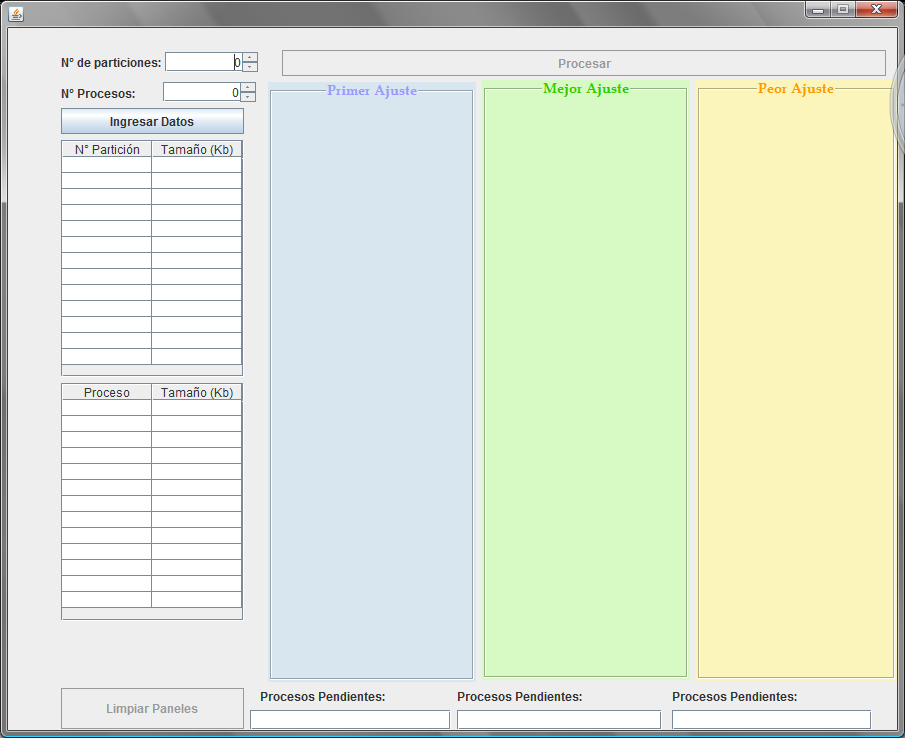
ADMINISTRACION DE MEMORIA -PROGRAMA: ESQUEMA DE ASIGNACION DE PARTICIONES MULTIPLES

El siguiente programa ha sido elaborado con el IDE de desarrollo para java Netbeans 6.8 , lenguaje Java.

***INTERFAZ DE PROGRAMA:***

***Control JButton1***: colocara cada proceso en la partición que le corresponde de acuerdo al ajuste que se haga.

***Control Spinner1:*** se hará el ingreso del número de particiones.



***Control Spinner2:*** se hará el ingreso del número de procesos.

***Control JButton2***: permitirá el ingreso de particiones y procesos, de acuerdo a la cantidad especificada.

***Control JButton2***: permitirá el ingreso de particiones y procesos, de acuerdo a la cantidad especificada.

***Control JPanel2***: Mostrara la Grafica de la Memoria con las particiones respectivas, y los procesos a los que se ha asignado de acuerdo al Mejor ajuste

***Control JTable2***: se ingresaran los tamaños de cada partición.

***Control JPanel1***: Mostrara la Grafica de la Memoria con las particiones respectivas, y los procesos a los que se ha asignado de acuerdo al Primer ajuste

***Control JPanel3***: Mostrara la Grafica de la Memoria con las particiones respectivas, y los procesos a los que se ha asignado de acuerdo al Peor ajuste

***Control JTable1***: se ingresaran los tamaños de de cada Proceso

***Control JButton3***: Limpia los paneles, es decir las graficas, para un nuevo ingreso de datos

***Control JTextField***: Mostrara la suma de Procesos pendientes en cada Ajuste.

* La función ***Nparticiones***, nos retorna el número de particiones de la memoria.
* La función ***Nprocesos***, nos retorna el número de procesos a los cuales se les asignara memoria.
* El vector ***v\_part***, guardará todos los valores de las particiones insertados en la tabla ***jtable2***, en la columna”***tamaño”***
* El vector ***v\_proc,*** guardará el tamaño de cada proceso insertado en la tabla ***jtable1,*** en la columna ***“tamaño”***

***FUNCION PRIMER AJUSTE:***

Esta función lo primero que hará es dibujar en el panel jPanel1,la memoria con el numero de particiones que le corresponde, luego se recorrerá los vectores ***v\_part*** y ***v\_proc,*** y se comenzara a comparar a cada proceso con cada una de las particiones, si se encuentra que es menor que una de éstas, entonces en la partición correspondiente cargará éste proceso, ***,*** luego con la función ***InsertarProcesos*** y en el interfaz del programa en la posición que asigne la variable ***acum*** se mostrará como texto por ejemplo: ***“proceso 1: 1000 kb”,*** y por último el valor de la partición en la que cargo el proceso se hará cero, así como también el valor del proceso que cargo en la partición, todo esto se hará para tener un control de los procesos que ya se les esta asignando memoria.

public void PrimerAjuste(){

Vector<Integer> v\_part= new Vector<Integer>() ;Vector<Integer> v\_proc= new Vector<Integer>() ;

for(int i=1;i<Nparticiones();i++){

v\_part.add(Integer.parseInt(String.valueOf(jTable2.getValueAt(i,1)))); }

for(int j=0;j<Nprocesos();j++){

v\_proc.add(Integer.parseInt(String.valueOf(jTable1.getValueAt(j,1))));}

int v=0;

int acum\_y=45;//posision y desde donde se ubicaran los procesos en las particiones

for(int j=0;j<v\_part.size();j++){ v=0;

for(int i=0;i<v\_proc.size();i++){

if(v==0){

if(v\_proc.get(i)<v\_part.get(j) && !v\_proc.get(i).equals(new Integer(0))){

acum\_y=67+45\*(j);

InsertarProcesos ("Proceso "+(i+1)+": "+v\_proc.get(i)+"Kb",acum\_y,false,jPanel1);

v\_part.set(j,new Integer(0));

v=1;}

else if(v\_proc.get(i)==v\_part.get(j) && !v\_proc.get(i).equals(new Integer(0))){

acum\_y=67+45\*(j);

InsertarProcesos ("Proceso "+(i+1)+": "+v\_proc.get(i)+"Kb",acum\_y,true,jPanel1);

v\_part.set(j,new Integer(0));

v=1;

}if(v==1){ v\_proc.set(i,new Integer(0)); }}}}

ProcPendientes\_PartVacias(v\_proc,v\_part,jTextField1,jPanel1 );

***FUNCION MEJOR AJUSTE:***

Primero se comenzara a recorrer los dos vectores: ***v\_part y v\_proc***, esta función trabajara con un vector resta de tipo hashtable, en este vector se guardarán las restas entre el tamaño del proceso y el tamaño de la partición, para ver cual resta es la mínima, éste valor se guardará en la variable ***min,*** y la posición de ésta resta, se guarda en la variable ***pos,*** así entonces se irá obteniendo las restas de cada proceso con cada una de las particiones, y con la función ***InsertarProcesos,*** se mostrará en el interfaz del programa qué procesos están cargados en las particiones

public void MejorAjuste(){

Vector<Integer> v\_part= new Vector<Integer>() ;

Vector<Integer>v\_proc= new Vector<Integer>() ;

Hashtable<Integer,Integer> resta= new Hashtable<Integer,Integer>() ;

for(int i=1;i<Nparticiones();i++){

v\_part.add(Integer.parseInt(String.valueOf(jTable2.getValueAt(i,1))));}

for(int j=0;j<Nprocesos();j++){v\_proc.add(Integer.parseInt(String.valueOf(jTable1.getValueAt(j,1)))); }

int pos=-1; int acum\_y=45;

for(int j=0;j<v\_proc.size();j++){

for(int i=0;i<v\_part.size();i++){

if(!v\_part.get(i).equals(new Integer(0)) && !v\_proc.get(j).equals(new Integer(0))){

resta.put(i,v\_part.get(i)-v\_proc.get(j));} }

Enumeration<Integer> e=resta.keys();

Integer min= new Integer(100000);

while(e.hasMoreElements()){

Integer indice=e.nextElement();

if(resta.get(indice)<min && resta.get(indice)>=new Integer(0)){

min=resta.get(indice);

pos=indice; }

} if(!min.equals(new Integer(100000))){

acum\_y=67+45\*(pos);

if(!min.equals(new Integer(0))){

InsertarProcesos ("Proceso "+(j+1)+": "+v\_proc.get(j)+"Kb",acum\_y,false,jPanel2);

}

else{

InsertarProcesos ("Proceso "+(j+1)+": "+v\_proc.get(j)+"Kb",acum\_y,true,jPanel2);

}

v\_proc.set(j,new Integer(0));}

resta.clear();//limpiamos el vestor resta

if(pos!=-1)v\_part.set(pos,new Integer(0));

}ProcPendientes\_PartVacias(v\_proc,v\_part,jTextField2,jPanel2 );}

***FUNCION PEOR AJUSTE:***

El procedimiento es el mismo que la función del Mejor ajuste, solo que aquí en vez de guardar la resta mínima, se guardara la máxima, pues el algoritmo se trata de que cuando un proceso llega, tiene que cargar el partición mas grande que haya.

public void PeorAjuste(){

Vector<Integer> v\_part= new Vector<Integer>() ;

Vector<Integer>v\_proc= new Vector<Integer>() ;

Hashtable<Integer,Integer> resta= new Hashtable<Integer,Integer>() ;

Hashtable<Integer,Integer> Proc\_f= new Hashtable<Integer,Integer>() ;//particion y su posicion

Hashtable<Integer,Integer> Part\_f= new Hashtable<Integer,Integer>() ;// proceso y su particion

for(int i=1;i<Nparticiones();i++){

v\_part.add(Integer.parseInt(String.valueOf(jTable2.getValueAt(i,1))));}

for(int j=0;j<Nprocesos();j++){

v\_proc.add(Integer.parseInt(String.valueOf(jTable1.getValueAt(j,1)))); }

int pos=-1;

int acum\_y=45;//posision y desde donde se ubicaran los procesos en las particiones

for(int j=0;j<v\_proc.size();j++){

for(int i=0;i<v\_part.size();i++){

if(!v\_part.get(i).equals(new Integer(0)) && !v\_proc.get(j).equals(new Integer(0))){

resta.put(i,v\_part.get(i)-v\_proc.get(j));

} }

Enumeration<Integer> e=resta.keys();

Integer max=new Integer(-1);

while(e.hasMoreElements()){

Integer indice=e.nextElement();

if(resta.get(indice)>=max && resta.get(indice)>=new Integer(0)){

max=resta.get(indice);

pos=índice; }

} if(!max.equals(new Integer(-1))){

acum\_y=67+45\*(pos);

if(!max.equals(new Integer(0))){

InsertarProcesos ("Proceso "+(j+1)+": "+v\_proc.get(j)+"Kb",acum\_y,false,jPanel3);

}

else{

InsertarProcesos ("Proceso "+(j+1)+": "+v\_proc.get(j)+"Kb",acum\_y,true,jPanel3);

}

v\_proc.set(j,new Integer(0));}

resta.clear();//limpiamos el vestor resta

if(pos!=-1) v\_part.set(pos,new Integer(0));}

ProcPendientes\_PartVacias(v\_proc,v\_part,jTextField3,jPanel3 ); }

***FUNCION INSERTA PARTICIONES: FUNCION INSERTAPROCESOS***

public void InsertarProcesos(String proceso,int acum\_y,boolean igual\_o\_menor,JPanel jPanel){

int x=35;

JTextField txtp,txt;

txtp=new JTextField();

txt=new JTextField();

if (igual\_o\_menor==true){

txtp.setSize(105,43);}

else{

txtp.setSize(105,30);

txt.setSize(105,43-30);

txt.setLocation(x,acum\_y+30);

}

txtp.setText(proceso);

txtp.setLocation(x,acum\_y);

jPanel.add(txtp);

jPanel.add(txt);

txtp.setEditable(false);

txt.setEditable(false);

}

public void InsertarParticiones(int x, int y,JPanel jPanel){

int total;

total=0;

for(int i = 0;i<Nparticiones() ;i++){

y=y+45;

JTextField txt;

JLabel lbl;

JSeparator separador;

txt=new JTextField();

lbl=new JLabel();

separador=new JSeparator();

separador.setSize(105+10,2);

lbl.setSize(30,10);

if(i==0){//Particion del S.O

txt.setBackground(Color.lightGray);

txt.setSize(105,43-25);

separador.setLocation(x,y);

txt.setLocation(x,y+2);

txt.setText("S.O");

lbl.setLocation(x+105,y+45-25+2);

y=y-25;

}

else{

separador.setLocation(x,y);

lbl.setLocation(x+105,y+45+2);

}

jPanel.add(separador);

jPanel.add(lbl);

jPanel.add(txt);

separador.setBackground(Color.black);

total=total+Integer.parseInt(String.valueOf(jTable2.getValueAt(i,1)));

lbl.setText(String.valueOf(total))

}

}

***FUNCION PARA PROCESOS PENDIENTES:***

void ProcPendientes\_PartVacias(Vector<Integer> v\_proc,Vector<Integer> v\_part, JTextField jText,JPanel jPanel){

int suma=0,i=1;

int acum\_y=45;

String texto=" ";

for (Integer p:v\_proc){//for each

if(!p.equals(new Integer(0))){ texto=texto+"P"+String.valueOf(i)+" + "; suma+=p;} i+=1;

}

jText.setText(texto.substring(1,texto.length()-2)+" = "+suma+" Kb.");

i=0;

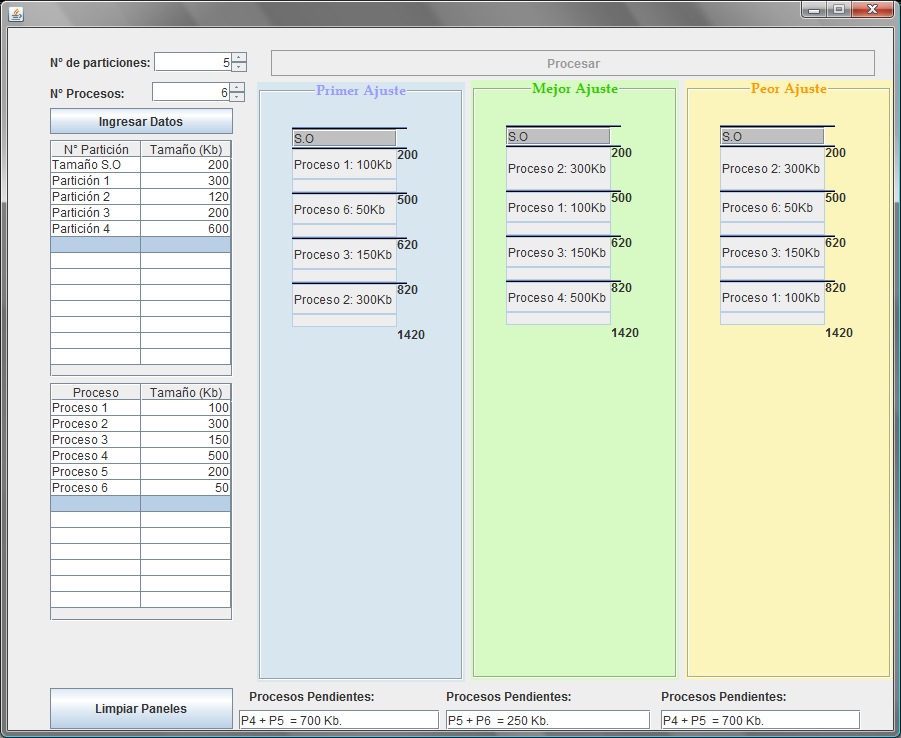
for (Integer par:v\_part){//for each

if(!par.equals(new Integer(0))){ acum\_y=67+45\*(i);InsertarProcesos ("",acum\_y,true,jPanel);}i+=1;

}

}

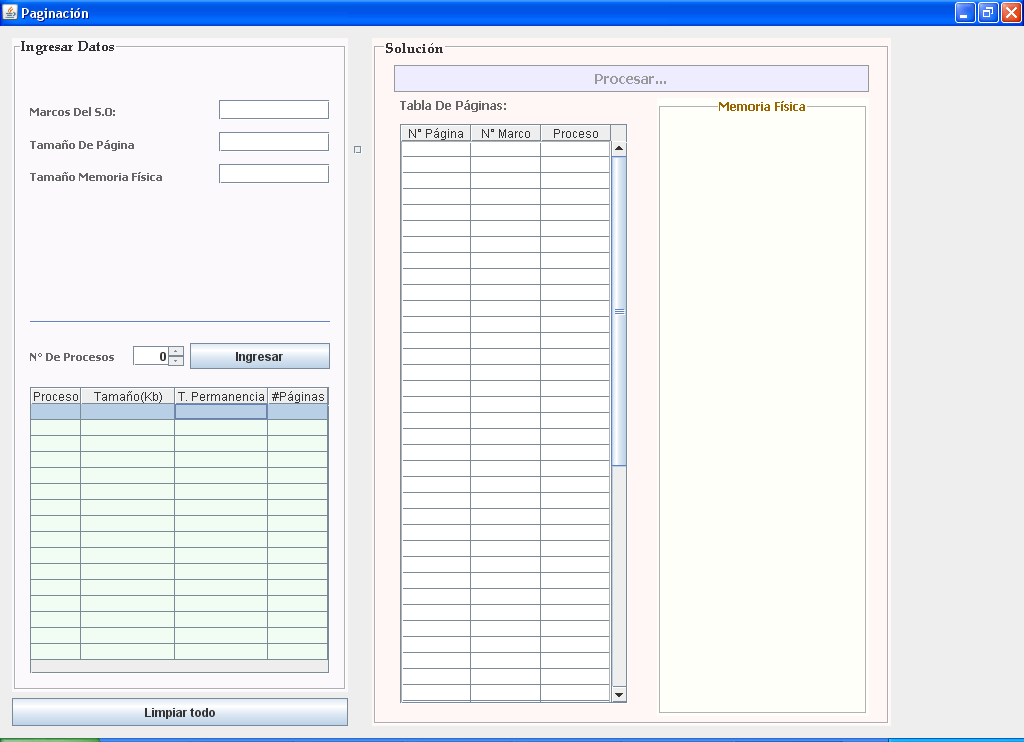
***EJECUCION DEL PROGRAMA-EJEMPLO***



ADMINISTRACIÓN DE MEMORIA-PROGRAMA :ESQUEMA DE PAGINACIÓN)

1. ***Descripción del Programa:***

La implementación del Método de Paginación, ha sido elaborada con la Plataforma Netbeans, utilizando el lenguaje Java.

1. ***Descripción del Interfaz del Programa:***

***AREA 3***

***AREA 5***

***AREA 2***

***AREA 4***

***AREA 1***

* ***Área 1:*** Se encuentra dentro del ***control JPanel (jPanel2),*** y dentro de ella se hará el ingreso de los siguientes datos: tamaño del Sistema Operativo ***(control jTextField1),*** tamaño de página***(control jTextField2)*** y Tamaño de de Memoria Física ***(control jTextField3),***
* ***Área 2:*** Se encuentra también del ***control JPanel (jPanel2),*** y dentro de ella se ingresará el número de procesos que pedirán memoria ***(control jSpinner1),*** y en una tabla ***(Control jTable1), se*** mostrará dichos procesos con sus respectivos tamaños, tiempo de permanencia en la CPU, y el numero de paginas que se requiere para cada proceso.
* ***Área 3:*** Se encuentra dentro de un Panel ***(control jPanel2),***y dentro de éste Panel encontramos un botón ***(control jButton1)***, el cual se encargará de procesar la información en cada instante de tiempo en el que entra y sale un proceso de la memoria.
* ***Área 4:*** Nos muestra la Tabla de Paginas ***(Control jTable2)*** con las páginas cargadas en memoria y los marcos en los que carga cada una de ellas, además se muestra el proceso al cual pertenecen, está tabla de paginas variará en cada instante de tiempo n el que entra y sale un proceso de Memoria.
* ***Área 5:*** En esta área se dibujara el Esquema de la distribución de páginas en la Memoria Física ***(control jPanel1).***

1. ***DESCRIPCIÓN DEL CÓDIGO FUENTE***
   1. ***Descripción de Vectores, variables usadas***

* ***La variable nveces, guardara el numero de veces que se hace click en el botón procesar (control jButton1),y así poder saber si la información que se ingresa en la tabla jtable2 es nueva, y por ende la paginación que se hará, será nueva.***
* ***El vector v\_marco guardará la lista de marcos disponibles en un inicio en la memoria física sin tomar en cuenta los marcos ocupados por el sistema operativo.***
* ***El vector v\_tiempo guarda los tiempos de permanencia de los procesos que van entrando a la Memoria física, en cada instante de tiempo.***
* ***El vector v\_pags\_proc, guarda el número de páginas que necesita cada proceso.***
* ***El vector de tipo integer: v\_proc\_pendientes, guardara los procesos pendientes en un instante de tiempo.***
* ***La función n\_procesos retorna el valor que se ingrese en el control jSpinner1, es decir el numero de procesos que pedirán memoria.***
* ***La función Marcos\_so, retorna el valor que se ingrese en el control jTextField1, es decir la cantidad de marcos que ocupa el sistema operativo.***
* ***La función tamañoPag, retorna el valor que se ingrese en el control jTextField2, es decir el numero de paginas que necesite cada proceso.***
* ***La función TamañoMem, retorna el valor que se ingrese en el control jTextField3, es decir el tamaño total de la memoria física.***
* ***La función Nmarcos retorna el valor después de dividir el tamaño de la memoria física y el tamaño de páginas, para poder hallar el número de marcos totales.***
  1. ***Descripción de Funciones Creadas***
     1. ***Función Paginación:***

Lo primero que hará esta función es comprobar si la variable ***nveces*** es igual o diferente de cero,

***Si estamos dentro del primer caso (nveces=0, inicio del proceso de paginacion):***

Se llenara el vector: ***v\_marco***, con sus respectivos índices, luego se llenara el vector ***v\_proc\_pendientes*** con los valores que se encuentran en la columna “tamaño” de la tabla ***jtable1,*** es decir el tamaño de cada proceso que se ingresaron en un inicio. A continuación se declara la variable ***tamaño\_proc***, que guardara el tamaño de cada proceso, para que pueda ser dividido la variable ***tamaño\_pag***, (guarda el valor de la función ***(TamañoPag),*** y con esto podamos obtener el numero de páginas que necesita cada proceso, y estos últimos valores serán guardados en el vector ***v\_pags\_proc.***

Finalmente los valores del vector ***v\_pags\_proc*** se agregaran a la columna ***“#paginas” de la tabla jtable1.***

**public void Paginacion\_(){**

PTexSale=" Sale el proceso: ";

PTexTiempo=" En el tiempo T=";

if(nveces==0){

for(Integer i=Marcos\_so();i<Nmarcos();i++){

v\_marco.add(i);

}

for(int i=0;i<n\_procesos();i++){

v\_Proc\_pendientes.add(Integer.parseInt(String.valueOf(jTable1.getValueAt(i,1))));

}

for(int j=0;j<n\_procesos();j++){

Double tamaño\_proc,tamaño\_pag,ceil;

tamaño\_proc=Double.parseDouble(String.valueOf(jTable1.getValueAt(j,1)));

tamaño\_pag=TamañoPag().doubleValue();

ceil=Math.ceil(tamaño\_proc/tamaño\_pag);

v\_Pags\_proc.add(ceil.intValue());

}

for(int i=0;i<v\_Pags\_proc.size();i++){

jTable1.setValueAt(v\_Pags\_proc.get(i), i,3);

}

}

***Si estamos dentro del primer caso (nveces>0):***

Se llama a la función ***EliminaOcupados,*** luego pasamos a hallar el tiempo mínimo que se encuentra en el vector ***v\_tiempo***, esto se hace para ver cada qué tiempo tendremos que detenernos y ver si algún proceso libera memoria y si otro proceso pendiente puede entrar a cargar en ese espacio libre, el tiempo mínimo hallado lo guardamos en la variable ***min,*** este valor tendrá que restarse a cada valor del vector ***v\_tiempo*** que sea mayor a cero***,*** y así poder llevar un control de los tiempos de permanencia de cada proceso en cada instante de tiempo.

En una tabla temporal ***(jTable3),*** se copiara la información de los procesos que aún están ocupando marcos en la memoria física de la tabla ***jtable2***,al momento de copiar los datos de una tabla a otra, se tiene que tener en cuenta el índice desde donde se copiaran los elementos, este índice se guardara en la variable ***pos\_tabla\_temp,*** todo esto se hace para mantener la información actual de los marcos libres y ocupados en cada instante de tiempo; luego de esto, se podrá limpiar la tabla de paginas ***jtable2***, para que pueda mostrar la información en un nuevo instante de tiempo, así como también inicializar la posición de ésta tabla nuevamente en cero, con la variable ***pos\_tabla\_pag.***

Por otro lado, lo que hace ésta función es verificar desde un inicio, si el número de páginas de cada proceso que están en el vector ***v\_pags\_proc***, son menores que los marcos disponibles ***(vector v\_marco )*** en memoria y alcanzan totalmente en ella, si es que es así, inmediatamente esas paginas pasan a copiarse en orden en aquellos marcos, y para tener control de los números de marcos que ya están ocupados en un instante de tiempo, se harán cero todos aquellos marcos ocupados en el vector ***v\_marco,*** lo mismo se hace para tener el control de los procesos a los cuales ya se les asigno memoria, también se harán cero los valores de aquellos procesos en el vector ***v\_proc\_pendientes.***

A los valores anteriormente mencionados se les asigna el valor de cero, para que en otro instante de tiempo, (en el caso de los procesos), poder comparar el número de paginas solo de los procesos que están en espera de memoria, es decir los que no son cero.

Y en el caso de los marcos, el valor cero sirve para que se lleve una cuenta de los marcos libres en la función ***MarcosDisponibles***,y así ver si algún proceso pendiente puede pasar a cargar en dichos marcos.

El uso de las tabla temporal ***(jtable3) que*** esta como ***no visible,*** servirá de ayuda a la tabla de paginas ***(jtable2),*** como se dijo anteriormente, es para llevar un control de los procesos que aun no liberan memoria, así como también los marcos que ya se están liberando, estos marcos liberados lo único que harán es apilarse al vector ***v\_marco***, en el índice que les toca.

La otra parte del código muestra en el botón procesar (button1), el tiempo en que sale y entra y uno o mas procesos, para esto se ha utilizado las variables globales de tipo string ***PtextEntra, que guarda:”Entra el Proceso:”, la variable PtexSale: “Sale el Proceso:”, PtexTiempo: “En el Tiempo T=”,*** y finalmente también se ha utilizado la variable global de tipo ***integer TiempoAcumulado***, que guardará el tiempo actual en el que nos encontramos; todos estos valores se concatenaran y mostraran en botón procesar.

if(nveces>0){

v\_marco=EliminarOcupados(v\_marco);

Enumeration<Integer> e=v\_tiempo.keys();

Integer min= new Integer(100000);

while(e.hasMoreElements()){

Integer indice=e.nextElement();

if(v\_tiempo.get(indice)<min && v\_tiempo.get(indice)>cero){

min=v\_tiempo.get(indice) }

Hashtable<Integer,Integer> tiempos\_temporales= new Hashtable<Integer,Integer> ();

Enumeration<Integer> temp=v\_tiempo.keys();

while(temp.hasMoreElements()){

Integer i=temp.nextElement();

if(v\_tiempo.get(i).equals(min)){

tiempos\_temporales.put(i, v\_tiempo.get(i));

PTexSale=PTexSale+(i)+", "; }

tiempos\_temporales.clear();

for(int i=0;i<jTable2.getRowCount();i++){

jTable2.setValueAt(" ", i,0);

jTable2.setValueAt(" ", i,1);

jTable2.setValueAt(" ", i,2); }

pos\_tabla\_pag=0;

for(int j=0;j<Nmarcos()-Marcos\_so()-v\_marco.size();j++){

TablaPagina(j,Integer.parseInt(String.valueOf(jTable3.getValueAt(j,0))),Integer.parseInt(String.valueOf(jTable3.getValueAt(j,1))),String.valueOf(jTable3.getValueAt(j,2)),jTable2);

pos\_tabla\_pag=j+1 }

jTable3.removeRowSelectionInterval(0,39);}

PTexEntra = " Entra el proceso: ";

for (int i = 0; i < v\_Pags\_proc.size(); i++) {

if (!v\_Proc\_pendientes.get(i).equals(cero)) {

if (v\_Pags\_proc.get(i).intValue() <= MarcosDisponibles(v\_marco).intValue()) {

PTexEntra = PTexEntra + (i + 1) + ", ";

Integer cont = 0;

int n\_paginas = v\_Pags\_proc.get(i).intValue();

for (int j = 0; j < v\_marco.size(); j++) {

if (!v\_marco.get(j).equals(cero)) {

TablaPagina(pos\_tabla\_pag, cont, v\_marco.get(j), "Proc" + (i + 1), jTable2);

pos\_tabla\_pag++;

v\_marco.set(j, cero);

cont++;

if (cont == n\_paginas) { v\_tiempo.put(Integer.parseInt(String.valueOf(jTable1.getValueAt(i,0)).substring(4)), Integer.parseInt(String.valueOf(jTable1.getValueAt(i, 2))));

v\_Proc\_pendientes.set(i, cero);

break;

}

}

}

}

}

}

if(PTexEntra.equals(" Entra el proceso: ")){PTexEntra="";}

else{ PTexEntra=PTexEntra.substring(0,PTexEntra.length()-2)+" ";}

if(PTexSale.equals(" Sale el proceso: ")){ PTexSale="";}

else{ PTexSale=PTexSale.substring(0,PTexSale.length()-2)+" ";}

if(TiempoAcumulado>= new Integer(100000)){PTexTiempo="Terminó Todo..."; jButton1.setEnabled(false);}

}

* + 1. ***Función EliminarOcupados:***

Esta función es de tipo vector, y lo que hace es recorrer el vector v\_marco, y ver que números de marcos están ocupados (los que son cero) a estos últimos los elimina y una vez que los elimina, retorna el vector, pero solo con los numero de marcos que están libres(los que no son cero).

**public Vector<Integer> EliminarOcupados(Vector<Integer>v\_marco)**{

for(int i=0;i<v\_marco.size();i++){

if(v\_marco.get(i).equals(cero)){

v\_marco.remove(i);

i--;

}

}

return v\_marco;

}

* + 1. ***Función MarcosDisponibles:***

Es una función de tipo integer, y lo que hace es contar cuantos marcos están disponibles, para si en caso exista algún proceso pendiente con menos o igual numero de paginas que los marcos disponibles los pueda ocupar, para esto se utiliza una variable ***cont***, que se inicializa en cero, y aumenta a la medida en que se encuentra un marco diferente de cero en el vector ***v\_marco,*** luego la función nos retorna la variable ***cont.***

**public Integer MarcosDisponibles(Vector<Integer> marco)**{

Integer cont =cero;

for(Integer o:marco){

if(!o.equals(cero))cont+=uno;

}

return cont;

}

* + 1. ***Función TablaPágina***

Esta función no nos retorna ningún valor, lo único que hace es copiar los elementos de una tabla a otra, por eso tendrá como parámetros, la posición de la fila desde donde copiara los valores (i), el nombre de columnas de la tabla, y la tabla a donde se copiara la información. Por ejemplo, llenar la tabla de paginas (jtable2) con los datos de la tabla temporal, con sus valores correspondientes a cada fila y columna.

public void TablaPagina(int i,Integer pagina,Integer marco,String proceso,JTable jTable){

jTable.setValueAt(pagina, i, 0);

jTable.setValueAt(marco, i,1);

jTable.setValueAt(proceso, i,2);

}

* + 1. ***Función InsertarMarcos***

Lo que hace esta función es insertar como texto el nombre de cada marco en el área donde se dibuja la memoria física en el interfaz del programa y al mismo tiempo dividir la memoria con el número de marcos que ésta tendrá de acuerdo a los datos ingresados en el ***área 1*** delinterfaz del programa, pues se tendrá en cuenta el número de marcos asignados al sistema operativo.

**public void InsertarMarcos(int x, int y,JPanel jPanel){**

for(int i=Marcos\_so();i<=Nmarcos()+1;i++){

y=y+20;

JTextField txt=new JTextField();

JLabel marco=new JLabel();

JSeparator separador=new JSeparator();

separador.setSize(100,2);

marco.setSize(45,10);

txt.setBackground(Color.yellow);

txt.setSize(100,30-5);

separador.setLocation(x,y);

separador.setBackground(Color.black);

if (i==Nmarcos()+1){

jPanel.add(separador);

continue;

}

if(i==Marcos\_so()){

txt.setSize(100,30+10);

jPanel.add(txt);

txt.setLocation(x,y+2);

txt.setText("S.O");

y=y+2;

}

else{

jPanel.add(marco);

marco.setLocation(x-47,y+28-2);

marco.setText("Marco #"+String.valueOf(i-1)) ;

marco.setFont(new java.awt.Font("Arial",1,9));

}

jPanel.add(separador);

}

JSeparator separador=new JSeparator();

y=y+20;

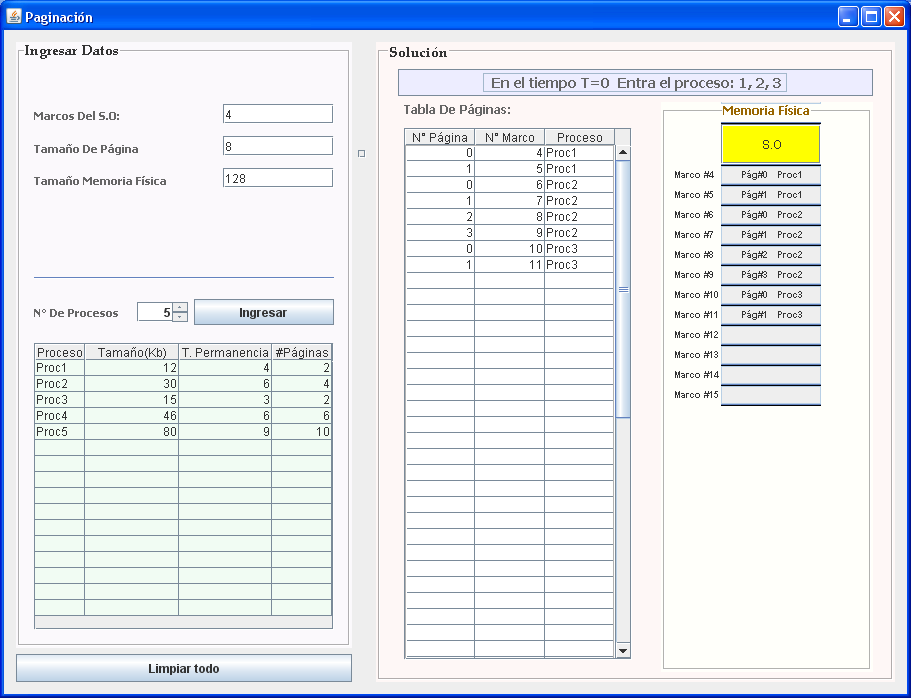
separador.setSize(100,2);

separador.setLocation(x,y);

separador.setBackground(Color.black);

jPanel.add(separador);

}

******

* + 1. ***Función InsertandoMarcosNulos***

Comenzara a dibujar la memoria física con la cantidad de marcos totales, sin considerar los que ocupa el sistema operativo

**void InsertandoMarcosNulos(){**

for (Integer marco:v\_marco){

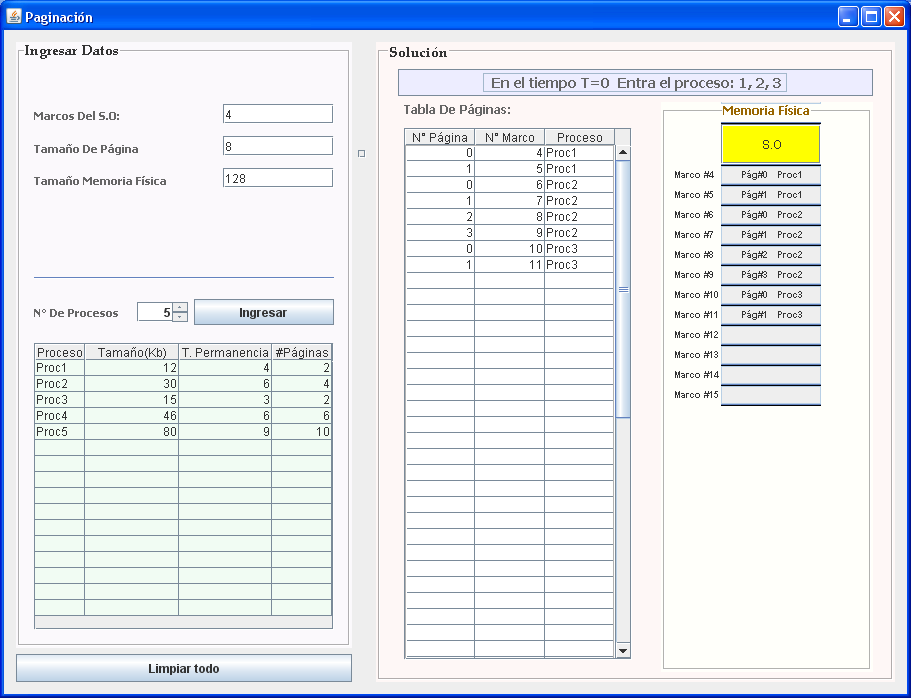
Dibujando("",marco-4,jPanel1);

}

}

* + 1. ***Función ObteniendoPaginasTabla***

Esta función lo primero que hará es recorrer la tabla de paginas (jtable2), y con estos valores, presentar la distribución de paginas de los procesos cargados en los marcos de la memoria física que se dibuja en el panel ***(jPanel1)*** del interfaz del programa, esto se hará con la ayuda de la función ***Dibujando***, que esta a continuación.



**public void ObteniendoPaginasTabla(){**

for(int i=0;i<jTable2.getRowCount();i++){

if(!jTable2.getValueAt(i,2).toString().equals(" ")){

Dibujando(" Pág#"+jTable2.getValueAt(i,0)+" "+jTable2.getValueAt(i,2),Integer.parseInt(String.valueOf(jTable2.getValueAt(i,1)))-Marcos\_so(),jPanel1);

}

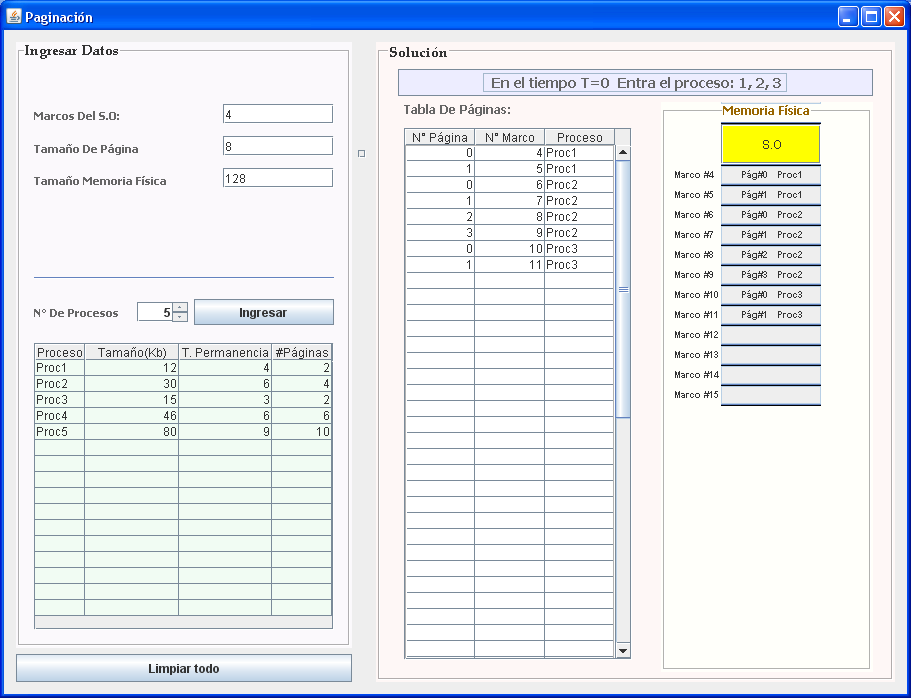
else {break;}

}

}

* + 1. ***Función Dibujando***

Esta función se encargará de presentar en el panel ***jpanel1***, la distribución de páginas que cargarán en los marcos pero en cada instante de tiempo, es decir solo nos muestra el esquema de distribución para poder visualizar como va quedando la memoria tras cargar una pagina en un marco o tras la liberación de memoria.

******

**public void Dibujando(String pagina,int indice,JPanel jPanel){**

indice=44+20\*indice;

int x=60;

JTextField txt;

txt=new JTextField();

txt.setSize(100,18);

txt.setLocation(x,indice+20);

txt.setText(pagina);

txt.setFont(new java.awt.Font("Arial", 1,9));

jPanel.add(txt);

txt.setEditable(false);

}

public void clear(){

if (jPanel1 !=null ){

Graphics gr =jPanel1.getGraphics();

repaint();

}

}

1. ***CONTENIDO DE CONTROLES***
   1. ***Botón “Ingresar “Control jButto2:***

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

for(int i=0;i<Integer.parseInt(String.valueOf(jSpinner1.getValue()));i++){

jTable1.setValueAt("Proc"+(i+1), i,0);

}

jButton1.setEnabled(true);

* 1. ***Botón “Procesar“ Control jButton1:***

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

jPanel1.removeAll();

clear();

Paginacion\_();

if(nveces==0){

jButton1.setText(PTexTiempo+"0"+PTexEntra);

}

else jButton1.setText(PTexTiempo+PTexEntra+PTexSale);

nveces+=1;

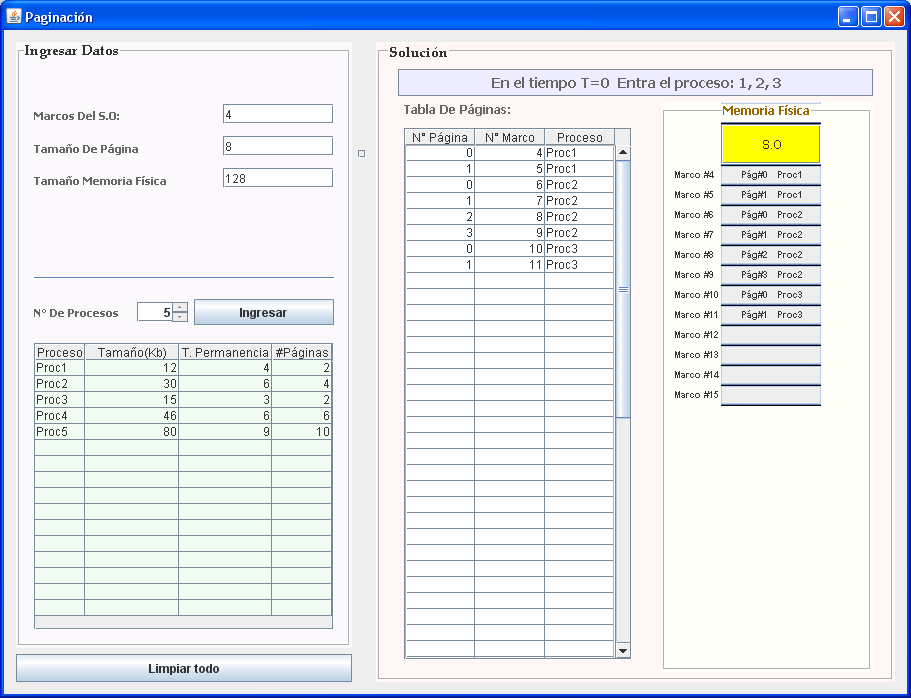
InsertarMarcos(60,0,jPanel1);

ObteniendoPaginasTabla();

InsertandoMarcosNulos();

}

1. ***EJECUCIÓN DEL PROGRAMA-EN EL INSTANTE 0***

******