117	17
118	18
119	19
120	20

MENU PRINCIPAL

```
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 1

Se ha eliminado el trabajo <101><1><1> de la cola
Se ha eliminado el trabajo <103><3><3> de la cola
Se ha eliminado el trabajo <105><5><5> de la cola
Se ha eliminado el trabajo <107><7><7> de la cola
Se ha eliminado el trabajo <109><9><9> de la cola
Se ha eliminado el trabajo <111><11><11> de la cola
Se ha eliminado el trabajo <113><13><13> de la cola
Se ha eliminado el trabajo <115><15><15> de la cola
Se ha eliminado el trabajo <117><17><17> de la cola
Se ha eliminado el trabajo <119><19><19> de la cola
Se ha eliminado el trabajo <102><2><2> de la cola
Se ha eliminado el trabajo <108><8><8> de la cola
Se ha eliminado el trabajo <114><14><14> de la cola
Se ha eliminado el trabajo <120><20><20> de la cola
Se ha eliminado el trabajo <104><4><4> de la cola
Se ha eliminado el trabajo <116><16><16> de la cola


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 1


        INFORME DE TRABAJOS con Prioridad 1

ID Usuario              Titulo
-------------------------------------------------------------
   106                  6
   110                  10
   112                  12
   118                  18


      MENU PRINCIPAL
```

Otro ejemplo:

```
Elija opcion: 6

  Introduce una prioridad: 2


        INFORME DE TRABAJOS con Prioridad 2

ID Usuario              Titulo
_____________________________________________________________
   201                  1
   202                  2
   203                  3
   204                  4
   205                  5
   206                  6


      MENU PRINCIPAL
```

```
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 2

Se ha eliminado el trabajo <201><1><1> de la cola
Se ha eliminado el trabajo <203><3><3> de la cola
Se ha eliminado el trabajo <205><5><5> de la cola
Se ha eliminado el trabajo <202><2><2> de la cola


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 2


        INFORME DE TRABAJOS con Prioridad 2

ID Usuario              Titulo
_____________________________________________________________
   204                  4
   206                  6


      MENU PRINCIPAL
```

Otro ejemplo (cola con 2 trabajos: numElemsCola = distancia inicial = 2):

```
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 2

Se ha eliminado el trabajo <204><4><4> de la cola


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 2


        INFORME DE TRABAJOS con Prioridad 2

ID Usuario              Titulo
-------------------------------------------------------------------
   206                  6


      MENU PRINCIPAL
```

Otro ejemplo (cola con 1 solo trabajo: 1 = numElemsCola < distancia inicial = 2):

```
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 2

No se ha reducido la espera en la prioridad 2, ya que solo hay 1 trabajo en esta cola.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 2


        INFORME DE TRABAJOS con Prioridad 2

ID Usuario              Titulo
_____________________________________________________________
   206                  6


      MENU PRINCIPAL
```

Otro ejemplo (cola vacía)

```
0.- Salir
Elija opcion: 6

  Introduce una prioridad: 9


ERROR: No hay trabajos con esa prioridad a la espera de impresion.


      MENU PRINCIPAL
1.- Enviar un trabajo a la impresora
2.- Imprimir trabajos
3.- Mostrar trabajo mas pesado
4.- Mostrar tiempo de espera de un usuario
5.- Informe de trabajos por prioridad
6.- Informe de trabajos de una prioridad
7.- Reducir espera en una prioridad
8.- Reiniciar el Sistema de impresion
0.- Salir
Elija opcion: 7

  Introduce una prioridad: 9


ERROR: No hay trabajos con esa prioridad a la espera de impresion.


      MENU PRINCIPAL
```

Recordatorio: La Opción 8 ya fue comprobada al probar antes el correcto funcionamiento de la opción 3.

Y en este punto, ya están probadas y comprobadas todas las pruebas de ejecución de las posibilidades o casos posibles en todas las opciones del Menú Principal del programa.

3.- Cuestiones

3.1.- Cuestión 1:

Si hubiésemos llevado a cabo la implementación en lenguaje C, ¿cuál sería el proceso a implementar en la opción 8 del menú?. Represéntelo en pseudocódigo. ¿Cuál es la diferencia fundamental respecto a la implementación que hemos realizado en Java?

Sea `tRegTrabajo` un struct equivalente al objeto **Trabajo** en Java, y teniendo un array de tamaño dinámico de `tRegTrabajo` de la manera `tRegTrabajo cola[cont]`.
El método **reiniciarSistema** quedaría de la siguiente forma:

```
void reiniciarSistema(tRegTrabajo cola[cont]){

     int i;
     struct **cola = malloc(10*sizeof(struct*));

     for(i = 0; i < cont; i++)
          cola[i] = malloc(i+1);
     for(i = 0; i < 10; i++)
          free(cola[i]);
}
```

Planteado el problema en lenguaje C, al no tener este lenguaje un recolector automático de basura, utilizamos la funciones **malloc()** que asigna el número de bytes específicos y la función **free()** que libera el bloque de memoria especificada.

3.2.- Cuestión 2:

Si consideramos la implementación del TAD Cola realizada en clase de teoría mediante la clase ArrayCola… ¿Qué diferencias existen entre la aplicación implementada en esta práctica y la que habríamos realizado si hubiésemos utilizado la clase ArrayCola?

El TAD cola planteado en clase basa su implementación en un array de tamaño dinámico que almacena referencias al objeto genérico (en nuestro caso sería Trabajo), mientras que nuestra cola es una lista formada por nodos genéricos que se ordenan como haría el ArrayCola.
Las operaciones usando la clase ArrayCola resultan más simples puesto que no hay que tener en cuenta el uso de nodos y se trabaja directamente con las referencia a objetos y posiciones del array o vector. Además de estar más familiarizados con su uso y haber una gran cantidad de tareas ya automatizadas en el uso de este tipo de colección.