Código del proyecto.

Clases:

Clases Camas:

**package** clases;

**public** **class** Camas {

**private** **int** nroCama, categoria, estado;

**private** **double** precioxdia;

**public** Camas(**int** nroCama, **int** categoria, **double** precioxdia, **int** estado) {

**this**.nroCama = nroCama;

**this**.categoria = categoria;

**this**.estado = estado;

**this**.precioxdia = precioxdia;

}

**public** **int** getNroCama() {

**return** nroCama;

}

**public** **void** setNroCama (**int** nroCama) {

**this**.nroCama = nroCama;

}

**public** **int** getCategoria() {

**return** categoria;

}

**public** **void** setCategoria(**int** categoria) {

**this**.categoria = categoria;

}

**public** **int** getEstado() {

**return** estado;

}

**public** **void** setEstado(**int** estado) {

**this**.estado = estado;

}

**public** **double** getPrecioxDia() {

**return** precioxdia;

}

**public** **void** setPrecioxDia(**double** precioxdia) {

**this**.precioxdia = precioxdia;

}

}

Clase Consumo:

**package** clases;

**public** **class** Consumo {

**private** **int** codConsumo, codPaciente, fechaConsumo, estado;

**private** **double** totalPago;

**public** Consumo(**int** codConsumo, **int** codPaciente, **int** fechaConsumo, **int** estado, **double** totalPago) {

**this**.codConsumo = codConsumo;

**this**.codPaciente = codPaciente;

**this**.fechaConsumo = fechaConsumo;

**this**.estado = estado;

**this**.totalPago = totalPago;

}

**public** **int** getCodConsumo() {

**return** codConsumo;

}

**public** **void** setCodConsumo(**int** codConsumo) {

**this**.codConsumo = codConsumo;

}

**public** **int** getCodPaciente() {

**return** codPaciente;

}

**public** **void** setCodPaciente(**int** codPaciente) {

**this**.codPaciente = codPaciente;

}

**public** **int** getFechaConsumo() {

**return** fechaConsumo;

}

**public** **void** setFechaConsumo(**int** fechaConsumo) {

**this**.fechaConsumo = fechaConsumo;

}

**public** **int** getEstado() {

**return** estado;

}

**public** **void** setEstado(**int** estado) {

**this**.estado = estado;

}

**public** **double** getTotalPago() {

**return** totalPago;

}

**public** **void** setTotalPago(**double** totalPago) {

**this**.totalPago = totalPago;

}

}

Clase DetalleConsumo

**package** clases;

**public** **class** DetalleConsumo {

**private** **int** codConsumo;

**private** **int** codProducto;

**private** **int** cantidad;

**private** **double** precioUnitario;

**private** **double** subtotal;

**public** DetalleConsumo(**int** codConsumo, **int** codProducto, **int** cantidad, **double** precioUnitario, **double** subtotal) {

**this**.codConsumo = codConsumo;

**this**.codProducto = codProducto;

**this**.cantidad = cantidad;

**this**.precioUnitario = precioUnitario;

**this**.subtotal = subtotal;

}

**public** **int** getCodConsumo() {

**return** codConsumo;

}

**public** **void** setCodConsumo(**int** codConsumo) {

**this**.codConsumo = codConsumo;

}

**public** **int** getCodProducto() {

**return** codProducto;

}

**public** **void** setCodProducto(**int** codProducto) {

**this**.codProducto = codProducto;

}

**public** **int** getCantidad() {

**return** cantidad;

}

**public** **void** setCantidad(**int** cantidad) {

**this**.cantidad = cantidad;

}

**public** **double** getPrecioUnitario() {

**return** precioUnitario;

}

**public** **void** setPrecioUnitario(**double** precioUnitario) {

**this**.precioUnitario = precioUnitario;

}

**public** **double** getSubtotal() {

**return** subtotal;

}

**public** **void** setSubtotal(**double** subtotal) {

**this**.subtotal = subtotal;

}

}

Clase Empleado:

**package** clases;

**public** **class** Empleado {

**private** **int** codEmpleado;

**private** String apellidos;

**private** String nombres;

**private** **int** tipoEmpleado;

**private** String login;

**private** String password;

**private** **int** turno;

**public** Empleado(**int** codEmpleado, String apellidos, String nombres, **int** tipoEmpleado, String login, String password, **int** turno) {

**this**.codEmpleado = codEmpleado;

**this**.apellidos = apellidos;

**this**.nombres = nombres;

**this**.tipoEmpleado = tipoEmpleado;

**this**.login = login;

**this**.password = password;

**this**.turno = turno;

}

**public** **int** getCodEmpleado() {

**return** codEmpleado;

}

**public** **void** setCodEmpleado(**int** codEmpleado) {

**this**.codEmpleado = codEmpleado;

}

**public** String getNombres() {

**return** nombres;

}

**public** **void** setNombres(String nombres) {

**this**.nombres = nombres;

}

**public** String getApellidos() {

**return** apellidos;

}

**public** **void** setApellidos(String apellidos) {

**this**.apellidos = apellidos;

}

**public** **int** getTipoEmpleado() {

**return** tipoEmpleado;

}

**public** **void** setTipoEmpleado(**int** tipoEmpleado) {

**this**.tipoEmpleado = tipoEmpleado;

}

**public** String getLogin() {

**return** login;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **int** getTurno() {

**return** turno;

}

**public** **void** setTurno(**int** turno) {

**this**.turno = turno;

}

}

Clase Hospitalización:

**package** clases;

**public** **class** Hospitalizacion {

// Atributos Privados

**private** **int** CodHosp, CodPaci, CodEmpl, NumCama, FecLleg, HorLleg, FecSali, HorSali, Estado;

// Constructor

**public** Hospitalizacion(**int** CodHosp, **int** CodPaci, **int** CodEmpl, **int** NumCama, **int** FecLleg, **int** HorLleg, **int** FecSali,

**int** HorSali, **int** Estado) {

**this**.CodHosp = CodHosp;

**this**.CodPaci = CodPaci;

**this**.CodEmpl = CodEmpl;

**this**.NumCama = NumCama;

**this**.FecLleg = FecLleg;

**this**.HorLleg = HorLleg;

**this**.FecSali = FecSali;

**this**.HorSali = HorSali;

**this**.Estado = Estado;

}

// Métodos de acceso público: set/get

**public** **int** getCodHosp() {

**return** CodHosp;

}

**public** **void** setCodHosp(**int** codHosp) {

**this**.CodHosp = codHosp;

}

**public** **int** getCodPaci() {

**return** CodPaci;

}

**public** **void** setCodPaci(**int** codPaci) {

**this**.CodPaci = codPaci;

}

**public** **int** getCodEmpl() {

**return** CodEmpl;

}

**public** **void** setCodEmpl(**int** codEmpl) {

**this**.CodEmpl = codEmpl;

}

**public** **int** getNumCama() {

**return** NumCama;

}

**public** **void** setNumCama(**int** numCama) {

**this**.NumCama = numCama;

}

**public** **int** getFecLleg() {

**return** FecLleg;

}

**public** **void** setFecLleg(**int** fecLleg) {

**this**.FecLleg = fecLleg;

}

**public** **int** getHorLleg() {

**return** HorLleg;

}

**public** **void** setHorLleg(**int** horLleg) {

**this**.HorLleg = horLleg;

}

**public** **int** getFecSali() {

**return** FecSali;

}

**public** **void** setFecSali(**int** fecSali) {

**this**.FecSali = fecSali;

}

**public** **int** getHorSali() {

**return** HorSali;

}

**public** **void** setHorSali(**int** horSali) {

**this**.HorSali = horSali;

}

**public** **int** getEstado() {

**return** Estado;

}

**public** **void** setEstado(**int** estado) {

**this**.Estado = estado;

}

}

Clase Medicamentos:

**package** clases;

**public** **class** Medicamentos {

**private** **int** CodMedicamento, stock;

**private** String nombre, laboratorio;

**private** **double** precio;

**public** Medicamentos(**int** CodMedicamento, String nombre, String laboratorio, **double** precio, **int** stock) {

**this**.CodMedicamento = CodMedicamento;

**this**.nombre = nombre;

**this**.laboratorio = laboratorio;

**this**.precio = precio;

**this**.stock = stock;

}

**public** **int** getStock() {

**return** stock;

}

**public** **void** setStock(**int** stock) {

**this**.stock = stock;

}

**public** String getNombre() {

**return** nombre;

}

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

**public** String getLaboratorio() {

**return** laboratorio;

}

**public** **void** setLaboratorio(String laboratorio) {

**this**.laboratorio = laboratorio;

}

**public** **double** getPrecio() {

**return** precio;

}

**public** **void** setPrecio(**double** precio) {

**this**.precio = precio;

}

**public** **int** getCodMedicamento() {

**return** CodMedicamento;

}

**public** **void** setCodMedicamento(**int** codMedicamento) {

**this**.CodMedicamento = codMedicamento;

}

**public** **void** restarStock(**int** stock) {

**this**.stock = **this**.stock - stock;

}

}

Clase Paciente

**package** clases;

**public** **class** Paciente {

**private** **int** codpaciente;

**private** String apellidos, nombres, telefono, DNI;

**public** Paciente (**int** codpaciente, String apellidos, String nombres, String telefono, String DNI) {

**this**.codpaciente = codpaciente;

**this**.apellidos = apellidos;

**this**.nombres = nombres;

**this**.telefono = telefono;

**this**.DNI = DNI;

}

**public** **int** getCodpaciente() {

**return** codpaciente;

}

**public** **void** setCodpaciente(**int** codpaciente) {

**this**.codpaciente = codpaciente;

}

**public** String getApellidos() {

**return** apellidos;

}

**public** **void** setApellidos(String apellidos) {

**this**.apellidos = apellidos;

}

**public** String getNombres() {

**return** nombres;

}

**public** **void** setNombres(String nombre) {

**this**.nombres = nombre;

}

**public** String getTelefono() {

**return** telefono;

}

**public** **void** setTelefono(String telefono) {

**this**.telefono = telefono;

}

**public** String getDNI() {

**return** DNI;

}

**public** **void** setDNI(String dNI) {

**this**.DNI = dNI;

}

}

Clase Servicios

**package** clases;

**public** **class** Servicio {

// Atributos Privados

**private** **int** CodServicio, Tipo;

**private** String Descripcion;

**private** **double** Precio;

// Constructor

**public** Servicio(**int** CodServicio, String Descripcion, **int** Tipo, **double** Precio) {

**this**.CodServicio = CodServicio;

**this**.Descripcion = Descripcion;

**this**.Tipo = Tipo;

**this**.Precio = Precio;

}

// Métodos de acceso público: set/get

**public** **int** getCodServicio() {

**return** CodServicio;

}

**public** **void** setCodServicio(**int** codServicio) {

**this**.CodServicio = codServicio;

}

**public** **int** getTipo() {

**return** Tipo;

}

**public** **void** setTipo(**int** tipo) {

**this**.Tipo = tipo;

}

**public** String getDescripcion() {

**return** Descripcion;

}

**public** **void** setDescripcion(String descripcion) {

**this**.Descripcion = descripcion;

}

**public** **double** getPrecio() {

**return** Precio;

}

**public** **void** setPrecio(**double** precio) {

**this**.Precio = precio;

}

}

Arreglos del Proyecto

Arreglo Consumo

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Consumo;

public class ArregloConsumo {

private ArrayList<Consumo> con;

private String archivo;

public ArregloConsumo(String archivo) {

con = new ArrayList<Consumo>();

this.archivo = archivo;

cargarConsumo();

}

public void adicionar(Consumo c) {

con.add(c);

}

public int tamaño() {

return con.size();

}

public Consumo obtener(int i) {

return con.get(i);

}

public void eliminar(Consumo c) {

con.remove(c);

}

public Consumo buscar(int codigo) {

Consumo c;

for (int i = 0; i < tamaño(); i++) {

c = obtener(i);

if (c.getCodConsumo() == codigo)

return c;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0) {

return 10000001;

}

else {

return obtener(tamaño() - 1).getCodConsumo() + 1;

}

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

con.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

con.clear();

}

public void grabarConsumo() {

try {

PrintWriter pw;

String linea;

Consumo x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodConsumo() + ";" + x.getCodPaciente() + ";" + x.getFechaConsumo() + ";" + x.getEstado()

+ ";" + x.getTotalPago();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarConsumo() {

try {

BufferedReader br;

String linea, s[];

int codConsumo, codPaciente, fechaConsumo, estado;

double totalPago;

Consumo nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codConsumo = Integer.parseInt(s[0].trim());

codPaciente = Integer.parseInt(s[1].trim());

fechaConsumo = Integer.parseInt(s[2].trim());

estado = Integer.parseInt(s[3].trim());

totalPago = Double.parseDouble(s[4].trim());

nuevo = new Consumo(codConsumo, codPaciente, fechaConsumo, estado, totalPago);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Detalle Consumo

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.DetalleConsumo;

public class ArregloDetalleConsumo {

private ArrayList<DetalleConsumo> det;

private String archivo;

public ArregloDetalleConsumo(String archivo) {

det = new ArrayList<DetalleConsumo>();

this.archivo = archivo;

cargarDetalleConsumo();

}

public void adicionar(DetalleConsumo d) {

det.add(d);

}

public int tamaño() {

return det.size();

}

public DetalleConsumo obtener(int i) {

return det.get(i);

}

public void eliminar(DetalleConsumo d) {

det.remove(d);

}

public DetalleConsumo buscar(int codigo) {

DetalleConsumo c;

for (int i = 0; i < tamaño(); i++) {

c = obtener(i);

if (c.getCodConsumo() == codigo)

return c;

}

return null;

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

det.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

det.clear();

}

public void grabarDetalleConsumo() {

try {

PrintWriter pw;

String linea;

DetalleConsumo x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodConsumo() + ";" + x.getCodProducto() + ";" + x.getCantidad() + ";"

+ x.getPrecioUnitario() + ";" + x.getSubtotal();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarDetalleConsumo() {

try {

BufferedReader br;

String linea, s[];

int codConsumo, codProducto, cantidad;

double precioUnitario, subtotal;

DetalleConsumo nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codConsumo = Integer.parseInt(s[0].trim());

codProducto = Integer.parseInt(s[1].trim());

cantidad = Integer.parseInt(s[2].trim());

precioUnitario = Double.parseDouble(s[3].trim());

subtotal = Double.parseDouble(s[4].trim());

nuevo = new DetalleConsumo(codConsumo, codProducto, cantidad, precioUnitario, subtotal);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Servicio

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Servicio;

public class ArregloServicio {

// Atributo privado

private ArrayList<Servicio> ser;

private String archivo;

// Constructor

public ArregloServicio(String archivo) {

ser = new ArrayList<Servicio>();

this.archivo = archivo;

cargarServicios();

}

// Operaciones públicas

public void adicionar(Servicio s) {

ser.add(s);

}

public Servicio obtener(int pos) {

return ser.get(pos);

}

public int tamaño() {

return ser.size();

}

public void eliminar(Servicio x) {

ser.remove(x);

}

public Servicio buscar(int CodServicio) {

Servicio x;

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

if (x.getCodServicio() == CodServicio)

return x;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0)

return 10000001;

else

return obtener(tamaño() - 1).getCodServicio() + 1;

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

ser.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

ser.clear();

}

public void grabarServicios() {

try {

PrintWriter pw;

String linea;

Servicio x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodServicio() + ";" + x.getDescripcion() + ";" + x.getTipo() + ";" +

x.getPrecio();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarServicios() {

try {

BufferedReader br;

String linea, s[], descripcion;

int codigo, tipo;

double precio;

Servicio nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codigo = Integer.parseInt(s[0].trim());

descripcion = s[1].trim();

tipo = Integer.parseInt(s[2].trim());

precio = Double.parseDouble(s[3].trim());

nuevo = new Servicio(codigo, descripcion, tipo, precio);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Paciente

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Paciente;

public class ArregloPaciente {

private ArrayList<Paciente> pac;

private String archivo;

public ArregloPaciente(String archivo) {

pac = new ArrayList<Paciente>();

this.archivo = archivo;

cargarPacientes();

}

public void adicionar(Paciente p) {

pac.add(p);

}

public int tamaño() {

return pac.size();

}

public Paciente obtener(int i) {

return pac.get(i);

}

public void eliminar(Paciente p) {

pac.remove(p);

}

public Paciente buscar(int codigo) {

Paciente p;

for (int i = 0; i < tamaño(); i++) {

p = obtener(i);

if (p.getCodpaciente() == codigo)

return p;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0) {

return 100001;

}

else {

return obtener(tamaño() - 1).getCodpaciente() + 1;

}

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

pac.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

pac.clear();

}

public void grabarPacientes() {

try {

PrintWriter pw;

String linea;

Paciente x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodpaciente() + ";" + x.getApellidos() + ";" + x.getNombres() + ";"

+ x.getTelefono() + ";" + x.getDNI();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarPacientes() {

try {

BufferedReader br;

String linea, s[], apellidos, nombres, telefono, dni;

int codigo;

Paciente nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codigo = Integer.parseInt(s[0].trim());

apellidos = s[1].trim();

nombres = s[2].trim();

telefono = s[3].trim();

dni = s[4].trim();

nuevo = new Paciente(codigo, apellidos, nombres, telefono, dni);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Medicamento

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Medicamentos;

public class ArregloMedicamento {

private ArrayList<Medicamentos> med;

private String archivo;

public ArregloMedicamento(String archivo) {

med = new ArrayList<Medicamentos>();

this.archivo = archivo;

cargarMedicamentos();

}

public void adicionar(Medicamentos m) {

med.add(m);

}

public int tamaño() {

return med.size();

}

public Medicamentos obtener(int i) {

return med.get(i);

}

public void eliminar(Medicamentos m) {

med.remove(m);

}

public Medicamentos buscar(int codigo) {

Medicamentos m;

for (int i = 0; i < tamaño(); i++) {

m = obtener(i);

if (m.getCodMedicamento() == codigo)

return m;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0) {

return 10001;

}

else {

return obtener(tamaño() - 1).getCodMedicamento() + 1;

}

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

med.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

med.clear();

}

public void grabarMedicamentos() {

try {

PrintWriter pw;

String linea;

Medicamentos x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodMedicamento() + ";" + x.getNombre() + ";" + x.getLaboratorio() + ";"

+ x.getPrecio() + ";" + x.getStock();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarMedicamentos() {

try {

BufferedReader br;

String linea, s[], nombre, laboratorio;

int codigo, stock;

double precio;

Medicamentos nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codigo = Integer.parseInt(s[0].trim());

nombre = s[1].trim();

laboratorio = s[2].trim();

precio = Double.parseDouble(s[3].trim());

stock = Integer.parseInt(s[4].trim());

nuevo = new Medicamentos(codigo, nombre, laboratorio, precio, stock);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Hospitalización

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Hospitalizacion;

public class ArregloHospitalizacion {

// Atributo privado

private ArrayList<Hospitalizacion> hos;

private String archivo;

// Constructor

public ArregloHospitalizacion(String archivo) {

hos = new ArrayList<Hospitalizacion>();

this.archivo = archivo;

cargarHospitalizacion();

}

// Operaciones públicas

public void adicionar(Hospitalizacion h) {

hos.add(h);

}

public Hospitalizacion obtener(int pos) {

return hos.get(pos);

}

public int tamaño() {

return hos.size();

}

public void eliminar(Hospitalizacion x) {

hos.remove(x);

}

public Hospitalizacion buscar(int CodHosp) {

Hospitalizacion x;

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

if (x.getCodHosp() == CodHosp)

return x;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0)

return 1000001;

else

return obtener(tamaño() - 1).getCodHosp() + 1;

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

hos.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

hos.clear();

}

public void grabarHospitalizacion() {

try {

PrintWriter pw;

String linea;

Hospitalizacion x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodHosp() + ";" + x.getCodPaci() + ";" + x.getCodEmpl() + ";" + x.getNumCama() + ";"

+ x.getFecLleg() + ";" + x.getHorLleg() + ";" + x.getFecSali() + ";" + x.getHorSali() + ";"

+ x.getEstado();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarHospitalizacion() {

try {

BufferedReader br;

String linea, s[];

int CodHosp, CodPaci, CodEmpl, NumCama, FecLleg, HorLleg, FecSali, HorSali, Estado;

Hospitalizacion nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

CodHosp = Integer.parseInt(s[0].trim());

CodPaci = Integer.parseInt(s[1].trim());

CodEmpl = Integer.parseInt(s[2].trim());

NumCama = Integer.parseInt(s[3].trim());

FecLleg = Integer.parseInt(s[4].trim());

HorLleg = Integer.parseInt(s[5].trim());

FecSali = Integer.parseInt(s[6].trim());

HorSali = Integer.parseInt(s[7].trim());

Estado = Integer.parseInt(s[8].trim());

nuevo = new Hospitalizacion(CodHosp, CodPaci, CodEmpl, NumCama, FecLleg, HorLleg, FecSali, HorSali,

Estado);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Empleado

package arreglos;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.ArrayList;

import clases.Empleado;

public class ArregloEmpleado {

private ArrayList<Empleado> emp;

private String archivo;

public ArregloEmpleado(String archivo) {

emp = new ArrayList<Empleado>();

this.archivo = archivo;

cargarEmpleados();

}

public void adicionar(Empleado e) {

emp.add(e);

}

public int tamaño() {

return emp.size();

}

public Empleado obtener(int i) {

return emp.get(i);

}

public void eliminar(Empleado e) {

emp.remove(e);

}

public Empleado buscar(int codigo) {

Empleado e;

for (int i = 0; i < tamaño(); i++) {

e = obtener(i);

if (e.getCodEmpleado() == codigo)

return e;

}

return null;

}

public int codigoCorrelativo() {

if (tamaño() == 0) {

return 1001;

}

else {

return obtener(tamaño() - 1).getCodEmpleado() + 1;

}

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

emp.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

emp.clear();

}

public void grabarEmpleados() {

try {

PrintWriter pw;

String linea;

Empleado x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getCodEmpleado() + ";" + x.getApellidos() + ";" + x.getNombres() + ";" +

x.getTipoEmpleado() + ";" + x.getLogin() + ";" + x.getPassword() + ";" + x.getTurno();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarEmpleados() {

try {

BufferedReader br;

String linea, s[], apellidos, nombres, login, password;

int codigo, tipoEmpleado, turno;

Empleado nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

codigo = Integer.parseInt(s[0].trim());

apellidos = s[1].trim();

nombres = s[2].trim();

tipoEmpleado = Integer.parseInt(s[3].trim());

login = s[4].trim();

password = s[5].trim();

turno = Integer.parseInt(s[6].trim());

nuevo = new Empleado(codigo, apellidos, nombres, tipoEmpleado, login, password, turno);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Arreglo Camas

package arreglos;

import java.io.\*;

import java.util.ArrayList;

import clases.Camas;

public class ArregloCamas {

private ArrayList<Camas> cam;

private String archivo;

public ArregloCamas(String archivo) {

cam = new ArrayList<Camas>();

this.archivo = archivo;

cargarCamas();

}

public void adicionar(Camas c) {

cam.add(c);

}

public int tamaño() {

return cam.size();

}

public Camas obtener(int i) {

return cam.get(i);

}

public void eliminar(Camas c) {

cam.remove(c);

}

public Camas buscar(int codigo) {

Camas c;

for (int i = 0; i < tamaño(); i++) {

c = obtener(i);

if (c.getNroCama() == codigo)

return c;

}

return null;

}

public String getArchivo() {

return archivo;

}

public void eliminarAlFinal() {

if (tamaño() > 0)

cam.remove(tamaño() - 1);

}

public void eliminarTodo() {

if (tamaño() > 0)

cam.clear();

}

public void grabarCamas() {

try {

PrintWriter pw;

String linea;

Camas x;

pw = new PrintWriter(new FileWriter(archivo));

for (int i = 0; i < tamaño(); i++) {

x = obtener(i);

linea = x.getNroCama() + ";" + x.getCategoria() + ";" + x.getPrecioxDia() + ";" + x.getEstado();

pw.println(linea);

}

pw.close();

}

catch (Exception e) {

}

}

public void cargarCamas() {

try {

BufferedReader br;

String linea, s[];

int nroCama, estado, categoria;

double precioxdia;

Camas nuevo;

br = new BufferedReader(new FileReader(archivo));

while ((linea = br.readLine()) != null) {

s = linea.split(";");

nroCama = Integer.parseInt(s[0].trim());

categoria = Integer.parseInt(s[1].trim());

precioxdia = Double.parseDouble(s[2].trim());

estado = Integer.parseInt(s[3].trim());

nuevo = new Camas(nroCama, categoria, precioxdia, estado);

adicionar(nuevo);

}

br.close();

}

catch (Exception e) {

}

}

public boolean existeArchivo() {

File f = new File(archivo);

return f.exists();

}

}

Formulario Principal

package gui;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.UIManager;

import javax.swing.border.EmptyBorder;

import arreglos.ArregloCamas;

import arreglos.ArregloPaciente;

import libreria.DiseñoObjetos;

import libreria.Libreria;

import java.awt.Color;

import java.awt.Font;

public class Proyecto\_AED\_2016\_1 extends JFrame implements ActionListener {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JMenuBar menuProyecto;

private JMenu mnArchivo;

private JMenu mnMantenimiento;

private JMenu mnHospitalización;

private JMenu mnReporte;

private JMenuItem mntmSalir;

private JLabel lblFondo;

private JMenu mnPago;

private JMenu mnRegistroDeConsumo;

private JMenuItem mntmEmpleados;

private JMenuItem mntmPacientes;

private JMenuItem mntmCamas;

private JMenuItem mntmMedicamentos;

private JMenuItem mntmServicios;

private JMenuItem mntmConsumo;

private JMenuItem mntmRegistrar;

private JMenuItem mntmIniciarSesin;

private JMenuItem mntmRegistrarPago;

private JMenuItem mntmPacientesAdmitidos;

private JMenuItem mntmHospitalizacionesPorFecha;

private JMenuItem mntmEmpledadoshospitalizaciones;

private JMenuItem mntmPacientespago;

// Variable global para obtener código del empleado

private int codigoEmpleado;

// Variable global para libreria

Libreria lb = new Libreria();

DiseñoObjetos ds = new DiseñoObjetos();

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Proyecto\_AED\_2016\_1 frame = new Proyecto\_AED\_2016\_1();

frame.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Proyecto\_AED\_2016\_1() {

final int ANCHO = 900, ALTO = 600, DX = 8, DY = 53;

setTitle("MN-Global \u00AE >>> Hospital Solidaridad\u00AE >>> Proyecto AED 2016 1");

setResizable(false);

setIconImage(new ImageIcon("imagenes/hospital.png").getImage());

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds((getToolkit().getScreenSize().width - ANCHO - DX) / 2,

(getToolkit().getScreenSize().height - ALTO - DY) / 2, ANCHO + DX, ALTO + DY);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

menuProyecto = new JMenuBar();

setJMenuBar(menuProyecto);

mnArchivo = new JMenu("Archivo");

mnArchivo.setFont(new Font("Arial", Font.BOLD, 12));

mnArchivo.setForeground(Color.BLACK);

menuProyecto.add(mnArchivo);

mntmSalir = new JMenuItem("Salir");

mntmSalir.setFont(new Font("Arial", Font.BOLD, 12));

mntmSalir.setForeground(Color.BLACK);

mntmSalir.addActionListener(this);

mntmIniciarSesin = new JMenuItem("Iniciar sesi\u00F3n");

mntmIniciarSesin.setFont(new Font("Arial", Font.BOLD, 12));

mntmIniciarSesin.setForeground(Color.BLACK);

mntmIniciarSesin.addActionListener(this);

mntmIniciarSesin.setIcon(new ImageIcon("imagenes/usuario.png"));

mnArchivo.add(mntmIniciarSesin);

mntmSalir.setIcon(new ImageIcon(("imagenes/salir.png")));

mnArchivo.add(mntmSalir);

mnMantenimiento = new JMenu("Mantenimiento");

mnMantenimiento.setEnabled(false);

mnMantenimiento.setFont(new Font("Arial", Font.BOLD, 12));

menuProyecto.add(mnMantenimiento);

mntmEmpleados = new JMenuItem("Empleados");

mntmEmpleados.setFont(new Font("Arial", Font.BOLD, 12));

mntmEmpleados.setForeground(Color.BLACK);

mntmEmpleados.addActionListener(this);

mntmEmpleados.setIcon(new ImageIcon("imagenes/empleado.png"));

mnMantenimiento.add(mntmEmpleados);

mntmPacientes = new JMenuItem("Pacientes");

mntmPacientes.setFont(new Font("Arial", Font.BOLD, 12));

mntmPacientes.setForeground(Color.BLACK);

mntmPacientes.addActionListener(this);

mntmPacientes.setIcon(new ImageIcon("imagenes/paciente.png"));

mnMantenimiento.add(mntmPacientes);

mntmCamas = new JMenuItem("Camas");

mntmCamas.setFont(new Font("Arial", Font.BOLD, 12));

mntmCamas.setForeground(Color.BLACK);

mntmCamas.addActionListener(this);

mntmCamas.setIcon(new ImageIcon("imagenes/camas.png"));

mnMantenimiento.add(mntmCamas);

mntmMedicamentos = new JMenuItem("Medicamentos");

mntmMedicamentos.setFont(new Font("Arial", Font.BOLD, 12));

mntmMedicamentos.setForeground(Color.BLACK);

mntmMedicamentos.addActionListener(this);

mntmMedicamentos.setIcon(new ImageIcon("imagenes/medicamentos.png"));

mnMantenimiento.add(mntmMedicamentos);

mntmServicios = new JMenuItem("Servicios");

mntmServicios.setFont(new Font("Arial", Font.BOLD, 12));

mntmServicios.setForeground(Color.BLACK);

mntmServicios.addActionListener(this);

mntmServicios.setIcon(new ImageIcon("imagenes/servicios.png"));

mnMantenimiento.add(mntmServicios);

mnHospitalización = new JMenu("Hospitalizaci\u00F3n");

mnHospitalización.setEnabled(false);

mnHospitalización.setFont(new Font("Arial", Font.BOLD, 12));

mnHospitalización.addActionListener(this);

menuProyecto.add(mnHospitalización);

mntmRegistrar = new JMenuItem("Hospitalizar Paciente");

mntmRegistrar.setFont(new Font("Arial", Font.BOLD, 12));

mntmRegistrar.setForeground(Color.BLACK);

mntmRegistrar.addActionListener(this);

mntmRegistrar.setIcon(new ImageIcon("imagenes/hospitalizar.png"));

mnHospitalización.add(mntmRegistrar);

mnRegistroDeConsumo = new JMenu("Registro de Consumo");

mnRegistroDeConsumo.setEnabled(false);

mnRegistroDeConsumo.setFont(new Font("Arial", Font.BOLD, 12));

menuProyecto.add(mnRegistroDeConsumo);

mntmConsumo = new JMenuItem("Registrar Consumo de Pacientes");

mntmConsumo.setFont(new Font("Arial", Font.BOLD, 12));

mntmConsumo.setForeground(Color.BLACK);

mntmConsumo.addActionListener(this);

mntmConsumo.setIcon(new ImageIcon("imagenes/registroconsumo.png"));

mnRegistroDeConsumo.add(mntmConsumo);

mnPago = new JMenu("Pagos");

mnPago.setEnabled(false);

mnPago.setFont(new Font("Arial", Font.BOLD, 12));

mnPago.addActionListener(this);

menuProyecto.add(mnPago);

mntmRegistrarPago = new JMenuItem("Registrar Pago de Consumo");

mntmRegistrarPago.setFont(new Font("Arial", Font.BOLD, 12));

mntmRegistrarPago.setForeground(Color.BLACK);

mntmRegistrarPago.addActionListener(this);

mntmRegistrarPago.setIcon(new ImageIcon("imagenes/pagos.gif"));

mnPago.add(mntmRegistrarPago);

mnReporte = new JMenu("Reporte");

mnReporte.setEnabled(false);

mnReporte.setFont(new Font("Arial", Font.BOLD, 12));

menuProyecto.add(mnReporte);

mntmPacientesAdmitidos = new JMenuItem("Pacientes Admitidos");

mntmPacientesAdmitidos.setFont(new Font("Arial", Font.BOLD, 12));

mntmPacientesAdmitidos.setForeground(Color.BLACK);

mntmPacientesAdmitidos.addActionListener(this);

mntmPacientesAdmitidos.setIcon(new ImageIcon("imagenes/pacientesadmitidos.png"));

mnReporte.add(mntmPacientesAdmitidos);

mntmHospitalizacionesPorFecha = new JMenuItem("Hospitalizaciones Por Fecha");

mntmHospitalizacionesPorFecha.addActionListener(this);

mntmHospitalizacionesPorFecha.setForeground(Color.BLACK);

mntmHospitalizacionesPorFecha.setFont(new Font("Arial", Font.BOLD, 12));

mntmHospitalizacionesPorFecha.setIcon(new ImageIcon("imagenes/hospitaporfecha.png"));

mnReporte.add(mntmHospitalizacionesPorFecha);

mntmPacientespago = new JMenuItem("Pacientes/Pago");

mntmPacientespago.addActionListener(this);

mntmPacientespago.setForeground(Color.BLACK);

mntmPacientespago.setFont(new Font("Arial", Font.BOLD, 12));

mntmPacientespago.setIcon(new ImageIcon("imagenes/pagosrealizados.gif"));

mnReporte.add(mntmPacientespago);

mntmEmpledadoshospitalizaciones = new JMenuItem("Empledados/Hospitalizaciones");

mntmEmpledadoshospitalizaciones.addActionListener(this);

mntmEmpledadoshospitalizaciones.setForeground(Color.BLACK);

mntmEmpledadoshospitalizaciones.setFont(new Font("Arial", Font.BOLD, 12));

mntmEmpledadoshospitalizaciones.setIcon(new ImageIcon("imagenes/empleadosquehospitalizaron.png"));

mnReporte.add(mntmEmpledadoshospitalizaciones);

lblFondo = new JLabel(new ImageIcon("imagenes/MN-Global.png"));

lblFondo.setBounds(1, 0, ANCHO, ALTO);

contentPane.add(lblFondo);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == mntmEmpledadoshospitalizaciones) {

actionPerformedMntmEmpledadoshospitalizaciones(arg0);

}

if (arg0.getSource() == mntmPacientespago) {

actionPerformedMntmPacientespago(arg0);

}

if (arg0.getSource() == mntmHospitalizacionesPorFecha) {

actionPerformedMntmHospitalizacionesPorFecha(arg0);

}

if (arg0.getSource() == mntmPacientesAdmitidos) {

actionPerformedMntmPacientesAdmitidos(arg0);

}

if (arg0.getSource() == mntmRegistrarPago) {

actionPerformedMntmRegistrarPago(arg0);

}

if (arg0.getSource() == mntmIniciarSesin) {

actionPerformedMntmIniciarSesin(arg0);

}

if (arg0.getSource() == mntmRegistrar) {

actionPerformedMntmRegistrar(arg0);

}

if (arg0.getSource() == mntmConsumo) {

actionPerformedMntmConsumo(arg0);

}

if (arg0.getSource() == mntmServicios) {

actionPerformedMntmServicios(arg0);

}

if (arg0.getSource() == mntmMedicamentos) {

actionPerformedMntmMedicamentos(arg0);

}

if (arg0.getSource() == mntmCamas) {

actionPerformedMntmCamas(arg0);

}

if (arg0.getSource() == mntmPacientes) {

actionPerformedMntmPacientes(arg0);

}

if (arg0.getSource() == mntmEmpleados) {

actionPerformedMntmEmpleados(arg0);

}

if (arg0.getSource() == mntmSalir) {

actionPerformedMntmSalir(arg0);

}

}

protected void actionPerformedMntmSalir(ActionEvent arg0) {

int ok = confirmacionDeSalida();

if (ok == 0)

System.exit(0);

}

protected void actionPerformedMntmEmpleados(ActionEvent arg0) {

DlgEmpleado emp = new DlgEmpleado();

emp.setVisible(true);

}

protected void actionPerformedMntmPacientes(ActionEvent arg0) {

DlgPaciente pac = new DlgPaciente();

pac.setVisible(true);

}

protected void actionPerformedMntmCamas(ActionEvent arg0) {

DlgCamas cam = new DlgCamas();

cam.setVisible(true);

}

protected void actionPerformedMntmMedicamentos(ActionEvent arg0) {

DlgMedicamento med = new DlgMedicamento();

med.setVisible(true);

}

protected void actionPerformedMntmServicios(ActionEvent arg0) {

DlgServicios ser = new DlgServicios();

ser.setVisible(true);

}

protected void actionPerformedMntmConsumo(ActionEvent arg0) {

DlgRegistroConsumo con = new DlgRegistroConsumo();

con.setVisible(true);

}

protected void actionPerformedMntmRegistrar(ActionEvent arg0) {

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

ArregloCamas ac = new ArregloCamas("camas.txt");

DlgHospitalizacion hos = new DlgHospitalizacion();

hos.txtCodigoemple.setText("" + this.codigoEmpleado);

if (ac.tamaño() == 0) {

Libreria.mensajeAdvertenciaFrame(this,

"Antes de registrar una hospitalización debe registrar camas. Verificar.");

hos.dispose();

}

else if (ap.tamaño() == 0) {

Libreria.mensajeAdvertenciaFrame(this,

"Antes de registrar una hospitalización debe registrar pacientes. Verificar.");

hos.dispose();

}

else {

hos.setVisible(true);

}

}

protected void actionPerformedMntmIniciarSesin(ActionEvent arg0) {

DlgLogin log = new DlgLogin();

log.setVisible(true);

if (log.codigoEmpleado != -1) {

this.codigoEmpleado = log.codigoEmpleado;

if (log.tipoEmpleado == 0) {

activarmenu(true);

}else {

activarmenu(false);

}

}

}

protected void actionPerformedMntmRegistrarPago(ActionEvent arg0) {

DlgControlDePagos pag = new DlgControlDePagos();

pag.setVisible(true);

}

protected void actionPerformedMntmPacientesAdmitidos(ActionEvent arg0) {

DlgReportePacientesAdmitidos pac = new DlgReportePacientesAdmitidos();

pac.setVisible(true);

}

protected void actionPerformedMntmHospitalizacionesPorFecha(ActionEvent arg0) {

DlgReporteHospitalizacionesPorFecha hos = new DlgReporteHospitalizacionesPorFecha();

hos.setVisible(true);

}

protected void actionPerformedMntmPacientespago(ActionEvent arg0) {

DlgReportePacientesPagado pac = new DlgReportePacientesPagado();

pac.setVisible(true);

}

protected void actionPerformedMntmEmpledadoshospitalizaciones(ActionEvent arg0) {

DlgReporteEmpleadosHospitalizaciones emp = new DlgReporteEmpleadosHospitalizaciones();

emp.setVisible(true);

}

// métodos void con parámetros

void activarmenu(boolean dato) {

mnMantenimiento.setEnabled(dato);

mnHospitalización.setEnabled(true);

mnRegistroDeConsumo.setEnabled(true);

mnPago.setEnabled(true);

mnReporte.setEnabled(dato);

mntmIniciarSesin.setEnabled(false);

}

// MÉTODOS CON RETORNO SIN PARÁMETROS

public int confirmacionDeSalida() {

return JOptionPane.showConfirmDialog(this, "¿ Desea salir del programa ?", "Proyecto AED 2016 1", 2);

}

}

DlgCamas

package gui;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloCamas;

import clases.Camas;

import libreria.DiseñoObjetos;

import libreria.Validaciones;

import libreria.Libreria;

import javax.swing.JTable;

import java.awt.event.KeyListener;

import java.awt.event.KeyEvent;

import javax.swing.JRadioButton;

import javax.swing.ButtonGroup;

public class DlgCamas extends JDialog implements ActionListener, KeyListener {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JLabel lblNroCama;

private JLabel lblCategoria;

private JLabel lblPrecioPorDia;

private JTextField txtPrecioxDia;

private JScrollPane scrollPane;

private JButton btnBuscar;

private JButton btnAdicionar;

private JButton btnConsultar;

private JButton btnModificar;

private JButton btnEliminar;

private JButton btnAceptar;

private JButton btnCancelar;

private JTextField txtNroCama;

private JLabel lblEstado;

private JComboBox<String> cboCategoria;

private JComboBox<String> cboEstado;

private JScrollPane scpMedicamentos;

private JLabel lblMensaje;

private JButton btnGrabarCamas;

private JTable jtblCamas;

private ButtonGroup bp;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

/\*

\* Tipo de operación a procesar: Adicionar, Modificar, Eliminar o Consultar

\*/

private int tipoOperacion;

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// Declaración global

ArregloCamas ac = new ArregloCamas("camas.txt");

// Declaración global de libreria Validaciones

Validaciones v = new Validaciones();

// Declaración global de libreria DiseñoObjetos

DiseñoObjetos ds = new DiseñoObjetos();

// Declaración global de libreria

Libreria lb = new Libreria();

private JRadioButton rdbtnLibre;

private JRadioButton rdbtnOcupado;

private JRadioButton rdbtnTodo;

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgCamas dialog = new DlgCamas();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgCamas() {

setModal(true);

setResizable(false);

setTitle("MANTENIMIENTO | CAMAS");

setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

setBounds(100, 100, 783, 455);

contentPane = new JPanel();

contentPane.setBackground(Color.WHITE);

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

lblNroCama = new JLabel("NRO CAMA");

lblNroCama.setForeground(Color.BLACK);

lblNroCama.setHorizontalAlignment(SwingConstants.RIGHT);

lblNroCama.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblNroCama.setBounds(10, 59, 110, 14);

contentPane.add(lblNroCama);

lblCategoria = new JLabel("CATEGORIA");

lblCategoria.setForeground(Color.BLACK);

lblCategoria.setHorizontalAlignment(SwingConstants.RIGHT);

lblCategoria.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCategoria.setBounds(10, 99, 110, 14);

contentPane.add(lblCategoria);

lblPrecioPorDia = new JLabel("PRECIO POR D\u00CDA");

lblPrecioPorDia.setForeground(Color.BLACK);

lblPrecioPorDia.setHorizontalAlignment(SwingConstants.RIGHT);

lblPrecioPorDia.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblPrecioPorDia.setBounds(10, 137, 110, 14);

contentPane.add(lblPrecioPorDia);

lblEstado = new JLabel("ESTADO");

lblEstado.setForeground(Color.BLACK);

lblEstado.setHorizontalAlignment(SwingConstants.RIGHT);

lblEstado.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblEstado.setBounds(25, 181, 95, 14);

contentPane.add(lblEstado);

txtPrecioxDia = new JTextField();

txtPrecioxDia.addKeyListener(this);

txtPrecioxDia.setForeground(Color.BLACK);

txtPrecioxDia.setFont(new Font("Arial", Font.BOLD, 12));

txtPrecioxDia.setEditable(false);

txtPrecioxDia.setBounds(130, 130, 86, 29);

contentPane.add(txtPrecioxDia);

txtPrecioxDia.setColumns(10);

scrollPane = new JScrollPane();

scrollPane.setBounds(413, 219, -394, -54);

contentPane.add(scrollPane);

btnBuscar = new JButton("BUSCAR");

btnBuscar.addActionListener(this);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnBuscar.setEnabled(false);

btnBuscar.setBackground(new Color(102, 205, 170));

btnBuscar.setBounds(217, 52, 110, 29);

ds.setCurvasButton(btnBuscar, "imagenes/buscar.png");

contentPane.add(btnBuscar);

btnAdicionar = new JButton("ADICIONAR");

btnAdicionar.addActionListener(this);

btnAdicionar.setForeground(Color.BLACK);

btnAdicionar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAdicionar.setBackground(new Color(124, 252, 0));

btnAdicionar.setBounds(23, 227, 137, 39);

ds.setCurvasButton(btnAdicionar, "imagenes/adicionar.png");

contentPane.add(btnAdicionar);

btnConsultar = new JButton("CONSULTAR");

btnConsultar.addActionListener(this);

btnConsultar.setForeground(Color.BLACK);

btnConsultar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnConsultar.setBackground(Color.CYAN);

btnConsultar.setBounds(170, 227, 145, 39);

ds.setCurvasButton(btnConsultar, "imagenes/consultar.png");

contentPane.add(btnConsultar);

btnModificar = new JButton("MODIFICAR");

btnModificar.addActionListener(this);

btnModificar.setForeground(Color.BLACK);

btnModificar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnModificar.setBackground(new Color(0, 139, 139));

btnModificar.setBounds(170, 277, 145, 39);

ds.setCurvasButton(btnModificar, "imagenes/modificar.png");

contentPane.add(btnModificar);

btnEliminar = new JButton("ELIMINAR");

btnEliminar.addActionListener(this);

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnEliminar.setBackground(new Color(127, 255, 212));

btnEliminar.setBounds(23, 277, 137, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

contentPane.add(btnEliminar);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.addActionListener(this);

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAceptar.setEnabled(false);

btnAceptar.setBackground(new Color(30, 144, 255));

btnAceptar.setBounds(23, 327, 137, 39);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

contentPane.add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.addActionListener(this);

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnCancelar.setEnabled(false);

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setBounds(170, 327, 145, 39);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

contentPane.add(btnCancelar);

txtNroCama = new JTextField();

txtNroCama.addKeyListener(this);

txtNroCama.setForeground(Color.BLACK);

txtNroCama.setFont(new Font("Arial", Font.BOLD, 12));

txtNroCama.setEditable(false);

txtNroCama.setColumns(10);

txtNroCama.setBounds(130, 52, 70, 29);

contentPane.add(txtNroCama);

cboCategoria = new JComboBox<String>();

cboCategoria.setEnabled(false);

cboCategoria.setForeground(Color.BLACK);

cboCategoria.setFont(new Font("Arial", Font.BOLD, 12));

cboCategoria.setBounds(130, 92, 161, 29);

cboCategoria.addItem("ECONÓMICO");

cboCategoria.addItem("EJECUTIVO");

contentPane.add(cboCategoria);

cboEstado = new JComboBox<String>();

cboEstado.setEnabled(false);

cboEstado.setForeground(Color.BLACK);

cboEstado.setFont(new Font("Arial", Font.BOLD, 12));

cboEstado.setBounds(130, 174, 161, 29);

cboEstado.addItem("LIBRE");

cboEstado.addItem("OCUPADO");

contentPane.add(cboEstado);

scpMedicamentos = new JScrollPane();

scpMedicamentos.setBounds(336, 43, 430, 373);

contentPane.add(scpMedicamentos);

dtm = new DefaultTableModel(null, getColumnas());

jtblCamas = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblCamas.setFont(new Font("Arial", Font.BOLD, 14));

jtblCamas.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblCamas.setRowHeight(25);

scpMedicamentos.setViewportView(jtblCamas);

lblMensaje = new JLabel("");

lblMensaje.setOpaque(true);

lblMensaje.setHorizontalAlignment(SwingConstants.CENTER);

lblMensaje.setForeground(Color.WHITE);

lblMensaje.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 18));

lblMensaje.setBackground(Color.DARK\_GRAY);

lblMensaje.setBounds(10, 12, 316, 29);

contentPane.add(lblMensaje);

btnGrabarCamas = new JButton("GRABAR CAMAS");

btnGrabarCamas.addActionListener(this);

btnGrabarCamas.setBounds(25, 377, 290, 39);

btnGrabarCamas.setForeground(Color.BLACK);

btnGrabarCamas.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnGrabarCamas.setBackground(new Color(0, 250, 154));

ds.setCurvasButton(btnGrabarCamas, "imagenes/grabar.png");

contentPane.add(btnGrabarCamas);

rdbtnLibre = new JRadioButton("LIBRE");

rdbtnLibre.addActionListener(this);

rdbtnLibre.setSelected(true);

rdbtnLibre.setForeground(Color.BLACK);

rdbtnLibre.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnLibre.setBackground(Color.WHITE);

rdbtnLibre.setBounds(427, 12, 70, 23);

contentPane.add(rdbtnLibre);

rdbtnOcupado = new JRadioButton("OCUPADO");

rdbtnOcupado.addActionListener(this);

rdbtnOcupado.setForeground(Color.BLACK);

rdbtnOcupado.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnOcupado.setBackground(Color.WHITE);

rdbtnOcupado.setBounds(499, 12, 103, 23);

contentPane.add(rdbtnOcupado);

rdbtnTodo = new JRadioButton("TODO");

rdbtnTodo.addActionListener(this);

rdbtnTodo.setForeground(Color.BLACK);

rdbtnTodo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnTodo.setBackground(Color.WHITE);

rdbtnTodo.setBounds(604, 12, 70, 23);

contentPane.add(rdbtnTodo);

bp = new ButtonGroup();

bp.add(rdbtnLibre);

bp.add(rdbtnOcupado);

bp.add(rdbtnTodo);

modeloTabla();

listadoLibre();

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == rdbtnTodo) {

actionPerformedRdbtnTodo(arg0);

}

if (arg0.getSource() == rdbtnOcupado) {

actionPerformedRdbtnOcupado(arg0);

}

if (arg0.getSource() == rdbtnLibre) {

actionPerformedRdbtnLibre(arg0);

}

if (arg0.getSource() == btnGrabarCamas) {

actionPerformedBtnGrabarCamas(arg0);

}

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

if (arg0.getSource() == btnCancelar) {

actionPerformedBtnCancelar(arg0);

}

if (arg0.getSource() == btnAceptar) {

actionPerformedBtnAceptar(arg0);

}

if (arg0.getSource() == btnModificar) {

actionPerformedBtnModificar(arg0);

}

if (arg0.getSource() == btnEliminar) {

actionPerformedBtnEliminar(arg0);

}

if (arg0.getSource() == btnConsultar) {

actionPerformedBtnConsultar(arg0);

}

if (arg0.getSource() == btnAdicionar) {

actionPerformedBtnAdicionar(arg0);

}

}

protected void actionPerformedRdbtnLibre(ActionEvent arg0) {

listadoLibre();

}

protected void actionPerformedRdbtnOcupado(ActionEvent arg0) {

listadoOcupado();

}

protected void actionPerformedRdbtnTodo(ActionEvent arg0) {

listado();

}

protected void actionPerformedBtnAdicionar(ActionEvent arg0) {

tipoOperacion = ADICIONAR;

lblMensaje.setText("Adicionando camas");

habilitarEntradas(true);

habilitarOperaciones(false);

txtNroCama.requestFocus();

}

protected void actionPerformedBtnConsultar(ActionEvent arg0) {

tipoOperacion = CONSULTAR;

lblMensaje.setText("Consultando camas");

habilitarBusqueda(true);

habilitarOperaciones(false);

cboEstado.setEnabled(false);

}

protected void actionPerformedBtnModificar(ActionEvent arg0) {

tipoOperacion = MODIFICAR;

lblMensaje.setText("Modificando camas");

habilitarBusqueda(true);

habilitarEntradas(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

tipoOperacion = ELIMINAR;

lblMensaje.setText("Eliminando camas");

habilitarBusqueda(true);

habilitarOperaciones(false);

cboEstado.setEnabled(false);

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

consultarCama();

}

protected void actionPerformedBtnAceptar(ActionEvent arg0) {

switch (tipoOperacion) {

case ADICIONAR:

adicionarCama();

break;

case CONSULTAR:

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

break;

case MODIFICAR:

modificarCama();

break;

case ELIMINAR:

eliminarCama();

}

}

protected void actionPerformedBtnCancelar(ActionEvent arg0) {

if (tipoOperacion != ADICIONAR)

habilitarBusqueda(false);

lblMensaje.setText("");

habilitarEntradas(false);

habilitarOperaciones(true);

}

protected void actionPerformedBtnGrabarCamas(ActionEvent arg0) {

if (ac.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + ac.getArchivo() + "\" ?");

if (ok == 0) {

ac.grabarCamas();

Libreria.mensajeInformacion(this, "\"" + ac.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + ac.getArchivo() + "\"");

}

else {

ac.grabarCamas();

Libreria.confirmacion(this, "\"" + ac.getArchivo() + "\" ha sido creado");

}

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

}

public void keyTyped(KeyEvent arg0) {

if (arg0.getSource() == txtPrecioxDia) {

keyTypedTxtPrecioxDia(arg0);

}

if (arg0.getSource() == txtNroCama) {

keyTypedTxtNroCama(arg0);

}

}

protected void keyTypedTxtNroCama(KeyEvent e) {

Validaciones.soloNumero(e, txtNroCama, 5);

char t = e.getKeyChar();

if (t == KeyEvent.VK\_ENTER && tipoOperacion != ADICIONAR)

consultarCama();

}

protected void keyTypedTxtPrecioxDia(KeyEvent e) {

Validaciones.numeroReal(txtPrecioxDia, e, 5);

}

// Métodos tipo void sin parámetros

void modeloTabla() {

TableColumnModel tc = jtblCamas.getColumnModel();

tc.getColumn(0).setPreferredWidth(100);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(1).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(2).setCellRenderer(tcr);

}

void listado() {

Libreria.limpiarJTable(jtblCamas, dtm);

for (int i = 0; i < ac.tamaño(); i++) {

Camas x = ac.obtener(i);

dtm.addRow(new Object[] { x.getNroCama(), getCategoria(x.getCategoria()),

String.format("%.2f", x.getPrecioxDia()), getEstado(x.getEstado()) });

}

}

void listadoLibre() {

Libreria.limpiarJTable(jtblCamas, dtm);

for (int i = 0; i < ac.tamaño(); i++) {

Camas x = ac.obtener(i);

if (x.getEstado() == 0) {

dtm.addRow(new Object[] { x.getNroCama(), getCategoria(x.getCategoria()),

String.format("%.2f", x.getPrecioxDia()), getEstado(x.getEstado()) });

}

}

}

void listadoOcupado() {

Libreria.limpiarJTable(jtblCamas, dtm);

for (int i = 0; i < ac.tamaño(); i++) {

Camas x = ac.obtener(i);

if (x.getEstado() == 1) {

dtm.addRow(new Object[] { x.getNroCama(), getCategoria(x.getCategoria()),

String.format("%.2f", x.getPrecioxDia()), getEstado(x.getEstado()) });

}

}

}

void actualizarTabla() {

if (rdbtnLibre.isSelected()) {

listadoLibre();

}

else if (rdbtnOcupado.isSelected()) {

listadoOcupado();

}

else {

listado();

}

}

void limpieza() {

txtNroCama.setText("");

txtPrecioxDia.setText("");

cboCategoria.setSelectedIndex(0);

cboEstado.setSelectedIndex(0);

}

void adicionarCama() {

if (leerNroCama() < 0) {

Libreria.mensajeAdvertencia(this, "No ingresó el NRO DE CAMA");

txtNroCama.requestFocus();

}

else if (leerPrecioxDia() < 0.0) {

Libreria.mensajeAdvertencia(this, "No ingresó el PRECIO POR DÍA");

txtPrecioxDia.setText("");

txtPrecioxDia.requestFocus();

}

else {

int c = 0;

for (int i = 0; i < ac.tamaño(); i++) {

if (ac.obtener(i).getNroCama() == leerNroCama()) {

c++;

}

}

if (c > 0) {

Libreria.mensajeAdvertencia(this, "El NRO DE CAMA ya existe");

txtNroCama.setText("");

txtNroCama.requestFocus();

}

else {

Camas nuevo = new Camas(leerNroCama(), leerCategoria(), leerPrecioxDia(), leerEstado());

ac.adicionar(nuevo);

lblMensaje.setText("");

actualizarTabla();

habilitarEntradas(false);

habilitarOperaciones(true);

}

}

}

void consultarCama() {

int nro = leerNroCama();

if (nro == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el NRO DE CAMA");

txtNroCama.requestFocus();

}

else {

Camas x = ac.buscar(leerNroCama());

if (x != null) {

txtNroCama.setText("" + x.getNroCama());

txtPrecioxDia.setText("" + x.getPrecioxDia());

cboCategoria.setSelectedIndex(x.getCategoria());

cboEstado.setSelectedIndex(x.getEstado());

}

else {

Libreria.mensajeAdvertencia(this, "El número " + leerNroCama() + " no existe");

limpieza();

lblMensaje.setText("");

txtNroCama.setText("");

txtNroCama.requestFocus();

}

}

}

void modificarCama() {

int nro = leerNroCama();

if (nro == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el NRO DE CAMA");

txtNroCama.requestFocus();

}

else {

Camas x = ac.buscar(leerNroCama());

if (x != null)

if (leerNroCama() < 0) {

Libreria.mensajeAdvertencia(this, "No ingresó el NRO DE CAMA");

txtNroCama.requestFocus();

}

else if (leerPrecioxDia() < 0.0) {

Libreria.mensajeAdvertencia(this, "No ingresó el PRECIO POR DÍA");

txtPrecioxDia.setText("");

txtPrecioxDia.requestFocus();

}

else {

lblMensaje.setText("");

x.setCategoria(leerCategoria());

x.setPrecioxDia(leerPrecioxDia());

x.setEstado(leerEstado());

actualizarTabla();

habilitarBusqueda(false);

habilitarEntradas(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El NRO " + leerNroCama() + " no existe");

txtNroCama.setText("");

txtNroCama.requestFocus();

}

}

}

void eliminarCama() {

int nro = leerNroCama();

if (nro == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el NRO DE CAMA");

txtNroCama.requestFocus();

}

else {

Camas x = ac.buscar(leerNroCama());

if (x != null) {

lblMensaje.setText("");

ac.eliminar(x);

actualizarTabla();

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El NRO " + leerNroCama() + " no existe");

txtNroCama.setText("");

txtNroCama.requestFocus();

}

}

}

// Métodos tipo void con parámetros

void habilitarBusqueda(boolean sino) {

btnBuscar.setEnabled(sino);

txtNroCama.setEditable(sino);

if (sino)

txtNroCama.requestFocus();

}

void habilitarEntradas(boolean sino) {

if (!sino)

limpieza();

txtNroCama.setEditable(sino);

txtPrecioxDia.setEditable(sino);

cboCategoria.setEnabled(sino);

}

void habilitarOperaciones(boolean sino) {

btnAdicionar.setEnabled(sino);

btnConsultar.setEnabled(sino);

btnModificar.setEnabled(sino);

btnEliminar.setEnabled(sino);

btnAceptar.setEnabled(!sino);

btnCancelar.setEnabled(!sino);

}

// Métodos que retornan valor sin parámetros

String[] getColumnas() {

String columnas[] = new String[] { "NRO CAMA", "CATEGORIA", "PRECIO POR DÍA", "ESTADO" };

return columnas;

}

int leerNroCama() {

return Libreria.leerEntero(txtNroCama);

}

int leerCategoria() {

return cboCategoria.getSelectedIndex();

}

double leerPrecioxDia() {

return Libreria.leerReal(txtPrecioxDia);

}

int leerEstado() {

return cboEstado.getSelectedIndex();

}

// Métodos que retornan valor con parámetros

String getEstado(int est) {

if (est == 0) {

return "LIBRE";

}

else {

return "OCUPADO";

}

}

String getCategoria(int est) {

if (est == 0) {

return "ECONÓMICO";

}

else {

return "EJECUTIVO";

}

}

}

DlgControlDePagos

package gui;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import arreglos.ArregloCamas;

import arreglos.ArregloConsumo;

import arreglos.ArregloDetalleConsumo;

import arreglos.ArregloHospitalizacion;

import arreglos.ArregloMedicamento;

import arreglos.ArregloPaciente;

import arreglos.ArregloServicio;

import clases.Camas;

import clases.Consumo;

import clases.DetalleConsumo;

import clases.Hospitalizacion;

import clases.Medicamentos;

import clases.Paciente;

import clases.Servicio;

import libreria.DiseñoObjetos;

import libreria.Fecha;

import libreria.Libreria;

import javax.swing.JPanel;

public class DlgControlDePagos extends JDialog implements ActionListener {

private static final long serialVersionUID = 1L;

private JLabel lblCdigoPaciente;

private JTextField txtCodigoPaciente;

private JButton btnBuscarPaciente;

private JScrollPane scrollPane;

private JTextArea txtS;

private JButton btnRegistrarPago;

private JComboBox<String> cboDia;

private JComboBox<String> cboMes;

private JComboBox<String> cboAno;

private JButton btnVerFactura;

// VARIABLE GLOBAL PARA OBTENER LA FECHA ACTUAL

private int fecha;

// VARIABLE GLOBAL PARA OBTENER LA HORA ACTUAL

private int hora;

// VARIABLE GLOBAL PARA ASIGNAR EL TOTAL DE IMPORTE QUE PAGARÀ EL CLIENTE

private double totalVenta = 0.0;

// VARIABLES GLOBALES PARA GUARDAR FACTURA

private int nroCama = 0;

private ArrayList<Integer> codDet;

private int codHospitalizacion = 0;

int cantDias = 0;

double precioxDia = 0.0;

// DECLARACIÓN GLOBAL DE ARREGLOS

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

ArregloCamas ac = new ArregloCamas("camas.txt");

ArregloConsumo acon = new ArregloConsumo("consumos.txt");

ArregloDetalleConsumo adet = new ArregloDetalleConsumo("detallesconsumos.txt");

ArregloMedicamento am = new ArregloMedicamento("medicamentos.txt");

ArregloServicio as = new ArregloServicio("servicios.txt");

// DECLARACIÓN GLOBAL DE LIBRERIA

DiseñoObjetos ds = new DiseñoObjetos();

private JPanel panel;

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Exception e) {

e.printStackTrace();

}

try {

DlgControlDePagos dialog = new DlgControlDePagos();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgControlDePagos() {

getContentPane().setBackground(Color.WHITE);

setModal(true);

setResizable(false);

setTitle("CONTROL DE PAGOS");

setBounds(100, 100, 601, 622);

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 127, 576, 456);

getContentPane().add(scrollPane);

scrollPane.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3, true));

txtS = new JTextArea();

txtS.setEditable(false);

scrollPane.setViewportView(txtS);

panel = new JPanel();

panel.setBackground(Color.DARK\_GRAY);

panel.setBounds(10, 11, 575, 105);

getContentPane().add(panel);

panel.setLayout(null);

panel.setBorder(BorderFactory.createLineBorder(Color.red, 3, false));

lblCdigoPaciente = new JLabel("C\u00D3DIGO PACIENTE ");

lblCdigoPaciente.setBounds(0, 18, 129, 14);

panel.add(lblCdigoPaciente);

lblCdigoPaciente.setForeground(Color.WHITE);

lblCdigoPaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCdigoPaciente.setHorizontalAlignment(SwingConstants.RIGHT);

txtCodigoPaciente = new JTextField();

txtCodigoPaciente.setBounds(130, 11, 105, 29);

panel.add(txtCodigoPaciente);

txtCodigoPaciente.setBackground(Color.WHITE);

txtCodigoPaciente.setHorizontalAlignment(SwingConstants.CENTER);

txtCodigoPaciente.setEditable(false);

txtCodigoPaciente.setForeground(Color.BLACK);

txtCodigoPaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigoPaciente.setColumns(10);

btnBuscarPaciente = new JButton("");

btnBuscarPaciente.setBounds(245, 11, 38, 29);

panel.add(btnBuscarPaciente);

btnBuscarPaciente.setBackground(new Color(102, 205, 170));

btnBuscarPaciente.addActionListener(this);

btnBuscarPaciente.setForeground(Color.BLACK);

btnBuscarPaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

ds.setCurvasButton(btnBuscarPaciente, "imagenes/buscar.png");

cboDia = new JComboBox<String>();

cboDia.setBounds(309, 11, 59, 29);

panel.add(cboDia);

cboDia.setForeground(Color.BLACK);

cboDia.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboDia.setEnabled(false);

Fecha.colocarItems(cboDia, 1, 31);

Fecha.colocarDiaActual(cboDia);

cboMes = new JComboBox<String>();

cboMes.setBounds(369, 11, 116, 29);

panel.add(cboMes);

cboMes.setForeground(Color.BLACK);

cboMes.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboMes.setEnabled(false);

Fecha.colocarMeses(cboMes);

Fecha.colocarMesActual(cboMes);

cboAno = new JComboBox<String>();

cboAno.setBounds(486, 11, 79, 29);

panel.add(cboAno);

cboAno.setForeground(Color.BLACK);

cboAno.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboAno.setEnabled(false);

cboAno.addItem("" + Fecha.añoActual());

fecha = Fecha.getFecha(cboDia, cboMes, cboAno);

btnRegistrarPago = new JButton("REGISTRAR PAGO");

btnRegistrarPago.setBounds(275, 54, 176, 40);

panel.add(btnRegistrarPago);

btnRegistrarPago.addActionListener(this);

btnRegistrarPago.setBackground(Color.CYAN);

btnRegistrarPago.setForeground(Color.BLACK);

btnRegistrarPago.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

ds.setCurvasButton(btnRegistrarPago, "imagenes/aceptar.png");

btnVerFactura = new JButton("VER FACTURA");

btnVerFactura.setBounds(103, 54, 150, 40);

panel.add(btnVerFactura);

btnVerFactura.addActionListener(this);

btnVerFactura.setForeground(Color.BLACK);

btnVerFactura.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnVerFactura.setBackground(Color.GREEN);

ds.setCurvasButton(btnVerFactura, "imagenes/listado.png");

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnVerFactura) {

actionPerformedBtnVerFactura(arg0);

}

if (arg0.getSource() == btnRegistrarPago) {

actionPerformedBtnRegistrarPago(arg0);

}

if (arg0.getSource() == btnBuscarPaciente) {

actionPerformedBtnBuscarPaciente(arg0);

}

}

protected void actionPerformedBtnBuscarPaciente(ActionEvent arg0) {

DlgHospitalizacion hos = new DlgHospitalizacion();

hos.setTitle("PACIENTES HOSPITALIZADOS");

hos.btnBuscarPaciente.setEnabled(false);

hos.rdbtnEconomico.setEnabled(false);

hos.rdbtnEjecutivo.setEnabled(false);

hos.cboNroCamas.setEnabled(false);

hos.btnAceptar.setEnabled(false);

hos.btnCancelar.setEnabled(false);

hos.btnGrabarHospitalizacion.setEnabled(false);

hos.rdbtnFiltrarPagado.setEnabled(false);

hos.rdbtnFiltrarTodo.setEnabled(false);

hos.setVisible(true);

if (hos.codigoPaciente.equals("")) {

txtS.setText("");

txtCodigoPaciente.setText("");

}

else {

txtCodigoPaciente.setText(hos.codigoPaciente);

txtS.setText("");

}

}

protected void actionPerformedBtnRegistrarPago(ActionEvent arg0) {

if (!txtS.getText().trim().equals("") && leerCodigoPaciente() != -1) {

int msj = Libreria.confirmacion(this, "¿DESEA GUARDAR EL REGISTRO DE FACTURA?");

if (msj == 0) {

guardarFactura();

nuevaFactura();

}

else {

Libreria.mensajeInformacion(this, "EL REGISTRO NO SE GUARDÓ");

}

}

}

protected void actionPerformedBtnVerFactura(ActionEvent arg0) {

if (leerCodigoPaciente() != -1) {

hora = Fecha.obtenerHoraActual();

verFactura();

}

}

void guardarFactura() {

actualizarCamas();

ac.grabarCamas();

Hospitalizacion h;

for (int i = 0; i < ah.tamaño(); i++) {

h = ah.obtener(i);

if (h.getCodHosp() == codHospitalizacion) {

h.setFecSali(fecha);

h.setHorSali(hora);

h.setEstado(2);

}

}

ah.grabarHospitalizacion();

if (!codDet.isEmpty()) {

for (int j = 0; j < codDet.size(); j++) {

Consumo con;

for (int i = 0; i < acon.tamaño(); i++) {

con = acon.obtener(i);

if (con.getEstado() == 0) {

if (con.getCodConsumo() == codDet.get(j)) {

con.setEstado(1);

}

}

}

}

Consumo con;

for (int l = 0; l < acon.tamaño(); l++) {

con = acon.obtener(l);

if (con.getCodConsumo() == codDet.get(codDet.size() - 1)) {

con.setTotalPago(con.getTotalPago() + (cantDias \* precioxDia));

}

}

adet.adicionar(new DetalleConsumo(codDet.get(codDet.size() - 1), nroCama, cantDias, precioxDia,

cantDias \* precioxDia));

}

else {

acon.adicionar(

new Consumo(acon.codigoCorrelativo(), leerCodigoPaciente(), fecha, 1, cantDias \* precioxDia));

adet.adicionar(new DetalleConsumo(acon.obtener(acon.tamaño() - 1).getCodConsumo(), nroCama, cantDias,

precioxDia, cantDias \* precioxDia));

}

acon.grabarConsumo();

adet.grabarDetalleConsumo();

am.grabarMedicamentos();

as.grabarServicios();

}

void verFactura() {

txtS.setText("");

imprimir(String.format("%85s", "HOSPITAL DE LA SOLIDARIDAD"));

imprimir(String.format("%85s", "RUC" + String.format("%11d", 1040501050)));

imprimir(String.format("%90s", "Av. Angamos Este 734 · (01) 2431120"));

imprimir(String.format("%85s", "BOLETA ELECTRÓNICA"));

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

imprimir(String.format("%65s", "FECHA EMISIÓN\t:\t" + Fecha.dd\_mm\_aaaa(fecha)));

imprimir(String.format("%62s", "HORA EMISIÓN\t:\t" + Fecha.hh\_mm\_ss(hora)));

datosPaciente();

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

// OBTENIENDO LOS CODIGOS DE CONSUMO DONDE HA COMPRADO EL PACIENTE

Consumo con;

codDet = new ArrayList<Integer>();

for (int i = 0; i < acon.tamaño(); i++) {

con = acon.obtener(i);

if (con.getEstado() == 0) {

if (con.getCodPaciente() == leerCodigoPaciente()) {

codDet.add(con.getCodConsumo());

}

}

}

// LISTANDO PRODUCTOS COMPRADOS POR EL PACIENTE

double total = 0.0;

if (codDet.size() > 0 && !codDet.isEmpty()) {

imprimir(String.format("%82s", "MEDICAMENTOS COMPRADOS"));

imprimir("\tPRODUCTO\tPRECIO\tCANTIDAD\tSUBTOTAL");

for (int i = 0; i < codDet.size(); i++) {

total += listadoProductoAdquiridos(codDet.get(i));

}

imprimir("\t\t\t SUBTOTAL\t " + String.format("%.2f", total));

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

}

// MOSTRANDO PRODUCTOS ADQUIRIDOS POR EL PACIENTE

if (codDet.size() > 0 && !codDet.isEmpty()) {

total = 0;

imprimir(String.format("%85s", "SERVICIOS COMPRADOS"));

imprimir("\tSERVICIO\tPRECIO\tCANTIDAD\tSUBTOTAL");

for (int i = 0; i < codDet.size(); i++) {

total += listadoServicios(codDet.get(i));

}

imprimir("\t\t\t SUBTOTAL\t " + String.format("%.2f", total));

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

}

// OBTENIENDO EL NÚMERO DE CAMA DEL PACIENTE Y LA CANTIDAD DE DIAS

// HOSPITALIZADO

// Y LOS DATOS DE LA HOSPITALIZACION DEL PACIENTE

Hospitalizacion h;

for (int i = 0; i < ah.tamaño(); i++) {

h = ah.obtener(i);

if (h.getEstado() == 1) {

if (h.getCodPaci() == leerCodigoPaciente()) {

codHospitalizacion = h.getCodHosp();

cantDias = cantidadDeDias(h.getFecLleg());

nroCama = h.getNumCama();

}

}

}

// OBTENIENDO DATOS DE LA CAMA DONDE SE HOSPITALIZÓ EL PACIENTE

int categoriaCama = 0;

Camas c;

for (int i = 0; i < ac.tamaño(); i++) {

c = ac.obtener(i);

if (c.getNroCama() == nroCama) {

precioxDia = c.getPrecioxDia();

categoriaCama = c.getCategoria();

}

}

// MOSTRANDO DE LA CAMA DONDE SE HOSPITALIZÓ

imprimir(String.format("%85s", "HOSPITALIZACIÓN EN CAMA"));

if (cantDias == 0) {

imprimir("\t CATERORIA CAMA\t:\t" + getCategoria(categoriaCama).toUpperCase());

imprimir("\t NRO CAMA\t\t:\t" + nroCama);

imprimir("\t PRECIO POR DÍA\t:\t" + String.format("%.2f", precioxDia));

imprimir("\t CANTIDAD DE DÍAS\t:\t" + 1);

imprimir("\t\t\t SUBTOTAL\t " + String.format("%.2f", precioxDia));

totalVenta += precioxDia;

}

else {

imprimir("\t CATERORIA CAMA\t:\t" + getCategoria(categoriaCama).toUpperCase());

imprimir("\t NRO CAMA\t\t:\t" + nroCama);

imprimir("\t PRECIO POR DÍA\t:\t" + String.format("%.2f", precioxDia));

imprimir("\t CANTIDAD DE DÍAS\t:\t" + cantDias);

imprimir("\t\t\t SUBTOTAL\t " + String.format("%.2f", precioxDia \* cantDias));

totalVenta += (precioxDia \* cantDias);

}

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

imprimir("\t\t\t TOTAL\t" + String.format("%.2f", totalVenta));

totalVenta = 0.0;

imprimir("\t ----------------------------------------------------------------");

imprimir("\t ----------------------------------------------------------------");

imprimir(String.format("%85s", "NO HAY DEVOLUCIÓN DE DINERO."));

imprimir(String.format("%80s", "TODO CAMBIO DE PRODUCTO SE HARÁ DENTRO"));

imprimir(String.format("%82s", "DE LAS 48 HORAS PREVIA PRESENTACIÓN DEL"));

imprimir(String.format("%80s", "COMPROBANTE Y VERIFICACIÓN POR PARTE DEL"));

imprimir(String.format("%85s", "ADMINISTRADOR"));

imprimir("");

imprimir(String.format("%83s", "WWW.SOLIDARIDAD.COM"));

imprimir(String.format("%83s", "GRACIAS POR SU COMPRA"));

}

void nuevaFactura() {

txtCodigoPaciente.setText("");

totalVenta = 0.0;

txtS.setText("");

Fecha.colocarDiaActual(cboDia);

Fecha.colocarMesActual(cboMes);

}

void actualizarCamas() {

Camas c;

for (int j = 0; j < ac.tamaño(); j++) {

c = ac.obtener(j);

if (c.getNroCama() == nroCama)

c.setEstado(0);

}

}

double listadoProductoAdquiridos(int codConsumo) {

DetalleConsumo det;

Medicamentos med;

int codProducto;

double total = 0.0;

for (int i = 0; i < adet.tamaño(); i++) {

det = adet.obtener(i);

if (codConsumo == det.getCodConsumo()) {

codProducto = det.getCodProducto();

for (int j = 0; j < am.tamaño(); j++) {

med = am.obtener(j);

if (codProducto == med.getCodMedicamento()) {

imprimir("\t" + med.getNombre() + "\t" + String.format("%.2f", med.getPrecio()) + "\t "

+ det.getCantidad() + "\t " + String.format("%.2f", det.getSubtotal()));

total += det.getSubtotal();

}

}

}

}

totalVenta += total;

return total;

}

double listadoServicios(int codConsumo) {

DetalleConsumo det;

Servicio ser;

int codProducto;

double total = 0.0;

for (int i = 0; i < adet.tamaño(); i++) {

det = adet.obtener(i);

if (codConsumo == det.getCodConsumo()) {

codProducto = det.getCodProducto();

for (int j = 0; j < as.tamaño(); j++) {

ser = as.obtener(j);

if (codProducto == ser.getCodServicio()) {

imprimir("\t" + ser.getDescripcion() + "\t" + String.format("%.2f", ser.getPrecio()) + "\t "

+ det.getCantidad() + "\t " + String.format("%.2f", det.getSubtotal()));

total += det.getSubtotal();

}

}

}

}

totalVenta += total;

return total;

}

void datosPaciente() {

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

Paciente p;

for (int i = 0; i < ap.tamaño(); i++) {

p = ap.obtener(i);

if (p.getCodpaciente() == leerCodigoPaciente()) {

imprimir("\t NOMBRE\t\t:\t" + p.getApellidos().toUpperCase() + " "

+ p.getNombres().toUpperCase());

imprimir(String.format("%54s", "DNI\t\t:\t" + p.getDNI()));

}

}

}

String getCategoria(int est) {

if (est == 0) {

return "ECONÓMICO";

}

else {

return "EJECUTIVO";

}

}

int cantidadDeDias(int feclleg) {

int cantidadDias = Fecha.CantidadDiasEntre2fechas(Fecha.obtenerAñoDeFecha(feclleg),

Fecha.obtenerMesDeFecha(feclleg), Fecha.obtenerDiaDeFecha(feclleg), Fecha.obtenerAñoDeFecha(fecha),

Fecha.obtenerMesDeFecha(fecha), Fecha.obtenerDiaDeFecha(fecha));

return cantidadDias;

}

int leerCodigoPaciente() {

return Libreria.leerEntero(txtCodigoPaciente);

}

void imprimir(String dato) {

txtS.append(dato + "\n");

}

}

DlgEmpleados

package gui;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloEmpleado;

import clases.Empleado;

import libreria.DiseñoObjetos;

import libreria.Validaciones;

import libreria.Libreria;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.JComboBox;

import javax.swing.JButton;

import java.awt.Font;

import java.awt.Color;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JTable;

import java.awt.event.KeyListener;

import java.awt.event.KeyEvent;

public class DlgEmpleado extends JDialog implements ActionListener, KeyListener {

private static final long serialVersionUID = 1L;

private JLabel lblCodigo;

private JLabel lblApellidos;

private JLabel lblNombres;

private JLabel lblTipo;

private JLabel lblLogin;

private JLabel lblPassword;

private JLabel lblTurno;

private JTextField txtApellidos;

private JTextField txtNombres;

private JComboBox<String> cboTipo;

private JTextField txtLogin;

private JTextField txtPassword;

private JComboBox<String> cboTurno;

private JButton btnAdicionar;

private JButton btnConsultar;

private JButton btnModificar;

private JButton btnEliminar;

private JButton btnAceptar;

private JButton btnCancelar;

private JButton btnBuscar;

private JLabel lblMensaje;

private JScrollPane scrollPane;

private JTextField txtCodigo;

private JButton btnGuardar;

private JTable jtblEmpleados;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

/\*

\* Tipo de operación a procesar: Adicionar, Modificar, Eliminar o Consultar

\*/

private int tipoOperacion;

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// Declaración global

ArregloEmpleado ae = new ArregloEmpleado("empleados.txt");

// Declaración global de librerias

Validaciones v = new Validaciones();

DiseñoObjetos ds = new DiseñoObjetos();

Libreria lb = new Libreria();

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgEmpleado dialog = new DlgEmpleado();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgEmpleado() {

getContentPane().setBackground(Color.WHITE);

setResizable(false);

setModal(true);

setTitle("MANTENIMIENTO | EMPLEADOS");

setBounds(100, 100, 1257, 571);

getContentPane().setLayout(null);

lblCodigo = new JLabel("CODIGO");

lblCodigo.setForeground(Color.BLACK);

lblCodigo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblCodigo.setHorizontalAlignment(SwingConstants.RIGHT);

lblCodigo.setBounds(14, 60, 71, 14);

getContentPane().add(lblCodigo);

lblApellidos = new JLabel("APELLIDOS");

lblApellidos.setForeground(Color.BLACK);

lblApellidos.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblApellidos.setHorizontalAlignment(SwingConstants.RIGHT);

lblApellidos.setBounds(0, 101, 85, 14);

getContentPane().add(lblApellidos);

lblNombres = new JLabel("NOMBRES");

lblNombres.setForeground(Color.BLACK);

lblNombres.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblNombres.setHorizontalAlignment(SwingConstants.RIGHT);

lblNombres.setBounds(0, 141, 85, 14);

getContentPane().add(lblNombres);

lblTipo = new JLabel("TIPO");

lblTipo.setForeground(Color.BLACK);

lblTipo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblTipo.setHorizontalAlignment(SwingConstants.RIGHT);

lblTipo.setBounds(0, 181, 85, 14);

getContentPane().add(lblTipo);

lblLogin = new JLabel("LOGIN");

lblLogin.setForeground(Color.BLACK);

lblLogin.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblLogin.setHorizontalAlignment(SwingConstants.RIGHT);

lblLogin.setBounds(39, 221, 46, 14);

getContentPane().add(lblLogin);

lblPassword = new JLabel("PASSWORD");

lblPassword.setForeground(Color.BLACK);

lblPassword.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblPassword.setHorizontalAlignment(SwingConstants.RIGHT);

lblPassword.setBounds(0, 261, 85, 14);

getContentPane().add(lblPassword);

lblTurno = new JLabel("TURNO");

lblTurno.setForeground(Color.BLACK);

lblTurno.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblTurno.setHorizontalAlignment(SwingConstants.RIGHT);

lblTurno.setBounds(20, 301, 65, 14);

getContentPane().add(lblTurno);

txtApellidos = new JTextField();

txtApellidos.addKeyListener(this);

txtApellidos.setEditable(false);

txtApellidos.setForeground(Color.BLACK);

txtApellidos.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtApellidos.setBounds(96, 94, 227, 29);

getContentPane().add(txtApellidos);

txtApellidos.setColumns(10);

txtNombres = new JTextField();

txtNombres.addKeyListener(this);

txtNombres.setEditable(false);

txtNombres.setForeground(Color.BLACK);

txtNombres.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtNombres.setBounds(95, 134, 228, 29);

getContentPane().add(txtNombres);

txtNombres.setColumns(10);

cboTipo = new JComboBox<String>();

cboTipo.setEnabled(false);

cboTipo.setForeground(Color.BLACK);

cboTipo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

cboTipo.setBounds(95, 174, 154, 29);

cboTipo.addItem("Administrador");

cboTipo.addItem("Cajero");

getContentPane().add(cboTipo);

txtLogin = new JTextField();

txtLogin.addKeyListener(this);

txtLogin.setEditable(false);

txtLogin.setForeground(Color.BLACK);

txtLogin.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtLogin.setBounds(95, 214, 154, 29);

getContentPane().add(txtLogin);

txtLogin.setColumns(10);

txtPassword = new JTextField();

txtPassword.addKeyListener(this);

txtPassword.setEditable(false);

txtPassword.setForeground(Color.BLACK);

txtPassword.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtPassword.setBounds(95, 254, 154, 29);

getContentPane().add(txtPassword);

txtPassword.setColumns(10);

cboTurno = new JComboBox<String>();

cboTurno.setEnabled(false);

cboTurno.setForeground(Color.BLACK);

cboTurno.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

cboTurno.setBounds(95, 294, 154, 29);

cboTurno.addItem("Noche");

cboTurno.addItem("Día");

getContentPane().add(cboTurno);

txtCodigo = new JTextField();

txtCodigo.addKeyListener(this);

txtCodigo.setEditable(false);

txtCodigo.setForeground(Color.BLACK);

txtCodigo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtCodigo.setColumns(10);

txtCodigo.setBounds(95, 53, 85, 29);

getContentPane().add(txtCodigo);

btnAdicionar = new JButton("ADICIONAR");

btnAdicionar.addActionListener(this);

btnAdicionar.setBackground(new Color(124, 252, 0));

btnAdicionar.setForeground(Color.BLACK);

btnAdicionar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAdicionar.setBounds(20, 347, 137, 39);

ds.setCurvasButton(btnAdicionar, "imagenes/adicionar.png");

getContentPane().add(btnAdicionar);

btnConsultar = new JButton("CONSULTAR");

btnConsultar.addActionListener(this);

btnConsultar.setBackground(Color.CYAN);

btnConsultar.setForeground(Color.BLACK);

btnConsultar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnConsultar.setBounds(177, 347, 145, 39);

ds.setCurvasButton(btnConsultar, "imagenes/consultar.png");

getContentPane().add(btnConsultar);

btnModificar = new JButton("MODIFICAR");

btnModificar.addActionListener(this);

btnModificar.setBackground(new Color(0, 139, 139));

btnModificar.setForeground(Color.BLACK);

btnModificar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnModificar.setBounds(177, 397, 145, 39);

ds.setCurvasButton(btnModificar, "imagenes/modificar.png");

getContentPane().add(btnModificar);

btnEliminar = new JButton("ELIMINAR");

btnEliminar.addActionListener(this);

btnEliminar.setBackground(new Color(127, 255, 212));

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnEliminar.setBounds(20, 397, 137, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

getContentPane().add(btnEliminar);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.addActionListener(this);

btnAceptar.setBackground(new Color(30, 144, 255));

btnAceptar.setEnabled(false);

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAceptar.setBounds(20, 447, 137, 39);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

getContentPane().add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.addActionListener(this);

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setEnabled(false);

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnCancelar.setBounds(177, 447, 145, 39);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

getContentPane().add(btnCancelar);

btnBuscar = new JButton("BUSCAR");

btnBuscar.addActionListener(this);

btnBuscar.setBackground(new Color(102, 205, 170));

btnBuscar.setEnabled(false);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnBuscar.setBounds(213, 53, 110, 29);

ds.setCurvasButton(btnBuscar, "imagenes/buscar.png");

getContentPane().add(btnBuscar);

lblMensaje = new JLabel("");

lblMensaje.setOpaque(true);

lblMensaje.setHorizontalAlignment(SwingConstants.CENTER);

lblMensaje.setForeground(Color.WHITE);

lblMensaje.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 18));

lblMensaje.setBackground(Color.DARK\_GRAY);

lblMensaje.setBounds(10, 11, 313, 29);

getContentPane().add(lblMensaje);

scrollPane = new JScrollPane();

scrollPane.setBounds(333, 11, 908, 520);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblEmpleados = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblEmpleados.setForeground(Color.BLACK);

jtblEmpleados.setFont(new Font("Arial", Font.BOLD, 14));

jtblEmpleados.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblEmpleados.setRowHeight(25);

scrollPane.setViewportView(jtblEmpleados);

btnGuardar = new JButton("GUARDAR EMPLEADOS");

btnGuardar.addActionListener(this);

btnGuardar.setForeground(Color.BLACK);

btnGuardar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnGuardar.setBackground(new Color(0, 250, 154));

btnGuardar.setBounds(20, 492, 303, 39);

ds.setCurvasButton(btnGuardar, "imagenes/grabar.png");

getContentPane().add(btnGuardar);

listado();

modeloTabla();

this.setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnGuardar) {

actionPerformedBtnGuardar(e);

}

if (e.getSource() == btnCancelar) {

actionPerformedBtnCancelar(e);

}

if (e.getSource() == btnAceptar) {

actionPerformedBtnAceptar(e);

}

if (e.getSource() == btnModificar) {

actionPerformedBtnModificar(e);

}

if (e.getSource() == btnEliminar) {

actionPerformedBtnEliminar(e);

}

if (e.getSource() == btnConsultar) {

actionPerformedBtnConsultar(e);

}

if (e.getSource() == btnAdicionar) {

actionPerformedBtnAdicionar(e);

}

if (e.getSource() == btnBuscar) {

actionPerformedBtnBuscar(e);

}

}

protected void actionPerformedBtnAdicionar(ActionEvent arg0) {

tipoOperacion = ADICIONAR;

lblMensaje.setText("Adicionando Empleado");

txtCodigo.setText("" + ae.codigoCorrelativo());

habilitarEntradas(true);

habilitarOperaciones(false);

txtCodigo.setEditable(false);

txtApellidos.requestFocus();

}

protected void actionPerformedBtnConsultar(ActionEvent arg0) {

tipoOperacion = CONSULTAR;

lblMensaje.setText("Consultando Empleado");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnModificar(ActionEvent arg0) {

tipoOperacion = MODIFICAR;

lblMensaje.setText("Modificando Empleado");

habilitarBusqueda(true);

habilitarEntradas(true);

habilitarOperaciones(false);

txtCodigo.requestFocus();

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

tipoOperacion = ELIMINAR;

lblMensaje.setText("Eliminando Empleado");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

consultarEmpleado();

}

protected void actionPerformedBtnAceptar(ActionEvent arg0) {

switch (tipoOperacion) {

case ADICIONAR:

adicionarEmpleado();

break;

case CONSULTAR:

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

lblMensaje.setText("");

break;

case MODIFICAR:

modificarEmpleado();

break;

case ELIMINAR:

eliminarEmpleado();

}

}

protected void actionPerformedBtnCancelar(ActionEvent arg0) {

if (tipoOperacion != ADICIONAR)

habilitarBusqueda(false);

lblMensaje.setText("");

habilitarEntradas(false);

habilitarOperaciones(true);

}

protected void actionPerformedBtnGuardar(ActionEvent e) {

if (ae.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + ae.getArchivo() + "\" ?");

if (ok == 0) {

ae.grabarEmpleados();

Libreria.mensajeInformacion(this, "\"" + ae.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + ae.getArchivo() + "\"");

}

else {

ae.grabarEmpleados();

Libreria.mensajeInformacion(this, "\"" + ae.getArchivo() + "\" ha sido creado");

}

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

}

public void keyTyped(KeyEvent arg0) {

if (arg0.getSource() == txtPassword) {

keyTypedTxtPassword(arg0);

}

if (arg0.getSource() == txtLogin) {

keyTypedTxtLogin(arg0);

}

if (arg0.getSource() == txtNombres) {

keyTypedTxtNombres(arg0);

}

if (arg0.getSource() == txtApellidos) {

keyTypedTxtApellidos(arg0);

}

if (arg0.getSource() == txtCodigo) {

keyTypedTxtCodigo(arg0);

}

}

protected void keyTypedTxtCodigo(KeyEvent e) {

Validaciones.soloNumero(e, txtCodigo, 4);

char t = e.getKeyChar();

if (t == KeyEvent.VK\_ENTER && tipoOperacion != ADICIONAR)

consultarEmpleado();

}

protected void keyTypedTxtApellidos(KeyEvent e) {

Validaciones.soloLetras(e, txtApellidos, 20);

}

protected void keyTypedTxtNombres(KeyEvent e) {

Validaciones.soloLetras(e, txtNombres, 20);

}

protected void keyTypedTxtLogin(KeyEvent e) {

Validaciones.letrasyNumeros(e, txtLogin, 10);

}

protected void keyTypedTxtPassword(KeyEvent e) {

Validaciones.letrasyNumeros(e, txtPassword, 10);

}

// Métodos tipo void sin parámetros

void modeloTabla() {

TableColumnModel tc = jtblEmpleados.getColumnModel();

tc.getColumn(0).setPreferredWidth(70);

tc.getColumn(1).setPreferredWidth(200);

tc.getColumn(2).setPreferredWidth(200);

tc.getColumn(3).setPreferredWidth(150);

tc.getColumn(4).setPreferredWidth(150);

tc.getColumn(5).setPreferredWidth(150);

tc.getColumn(6).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tc.getColumn(4).setCellRenderer(tcr);

tc.getColumn(5).setCellRenderer(tcr);

tc.getColumn(6).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

}

void limpieza() {

txtCodigo.setText("");

txtNombres.setText("");

txtApellidos.setText("");

cboTipo.setSelectedIndex(0);

txtLogin.setText("");

txtPassword.setText("");

cboTurno.setSelectedIndex(0);

}

void listado() {

Libreria.limpiarJTable(jtblEmpleados, dtm);

if (ae.tamaño() == 0) {

}

else {

for (int i = 0; i < ae.tamaño(); i++) {

Empleado x = ae.obtener(i);

dtm.addRow(new Object[] { x.getCodEmpleado(), x.getApellidos(), x.getNombres(),

getTipoEmpleado(x.getTipoEmpleado()), x.getLogin(), x.getPassword(),

getTurnoEmpleado(x.getTurno()) });

}

}

}

void adicionarEmpleado() {

if (leerApellidos().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su APELLIDO");

txtApellidos.setText("");

txtApellidos.requestFocus();

}

else if (leerNombres().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su NOMBRE");

txtNombres.requestFocus();

}

else if (leerLogin().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su LOGIN");

txtLogin.setText("");

txtLogin.requestFocus();

}

else if (leerPassword().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su PASSWORD");

txtPassword.setText("");

txtPassword.requestFocus();

}

else if (validarUsuarioExistente()) {

Libreria.mensajeError(this, "El LOGIN ingresado ya existe.");

txtLogin.setText("");

txtLogin.requestFocus();

}

else {

Empleado nuevo = new Empleado(leerCodigo(), leerApellidos(), leerNombres(), leerTipo(), leerLogin(),

leerPassword(), leerTurno());

ae.adicionar(nuevo);

listado();

habilitarEntradas(false);

habilitarOperaciones(true);

lblMensaje.setText("");

}

}

void consultarEmpleado() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Empleado");

txtCodigo.requestFocus();

}

else {

Empleado x = ae.buscar(leerCodigo());

if (x != null) {

txtNombres.setText(x.getNombres());

txtApellidos.setText(x.getApellidos());

cboTipo.setSelectedIndex(x.getTipoEmpleado());

txtLogin.setText(x.getLogin());

txtPassword.setText(x.getPassword());

cboTurno.setSelectedIndex(x.getTurno());

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

limpieza();

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void modificarEmpleado() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Empleado");

txtCodigo.requestFocus();

}

else {

Empleado x = ae.buscar(leerCodigo());

if (x != null)

if (leerApellidos().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su APELLIDO");

txtApellidos.setText("");

txtApellidos.requestFocus();

}

else if (leerNombres().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su NOMBRE");

txtNombres.requestFocus();

}

else if (leerLogin().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su LOGIN");

txtLogin.setText("");

txtLogin.requestFocus();

}

else if (leerPassword().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su PASSWORD");

txtPassword.setText("");

txtPassword.requestFocus();

}

else if (validarUsuarioExistente()) {

Libreria.mensajeError(this, "El LOGIN ingresado ya existe.");

txtLogin.setText("");

txtLogin.requestFocus();

}

else {

lblMensaje.setText("");

x.setNombres(leerNombres());

x.setApellidos(leerApellidos());

x.setTipoEmpleado(leerTipo());

x.setLogin(leerLogin());

x.setPassword(leerPassword());

x.setTurno(leerTurno());

listado();

habilitarBusqueda(false);

habilitarEntradas(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void eliminarEmpleado() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Empleado");

txtCodigo.requestFocus();

}

else {

Empleado x = ae.buscar(leerCodigo());

if (x != null) {

lblMensaje.setText("");

ae.eliminar(x);

listado();

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

// Métodos tipo void con parámetros

void habilitarBusqueda(boolean sino) {

btnBuscar.setEnabled(sino);

txtCodigo.setEditable(sino);

if (sino)

txtCodigo.requestFocus();

}

void habilitarEntradas(boolean sino) {

if (!sino)

limpieza();

txtNombres.setEditable(sino);

txtApellidos.setEditable(sino);

txtLogin.setEditable(sino);

txtPassword.setEditable(sino);

txtCodigo.setEditable(sino);

cboTipo.setEnabled(sino);

cboTurno.setEnabled(sino);

}

void habilitarOperaciones(boolean sino) {

btnAdicionar.setEnabled(sino);

btnConsultar.setEnabled(sino);

btnModificar.setEnabled(sino);

btnEliminar.setEnabled(sino);

btnAceptar.setEnabled(!sino);

btnCancelar.setEnabled(!sino);

}

// Métodos que retornan valor sin parámetros

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "APELLIDOS", "NOMBRES", "TIPO", "LOGIN", "PASSWORD", "TURNO" };

return columnas;

}

int leerCodigo() {

return Libreria.leerEntero(txtCodigo);

}

String leerNombres() {

return Libreria.leerCadena(txtNombres);

}

String leerApellidos() {

return Libreria.leerCadena(txtApellidos);

}

int leerTipo() {

return cboTipo.getSelectedIndex();

}

String leerLogin() {

return Libreria.leerCadena(txtLogin);

}

String leerPassword() {

return Libreria.leerCadena(txtPassword);

}

int leerTurno() {

return cboTurno.getSelectedIndex();

}

// Métodos que retornan valor con parámetros

boolean validarUsuarioExistente() {

Empleado e;

for (int i = 0; i < ae.tamaño(); i++) {

e = ae.obtener(i);

if (e.getCodEmpleado() != leerCodigo()) {

if (e.getLogin().equalsIgnoreCase(leerLogin())) {

return true;

}

}

else {

if (e.getLogin().equalsIgnoreCase(leerLogin())) {

return false;

}

}

}

return false;

}

String getTipoEmpleado(int tipo) {

if (tipo == 0) {

return "ADMINISTRADOR";

}

else {

return "CAJERO";

}

}

String getTurnoEmpleado(int turno) {

if (turno == 0) {

return "NOCHE";

}

else {

return "DÍA";

}

}

}

DlgHospitalización

package gui;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BorderFactory;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloCamas;

import arreglos.ArregloHospitalizacion;

import arreglos.ArregloPaciente;

import clases.Camas;

import clases.Hospitalizacion;

import libreria.DiseñoObjetos;

import libreria.Fecha;

import libreria.Libreria;

import java.awt.Font;

import java.awt.Point;

import javax.swing.SwingConstants;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.KeyEvent;

import java.awt.event.ItemListener;

import java.awt.event.ItemEvent;

import javax.swing.JRadioButton;

import javax.swing.JTable;

import javax.swing.JPanel;

public class DlgHospitalizacion extends JDialog implements ActionListener, KeyListener, ItemListener {

private static final long serialVersionUID = 1L;

private JLabel lblCodigoDeHospitalizacion;

private JLabel lblCodigoDePaciente;

private JLabel lblCodigoDeEmpleado;

private JLabel lblNumeroDeCama;

private JLabel lblFechaDeLlegada;

private JLabel lblEstado;

private JTextField txtCodigopac;

public JTextField txtCodigoemple;

public JButton btnAceptar;

public JButton btnCancelar;

private JTextField txtCodigohos;

private JScrollPane scrollPane;

public JComboBox<Integer> cboNroCamas;

public JRadioButton rdbtnEconomico;

public JRadioButton rdbtnEjecutivo;

private JLabel lblCategoraCama;

private JComboBox<String> cboDiaLlegada;

private JComboBox<String> cboMesLlegada;

private JComboBox<String> cboAnoLlegada;

public JRadioButton rdbtnAlojado;

public JRadioButton rdbtnPagado;

public JTable jtblHospitalizacion;

public JButton btnBuscarPaciente;

private JLabel lblBordeCategoria;

private JLabel lblBordeEstado;

public JButton btnGrabarHospitalizacion;

private JPanel pnlFiltrar;

public JRadioButton rdbtnFiltrarAlojado;

public JRadioButton rdbtnFiltrarPagado;

public JRadioButton rdbtnFiltrarTodo;

private JLabel lblBorderFiltro;

// Declaración global de código de paciente

public String codigoPaciente = "";

// Declación global variable para grupo de radiobuton

private ButtonGroup bp;

private ButtonGroup bp2;

private ButtonGroup bp3;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// Declaración global de arreglo Paciente

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

// Declaración global de arreglo CAmas

ArregloCamas ac = new ArregloCamas("camas.txt");

// Declaración global de arreglo Hospitalización

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

// Declaración global de librerias

Fecha f = new Fecha();

Libreria lb = new Libreria();

DiseñoObjetos ds = new DiseñoObjetos();

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgHospitalizacion dialog = new DlgHospitalizacion();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgHospitalizacion() {

getContentPane().setBackground(Color.WHITE);

setResizable(false);

setModal(true);

setTitle("HOSPITALIZACIÓN DE PACIENTES");

setBounds(100, 100, 1343, 418);

getContentPane().setLayout(null);

lblCodigoDeHospitalizacion = new JLabel("C\u00D3DIGO HOSPITALIZACI\u00D3N");

lblCodigoDeHospitalizacion.setForeground(Color.BLACK);

lblCodigoDeHospitalizacion.setHorizontalAlignment(SwingConstants.LEFT);

lblCodigoDeHospitalizacion.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCodigoDeHospitalizacion.setBounds(10, 15, 170, 14);

getContentPane().add(lblCodigoDeHospitalizacion);

lblCodigoDePaciente = new JLabel("C\u00D3DIGO PACIENTE");

lblCodigoDePaciente.setForeground(Color.BLACK);

lblCodigoDePaciente.setHorizontalAlignment(SwingConstants.LEFT);

lblCodigoDePaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCodigoDePaciente.setBounds(10, 55, 170, 14);

getContentPane().add(lblCodigoDePaciente);

lblCodigoDeEmpleado = new JLabel("C\u00D3DIGO EMPLEADO");

lblCodigoDeEmpleado.setForeground(Color.BLACK);

lblCodigoDeEmpleado.setHorizontalAlignment(SwingConstants.LEFT);

lblCodigoDeEmpleado.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCodigoDeEmpleado.setBounds(10, 91, 170, 14);

getContentPane().add(lblCodigoDeEmpleado);

lblNumeroDeCama = new JLabel("NRO CAMA");

lblNumeroDeCama.setForeground(Color.BLACK);

lblNumeroDeCama.setHorizontalAlignment(SwingConstants.LEFT);

lblNumeroDeCama.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblNumeroDeCama.setBounds(10, 173, 170, 14);

getContentPane().add(lblNumeroDeCama);

lblFechaDeLlegada = new JLabel("FECHA LLEGADA");

lblFechaDeLlegada.setForeground(Color.BLACK);

lblFechaDeLlegada.setHorizontalAlignment(SwingConstants.LEFT);

lblFechaDeLlegada.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblFechaDeLlegada.setBounds(10, 213, 170, 14);

getContentPane().add(lblFechaDeLlegada);

lblEstado = new JLabel("ESTADO");

lblEstado.setForeground(Color.BLACK);

lblEstado.setHorizontalAlignment(SwingConstants.LEFT);

lblEstado.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblEstado.setBounds(10, 253, 170, 14);

getContentPane().add(lblEstado);

txtCodigopac = new JTextField();

txtCodigopac.setForeground(Color.BLACK);

txtCodigopac.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigopac.setEditable(false);

txtCodigopac.setBounds(190, 48, 119, 29);

getContentPane().add(txtCodigopac);

txtCodigopac.setColumns(10);

txtCodigoemple = new JTextField();

txtCodigoemple.setForeground(Color.BLACK);

txtCodigoemple.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigoemple.setEditable(false);

txtCodigoemple.setBounds(190, 84, 119, 29);

getContentPane().add(txtCodigoemple);

txtCodigoemple.setColumns(10);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnAceptar.addActionListener(this);

btnAceptar.setBounds(58, 292, 149, 38);

btnAceptar.setBackground(Color.CYAN);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

getContentPane().add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnCancelar.addActionListener(this);

btnCancelar.setBounds(217, 292, 149, 38);

btnCancelar.setBackground(new Color(124, 252, 0));

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

getContentPane().add(btnCancelar);

txtCodigohos = new JTextField();

txtCodigohos.setForeground(Color.BLACK);

txtCodigohos.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigohos.setEditable(false);

txtCodigohos.setBounds(190, 8, 119, 29);

getContentPane().add(txtCodigohos);

txtCodigohos.setColumns(10);

scrollPane = new JScrollPane();

scrollPane.setBounds(449, 55, 883, 324);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblHospitalizacion = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblHospitalizacion.setFont(new Font("Arial", Font.BOLD, 14));

jtblHospitalizacion.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblHospitalizacion.setRowHeight(25);

scrollPane.setViewportView(jtblHospitalizacion);

jtblHospitalizacion.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

if (e.getClickCount() == 2) {

int row = jtblHospitalizacion.rowAtPoint(new Point(e.getX(), e.getY()));

if (row < 0) {

}

else {

codigoPaciente = jtblHospitalizacion.getValueAt(row, 1).toString();

dispose();

}

}

}

}

});

cboNroCamas = new JComboBox<Integer>();

cboNroCamas.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboNroCamas.setForeground(Color.BLACK);

cboNroCamas.setBounds(190, 166, 59, 29);

getContentPane().add(cboNroCamas);

rdbtnEconomico = new JRadioButton("Econ\u00F3mico");

rdbtnEconomico.setBackground(Color.WHITE);

rdbtnEconomico.addActionListener(this);

rdbtnEconomico.setSelected(true);

rdbtnEconomico.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

rdbtnEconomico.setForeground(Color.BLACK);

rdbtnEconomico.setBounds(195, 127, 98, 23);

getContentPane().add(rdbtnEconomico);

rdbtnEjecutivo = new JRadioButton("Ejecutivo");

rdbtnEjecutivo.setBackground(Color.WHITE);

rdbtnEjecutivo.addActionListener(this);

rdbtnEjecutivo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

rdbtnEjecutivo.setForeground(Color.BLACK);

rdbtnEjecutivo.setBounds(295, 127, 89, 23);

getContentPane().add(rdbtnEjecutivo);

lblBordeCategoria = new JLabel("");

lblBordeCategoria.setBounds(190, 124, 206, 29);

lblBordeCategoria.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1, true));

getContentPane().add(lblBordeCategoria);

bp = new ButtonGroup();

bp.add(rdbtnEconomico);

bp.add(rdbtnEjecutivo);

lblCategoraCama = new JLabel("CATEGOR\u00CDA CAMA");

lblCategoraCama.setForeground(Color.BLACK);

lblCategoraCama.setHorizontalAlignment(SwingConstants.LEFT);

lblCategoraCama.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCategoraCama.setBounds(10, 131, 170, 14);

getContentPane().add(lblCategoraCama);

cboDiaLlegada = new JComboBox<String>();

cboDiaLlegada.setEnabled(false);

cboDiaLlegada.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboDiaLlegada.setForeground(Color.BLACK);

cboDiaLlegada.setBounds(190, 206, 59, 29);

Fecha.colocarItems(cboDiaLlegada, 1, 31);

Fecha.colocarDiaActual(cboDiaLlegada);

getContentPane().add(cboDiaLlegada);

cboMesLlegada = new JComboBox<String>();

cboMesLlegada.setEnabled(false);

cboMesLlegada.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboMesLlegada.setForeground(Color.BLACK);

cboMesLlegada.setBounds(250, 206, 116, 29);

Fecha.colocarMeses(cboMesLlegada);

Fecha.colocarMesActual(cboMesLlegada);

getContentPane().add(cboMesLlegada);

cboAnoLlegada = new JComboBox<String>();

cboAnoLlegada.setEnabled(false);

cboAnoLlegada.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboAnoLlegada.setForeground(Color.BLACK);

cboAnoLlegada.setBounds(367, 206, 72, 29);

cboAnoLlegada.addItem("" + Fecha.añoActual());

getContentPane().add(cboAnoLlegada);

rdbtnAlojado = new JRadioButton("Alojado");

rdbtnAlojado.setEnabled(false);

rdbtnAlojado.setSelected(true);

rdbtnAlojado.setForeground(Color.BLACK);

rdbtnAlojado.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

rdbtnAlojado.setBackground(Color.WHITE);

rdbtnAlojado.setBounds(195, 249, 98, 23);

getContentPane().add(rdbtnAlojado);

rdbtnPagado = new JRadioButton("Pagado");

rdbtnPagado.setEnabled(false);

rdbtnPagado.setForeground(Color.BLACK);

rdbtnPagado.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

rdbtnPagado.setBackground(Color.WHITE);

rdbtnPagado.setBounds(295, 249, 89, 23);

getContentPane().add(rdbtnPagado);

this.setLocationRelativeTo(this);

bp2 = new ButtonGroup();

bp2.add(rdbtnAlojado);

bp2.add(rdbtnPagado);

lblBordeEstado = new JLabel("");

lblBordeEstado.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1, true));

lblBordeEstado.setBounds(190, 246, 206, 29);

getContentPane().add(lblBordeEstado);

btnBuscarPaciente = new JButton("...");

btnBuscarPaciente.addActionListener(this);

btnBuscarPaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnBuscarPaciente.setForeground(Color.BLACK);

btnBuscarPaciente.setBackground(new Color(127, 255, 212));

btnBuscarPaciente.setBounds(313, 48, 62, 29);

getContentPane().add(btnBuscarPaciente);

btnGrabarHospitalizacion = new JButton("GRABAR HOSPITALIZACI\u00D3N");

btnGrabarHospitalizacion.addActionListener(this);

btnGrabarHospitalizacion.setForeground(Color.BLACK);

btnGrabarHospitalizacion.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnGrabarHospitalizacion.setBounds(58, 341, 308, 38);

btnGrabarHospitalizacion.setBackground(new Color(0, 250, 154));

ds.setCurvasButton(btnGrabarHospitalizacion, "imagenes/grabar.png");

getContentPane().add(btnGrabarHospitalizacion);

pnlFiltrar = new JPanel();

pnlFiltrar.setBorder(null);

pnlFiltrar.setBackground(Color.WHITE);

pnlFiltrar.setBounds(760, 15, 259, 29);

getContentPane().add(pnlFiltrar);

pnlFiltrar.setLayout(null);

rdbtnFiltrarAlojado = new JRadioButton("ALOJADO");

rdbtnFiltrarAlojado.addActionListener(this);

rdbtnFiltrarAlojado.setSelected(true);

rdbtnFiltrarAlojado.setBackground(Color.WHITE);

rdbtnFiltrarAlojado.setForeground(Color.BLACK);

rdbtnFiltrarAlojado.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnFiltrarAlojado.setBounds(6, 0, 90, 23);

pnlFiltrar.add(rdbtnFiltrarAlojado);

rdbtnFiltrarPagado = new JRadioButton("PAGADO");

rdbtnFiltrarPagado.addActionListener(this);

rdbtnFiltrarPagado.setBackground(Color.WHITE);

rdbtnFiltrarPagado.setForeground(Color.BLACK);

rdbtnFiltrarPagado.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnFiltrarPagado.setBounds(98, 0, 90, 23);

pnlFiltrar.add(rdbtnFiltrarPagado);

rdbtnFiltrarTodo = new JRadioButton("TODO");

rdbtnFiltrarTodo.addActionListener(this);

rdbtnFiltrarTodo.setBackground(Color.WHITE);

rdbtnFiltrarTodo.setForeground(Color.BLACK);

rdbtnFiltrarTodo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

rdbtnFiltrarTodo.setBounds(190, 0, 78, 23);

pnlFiltrar.add(rdbtnFiltrarTodo);

bp3 = new ButtonGroup();

bp3.add(rdbtnFiltrarAlojado);

bp3.add(rdbtnFiltrarPagado);

bp3.add(rdbtnFiltrarTodo);

lblBorderFiltro = new JLabel("");

lblBorderFiltro.setBounds(756, 8, 274, 38);

getContentPane().add(lblBorderFiltro);

lblBorderFiltro.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1, true));

modeloTabla();

setCodigoHospitalizacion();

listadoAlojado();

listarCamas(cboNroCamas);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == rdbtnFiltrarTodo) {

actionPerformedRdbtnFiltrarTodo(arg0);

}

if (arg0.getSource() == rdbtnFiltrarPagado) {

actionPerformedRdbtnFiltrarPagado(arg0);

}

if (arg0.getSource() == rdbtnFiltrarAlojado) {

actionPerformedRdbtnFiltrarAlojado(arg0);

}

if (arg0.getSource() == btnGrabarHospitalizacion) {

actionPerformedBtnGrabarHospitalizacion(arg0);

}

if (arg0.getSource() == btnBuscarPaciente) {

actionPerformedBtnBuscarPaciente(arg0);

}

if (arg0.getSource() == rdbtnEjecutivo) {

actionPerformedRdbtnEjecutivo(arg0);

}

if (arg0.getSource() == rdbtnEconomico) {

actionPerformedRdbtnEconomico(arg0);

}

if (arg0.getSource() == btnCancelar) {

actionPerformedBtnCancelar(arg0);

}

if (arg0.getSource() == btnAceptar) {

actionPerformedBtnAceptar(arg0);

}

}

protected void actionPerformedBtnBuscarPaciente(ActionEvent arg0) {

rdbtnFiltrarAlojado.setSelected(true);

listadoAlojado();

DlgPaciente pac = new DlgPaciente();

pac.setTitle("LISTADO DE PACIENTES");

pac.setVisible(true);

int c = 0;

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

if (jtblHospitalizacion.getValueAt(i, 1).toString().equals("" + pac.codigoPaciente)

&& jtblHospitalizacion.getValueAt(i, 8).toString().equals("ALOJADO")) {

c++;

Libreria.mensajeInformacion(this,

"El paciente ya está alojado\n" + "Código de hospitalización : "

+ jtblHospitalizacion.getValueAt(i, 0).toString() + "\n" + "Nro de cama : "

+ jtblHospitalizacion.getValueAt(i, 3).toString() + "\n" + "Fecha Llegada : "

+ jtblHospitalizacion.getValueAt(i, 4).toString() + "\n" + "Hora Llegada : "

+ jtblHospitalizacion.getValueAt(i, 5).toString() + "\n");

txtCodigopac.setText("");

break;

}

}

if (c == 0) {

txtCodigopac.setText(pac.codigoPaciente);

}

}

protected void actionPerformedBtnGrabarHospitalizacion(ActionEvent arg0) {

if (ah.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + ah.getArchivo() + "\" ?");

if (ok == 0) {

ah.grabarHospitalizacion();

ac.grabarCamas();

Libreria.mensajeInformacion(this, "\"" + ah.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + ah.getArchivo() + "\"");

}

else {

ah.grabarHospitalizacion();

ac.grabarCamas();

Libreria.mensajeInformacion(this, "\"" + ah.getArchivo() + "\" ha sido creado");

}

}

protected void actionPerformedBtnAceptar(ActionEvent e) {

adicionarHospitalizacion();

}

protected void actionPerformedBtnCancelar(ActionEvent e) {

dispose();

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

}

public void keyTyped(KeyEvent arg0) {

}

public void itemStateChanged(ItemEvent e) {

}

protected void actionPerformedRdbtnEconomico(ActionEvent arg0) {

listarCamas(cboNroCamas);

}

protected void actionPerformedRdbtnEjecutivo(ActionEvent arg0) {

listarCamas(cboNroCamas);

}

protected void actionPerformedRdbtnFiltrarAlojado(ActionEvent arg0) {

listadoAlojado();

}

protected void actionPerformedRdbtnFiltrarPagado(ActionEvent arg0) {

listadoPagado();

}

protected void actionPerformedRdbtnFiltrarTodo(ActionEvent arg0) {

listadoTodo();

}

// Métodos tipo void sin parámetros

void setCodigoHospitalizacion() {

txtCodigohos.setText("" + ah.codigoCorrelativo());

}

void limpieza() {

txtCodigohos.setText("");

txtCodigopac.setText("");

listarCamas(cboNroCamas);

rdbtnEconomico.isSelected();

setCodigoHospitalizacion();

}

void modeloTabla() {

TableColumnModel tc = jtblHospitalizacion.getColumnModel();

tc.getColumn(0).setPreferredWidth(50);

tc.getColumn(1).setPreferredWidth(75);

tc.getColumn(2).setPreferredWidth(82);

tc.getColumn(3).setPreferredWidth(50);

tc.getColumn(4).setPreferredWidth(85);

tc.getColumn(5).setPreferredWidth(80);

tc.getColumn(6).setPreferredWidth(75);

tc.getColumn(7).setPreferredWidth(75);

tc.getColumn(8).setPreferredWidth(75);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(2).setCellRenderer(tcr);

tc.getColumn(1).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tc.getColumn(4).setCellRenderer(tcr);

tc.getColumn(5).setCellRenderer(tcr);

tc.getColumn(6).setCellRenderer(tcr);

tc.getColumn(7).setCellRenderer(tcr);

tc.getColumn(8).setCellRenderer(tcr);

}

void listadoTodo() {

Libreria.limpiarJTable(jtblHospitalizacion, dtm);

if (ah.tamaño() == 0) {

}

else {

for (int i = 0; i < ah.tamaño(); i++) {

Hospitalizacion x = ah.obtener(i);

if (x.getHorSali() != 0) {

dtm.addRow(new Object[] { x.getCodHosp(), x.getCodPaci(), x.getCodEmpl(), x.getNumCama(),

Fecha.dd\_mm\_aaaa(x.getFecLleg()), Fecha.hh\_mm\_ss(x.getHorLleg()),

Fecha.dd\_mm\_aaaa(x.getFecSali()), Fecha.hh\_mm\_ss(x.getHorSali()), Estado(x.getEstado()) });

}

else {

dtm.addRow(new Object[] { x.getCodHosp(), x.getCodPaci(), x.getCodEmpl(), x.getNumCama(),

Fecha.dd\_mm\_aaaa(x.getFecLleg()), Fecha.hh\_mm\_ss(x.getHorLleg()),

horaFechaSalida(x.getFecSali()), horaFechaSalida(x.getHorSali()), Estado(x.getEstado()) });

}

}

}

}

void listadoAlojado() {

Libreria.limpiarJTable(jtblHospitalizacion, dtm);

if (ah.tamaño() == 0) {

}

else {

for (int i = 0; i < ah.tamaño(); i++) {

Hospitalizacion x = ah.obtener(i);

if (x.getEstado() == 1) {

dtm.addRow(new Object[] { x.getCodHosp(), x.getCodPaci(), x.getCodEmpl(), x.getNumCama(),

Fecha.dd\_mm\_aaaa(x.getFecLleg()), Fecha.hh\_mm\_ss(x.getHorLleg()),

horaFechaSalida(x.getFecSali()), horaFechaSalida(x.getHorSali()), Estado(x.getEstado()) });

}

}

}

}

void listadoPagado() {

Libreria.limpiarJTable(jtblHospitalizacion, dtm);

if (ah.tamaño() == 0) {

}

else {

for (int i = 0; i < ah.tamaño(); i++) {

Hospitalizacion x = ah.obtener(i);

if (x.getEstado() == 2) {

dtm.addRow(new Object[] { x.getCodHosp(), x.getCodPaci(), x.getCodEmpl(), x.getNumCama(),

Fecha.dd\_mm\_aaaa(x.getFecLleg()), Fecha.hh\_mm\_ss(x.getHorLleg()),

Fecha.dd\_mm\_aaaa(x.getFecSali()), Fecha.hh\_mm\_ss(x.getHorSali()), Estado(x.getEstado()) });

}

}

}

}

void adicionarHospitalizacion() {

if (leerCodigopac() < 0) {

Libreria.mensajeAdvertencia(this, "No seleccionó CÓDIGO de paciente");

}

else if (cboNroCamas.getSelectedIndex() < 0) {

Libreria.mensajeAdvertencia(this, "No hay camas disponibles en la categoría");

}

else {

Hospitalizacion nueva = new Hospitalizacion(leerCodigohos(), leerCodigopac(), leerCodigoemple(),

leerNumerocama(), leerFechallegada(), Fecha.obtenerHoraActual(), 0, 0, leerEstado());

ah.adicionar(nueva);

fijarCamaOcupado(leerNumerocama());

limpieza();

listadoAlojado();

}

}

// Métodos tipo void con parámetros

void listarCamas(JComboBox<Integer> cbo) {

if (rdbtnEconomico.isSelected()) {

cboNroCamas.removeAllItems();

for (int i = 0; i < ac.tamaño(); i++) {

if (ac.obtener(i).getCategoria() == 0) {

if (ac.obtener(i).getEstado() == 0) {

cbo.addItem(ac.obtener(i).getNroCama());

}

}

}

}

else {

cboNroCamas.removeAllItems();

for (int i = 0; i < ac.tamaño(); i++) {

if (ac.obtener(i).getCategoria() == 1) {

if (ac.obtener(i).getEstado() == 0) {

cbo.addItem(ac.obtener(i).getNroCama());

}

}

}

}

}

void fijarCamaOcupado(int nroCama) {

Camas x = ac.buscar(nroCama);

x.setEstado(1);

}

// Métodos que retornan valor sin parámetros

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "COD PACIENTE", "COD EMPLEADO", "NRO CAMA", "FECHA LLEGADA",

"HORA LLEGADA", "FECHA SALIDA", "HORA SALIDA", "ESTADO" };

return columnas;

}

int leerCodigohos() {

return Libreria.leerEntero(txtCodigohos);

}

int leerCodigopac() {

return Libreria.leerEntero(txtCodigopac);

}

int leerCodigoemple() {

return Libreria.leerEntero(txtCodigoemple);

}

int leerNumerocama() {

return Integer.parseInt(cboNroCamas.getSelectedItem().toString());

}

int leerFechallegada() {

return Fecha.getFecha(cboDiaLlegada, cboMesLlegada, cboAnoLlegada);

}

int leerEstado() {

if (rdbtnAlojado.isSelected()) {

return 1;

}

else {

return 2;

}

}

private String Estado(int estado) {

if (estado == 1) {

return "ALOJADO";

}

else {

return "PAGADO";

}

}

private String horaFechaSalida(int num) {

return "----";

}

int leerCodigo() {

return Libreria.leerEntero(txtCodigopac);

}

}

DlgLogin

package gui;

import javax.swing.JDialog;

import javax.swing.JTextField;

import javax.swing.JLabel;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JPanel;

import javax.swing.SwingConstants;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JPasswordField;

import javax.swing.UIManager;

import arreglos.ArregloEmpleado;

import clases.Empleado;

import libreria.DiseñoObjetos;

import libreria.Libreria;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class DlgLogin extends JDialog implements ActionListener {

private static final long serialVersionUID = 1L;

private JTextField txtUsuario;

private JLabel lblUsuario;

private JLabel lblPassword;

private JPanel panelLogin;

private JButton btnIngresar;

private JPasswordField passwordField;

private JButton btnCancelar;

private JLabel lblImagen;

// Declaración global libreria Diseño Objetos

DiseñoObjetos ds = new DiseñoObjetos();

// Declaración global libreria ArregloEmpleado

ArregloEmpleado ae = new ArregloEmpleado("empleados.txt");

// Variable global de empleado que inicio sesión

public int codigoEmpleado = -1;

public int tipoEmpleado;

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgLogin dialog = new DlgLogin();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgLogin() {

getContentPane().setBackground(Color.DARK\_GRAY);

setModal(true);

setResizable(false);

setTitle("Inicio de Sesi\u00F3n");

setBounds(100, 100, 489, 276);

getContentPane().setLayout(null);

panelLogin = new JPanel();

panelLogin.setBackground(Color.WHITE);

panelLogin.setBorder(UIManager.getBorder("ProgressBar.border"));

panelLogin.setBounds(10, 11, 463, 226);

getContentPane().add(panelLogin);

panelLogin.setLayout(null);

panelLogin.setBorder(BorderFactory.createLineBorder(Color.RED, 2, true));

lblUsuario = new JLabel("USUARIO");

lblUsuario.setForeground(Color.BLACK);

lblUsuario.setHorizontalAlignment(SwingConstants.RIGHT);

lblUsuario.setBounds(12, 51, 80, 14);

panelLogin.add(lblUsuario);

lblUsuario.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblPassword = new JLabel("PASSWORD");

lblPassword.setForeground(Color.BLACK);

lblPassword.setHorizontalAlignment(SwingConstants.RIGHT);

lblPassword.setBounds(12, 103, 78, 14);

panelLogin.add(lblPassword);

lblPassword.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtUsuario = new JTextField();

txtUsuario.setForeground(Color.BLACK);

txtUsuario.setBounds(100, 38, 211, 41);

panelLogin.add(txtUsuario);

txtUsuario.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtUsuario.setColumns(10);

btnIngresar = new JButton("INGRESAR");

btnIngresar.addActionListener(this);

btnIngresar.setForeground(Color.BLACK);

btnIngresar.setBackground(new Color(152, 251, 152));

btnIngresar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnIngresar.setBounds(39, 146, 122, 41);

panelLogin.add(btnIngresar);

ds.setCurvasButton(btnIngresar, "imagenes/aceptar.png");

passwordField = new JPasswordField();

passwordField.setForeground(Color.BLACK);

passwordField.setBounds(100, 89, 211, 41);

panelLogin.add(passwordField);

btnCancelar = new JButton("CANCELAR");

btnCancelar.addActionListener(this);

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnCancelar.setBounds(178, 146, 122, 41);

panelLogin.add(btnCancelar);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

lblImagen = new JLabel("");

lblImagen.setBounds(321, 11, 132, 202);

lblImagen.setIcon(new ImageIcon("imagenes/login.jpg"));

panelLogin.add(lblImagen);

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnCancelar) {

actionPerformedBtnCancelar(e);

}

if (e.getSource() == btnIngresar) {

actionPerformedBtnIngresar(e);

}

}

protected void actionPerformedBtnIngresar(ActionEvent e) {

if (leerUsuario().equals("")) {

Libreria.mensajeAdvertencia(this, "Debe ingresar su USUARIO");

txtUsuario.requestFocus();

}

else if (leerPasword().equals("")) {

Libreria.mensajeAdvertencia(this, "Debe ingresar su PASSWORD");

passwordField.requestFocus();

}

else {

if (ae.tamaño() == 0) {

ae.adicionar(new Empleado(1001, "Huaman", "Esthefano", 0, "admin", "admin", 1));

ae.adicionar(new Empleado(1002, "Altamirano", "Pablo", 0, "pablo", "pablo123", 1));

ae.grabarEmpleados();

validarLogin();

}

else {

validarLogin();

}

}

}

protected void actionPerformedBtnCancelar(ActionEvent e) {

this.dispose();

}

// métodos void con parámetros

void validarLogin() {

int c = 0;

Empleado e;

for (int i = 0; i < ae.tamaño(); i++) {

e = ae.obtener(i);

if (e.getLogin().equalsIgnoreCase(leerUsuario())

&& e.getPassword().equalsIgnoreCase(leerPasword())) {

Libreria.mensajeInformacion(this,

"BIENVENIDO: " + e.getNombres().toUpperCase() + " " + e.getApellidos().toUpperCase());

c++;

codigoEmpleado = e.getCodEmpleado();

tipoEmpleado = e.getTipoEmpleado();

dispose();

break;

}

}

if (c == 0) {

Libreria.mensajeAdvertencia(this, "LOS DATOS INGRESADOS SON INCORRECTOS.");

txtUsuario.setText("");

passwordField.setText("");

txtUsuario.requestFocus();

}

}

// metódos con retorno sin parámetros

String leerUsuario() {

return Libreria.leerCadena(txtUsuario);

}

String leerPasword() {

return Libreria.getPassword(passwordField);

}

}

DlgMedicamentos

package gui;

import java.awt.Color;

import java.awt.Font;

import java.awt.Point;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloMedicamento;

import clases.Medicamentos;

import libreria.DiseñoObjetos;

import libreria.Libreria;

import libreria.Validaciones;

import java.awt.event.MouseListener;

public class DlgMedicamento extends JDialog implements ActionListener, KeyListener, MouseListener {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JLabel lblCodigo;

private JLabel lblNombre;

private JLabel lblLaboratorio;

private JLabel lblPrecio;

private JLabel lblStock;

private JTextField txtNombre;

private JTextField txtLaboratorio;

private JTextField txtPrecio;

private JTextField txtStock;

private JScrollPane scrollPane;

private JTextField txtCodigo;

private JButton btnBuscar;

private JScrollPane scpMedicamentos;

public JButton btnAdicionar;

public JButton btnConsultar;

public JButton btnModificar;

public JButton btnEliminar;

private JButton btnAceptar;

private JButton btnCancelar;

private JLabel lblMensaje;

public JButton btnGuardarMedicamentos;

private JTable jtblMedicamentos;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

/\*

\* Tipo de operación a procesar: Adicionar, Modificar, Eliminar o Consultar

\*/

private int tipoOperacion;

/\* Variables globales para realizar venta \*/

public int cantidadVender = 0;

public String nombreProducto = "";

public double precioProducto = 0.0;

public String codigoProducto = "";

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// Declaración global

ArregloMedicamento am = new ArregloMedicamento("medicamentos.txt");

// Declaración global de libreria Validaciones

Validaciones v = new Validaciones();

// Declaración global de libreria DiseñoObjetos

DiseñoObjetos ds = new DiseñoObjetos();

// Declaración global de libreria

Libreria lb = new Libreria();

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgMedicamento dialog = new DlgMedicamento();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

public DlgMedicamento() {

setResizable(false);

setModal(true);

setTitle("MANTENIMIENTO | MEDICAMENTOS");

setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

setBounds(100, 100, 1038, 487);

contentPane = new JPanel();

contentPane.setBackground(Color.WHITE);

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

lblCodigo = new JLabel("CODIGO");

lblCodigo.setForeground(Color.BLACK);

lblCodigo.setHorizontalAlignment(SwingConstants.RIGHT);

lblCodigo.setFont(new Font("Arial", Font.BOLD, 13));

lblCodigo.setBounds(13, 59, 86, 14);

contentPane.add(lblCodigo);

lblNombre = new JLabel("NOMBRE");

lblNombre.setForeground(Color.BLACK);

lblNombre.setHorizontalAlignment(SwingConstants.RIGHT);

lblNombre.setFont(new Font("Arial", Font.BOLD, 13));

lblNombre.setBounds(0, 98, 99, 14);

contentPane.add(lblNombre);

lblLaboratorio = new JLabel("LABORATORIO");

lblLaboratorio.setForeground(Color.BLACK);

lblLaboratorio.setHorizontalAlignment(SwingConstants.RIGHT);

lblLaboratorio.setFont(new Font("Arial", Font.BOLD, 13));

lblLaboratorio.setBounds(0, 136, 99, 14);

contentPane.add(lblLaboratorio);

lblPrecio = new JLabel("PRECIO");

lblPrecio.setForeground(Color.BLACK);

lblPrecio.setHorizontalAlignment(SwingConstants.RIGHT);

lblPrecio.setFont(new Font("Arial", Font.BOLD, 13));

lblPrecio.setBounds(12, 176, 89, 14);

contentPane.add(lblPrecio);

lblStock = new JLabel("STOCK");

lblStock.setForeground(Color.BLACK);

lblStock.setHorizontalAlignment(SwingConstants.RIGHT);

lblStock.setFont(new Font("Arial", Font.BOLD, 13));

lblStock.setBounds(23, 216, 76, 14);

contentPane.add(lblStock);

txtNombre = new JTextField();

txtNombre.addKeyListener(this);

txtNombre.setEditable(false);

txtNombre.setFont(new Font("Arial", Font.BOLD, 14));

txtNombre.setForeground(Color.BLACK);

txtNombre.setBounds(111, 91, 270, 29);

contentPane.add(txtNombre);

txtNombre.setColumns(10);

txtLaboratorio = new JTextField();

txtLaboratorio.addKeyListener(this);

txtLaboratorio.setEditable(false);

txtLaboratorio.setFont(new Font("Arial", Font.BOLD, 14));

txtLaboratorio.setForeground(Color.BLACK);

txtLaboratorio.setBounds(111, 129, 270, 29);

contentPane.add(txtLaboratorio);

txtLaboratorio.setColumns(10);

txtPrecio = new JTextField();

txtPrecio.addKeyListener(this);

txtPrecio.setEditable(false);

txtPrecio.setFont(new Font("Arial", Font.BOLD, 14));

txtPrecio.setForeground(Color.BLACK);

txtPrecio.setBounds(111, 169, 76, 29);

contentPane.add(txtPrecio);

txtPrecio.setColumns(10);

txtStock = new JTextField();

txtStock.addKeyListener(this);

txtStock.setEditable(false);

txtStock.setFont(new Font("Arial", Font.BOLD, 14));

txtStock.setForeground(Color.BLACK);

txtStock.setBounds(111, 209, 76, 29);

contentPane.add(txtStock);

txtStock.setColumns(10);

scrollPane = new JScrollPane();

scrollPane.setBounds(413, 219, -394, -54);

contentPane.add(scrollPane);

txtCodigo = new JTextField();

txtCodigo.addKeyListener(this);

txtCodigo.setEditable(false);

txtCodigo.setFont(new Font("Arial", Font.BOLD, 14));

txtCodigo.setForeground(Color.BLACK);

txtCodigo.setColumns(10);

txtCodigo.setBounds(111, 52, 126, 29);

contentPane.add(txtCodigo);

btnBuscar = new JButton("BUSCAR");

btnBuscar.addActionListener(this);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnBuscar.setEnabled(false);

btnBuscar.setBackground(new Color(102, 205, 170));

btnBuscar.setBounds(271, 52, 110, 29);

ds.setCurvasButton(btnBuscar, "imagenes/buscar.png");

contentPane.add(btnBuscar);

scpMedicamentos = new JScrollPane();

scpMedicamentos.setBounds(391, 11, 631, 436);

contentPane.add(scpMedicamentos);

dtm = new DefaultTableModel(null, getColumnas());

jtblMedicamentos = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblMedicamentos.addMouseListener(this);

jtblMedicamentos.setForeground(Color.BLACK);

jtblMedicamentos.setFont(new Font("Arial", Font.BOLD, 14));

jtblMedicamentos.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblMedicamentos.setRowHeight(25);

scpMedicamentos.setViewportView(jtblMedicamentos);

btnAdicionar = new JButton("ADICIONAR");

btnAdicionar.addActionListener(this);

btnAdicionar.setForeground(Color.BLACK);

btnAdicionar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAdicionar.setBackground(new Color(124, 252, 0));

btnAdicionar.setBounds(42, 260, 137, 39);

ds.setCurvasButton(btnAdicionar, "imagenes/adicionar.png");

contentPane.add(btnAdicionar);

btnConsultar = new JButton("CONSULTAR");

btnConsultar.addActionListener(this);

btnConsultar.setForeground(Color.BLACK);

btnConsultar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnConsultar.setBackground(Color.CYAN);

btnConsultar.setBounds(199, 260, 145, 39);

ds.setCurvasButton(btnConsultar, "imagenes/consultar.png");

contentPane.add(btnConsultar);

btnModificar = new JButton("MODIFICAR");

btnModificar.addActionListener(this);

btnModificar.setForeground(Color.BLACK);

btnModificar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnModificar.setBackground(new Color(0, 139, 139));

btnModificar.setBounds(199, 310, 145, 39);

ds.setCurvasButton(btnModificar, "imagenes/modificar.png");

contentPane.add(btnModificar);

btnEliminar = new JButton("ELIMINAR");

btnEliminar.addActionListener(this);

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnEliminar.setBackground(new Color(127, 255, 212));

btnEliminar.setBounds(42, 310, 137, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

contentPane.add(btnEliminar);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.addActionListener(this);

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAceptar.setEnabled(false);

btnAceptar.setBackground(new Color(30, 144, 255));

btnAceptar.setBounds(42, 360, 137, 39);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

contentPane.add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.addActionListener(this);

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnCancelar.setEnabled(false);

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setBounds(199, 360, 145, 39);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

contentPane.add(btnCancelar);

lblMensaje = new JLabel("");

lblMensaje.setOpaque(true);

lblMensaje.setHorizontalAlignment(SwingConstants.CENTER);

lblMensaje.setForeground(Color.WHITE);

lblMensaje.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 18));

lblMensaje.setBackground(Color.DARK\_GRAY);

lblMensaje.setBounds(10, 11, 376, 29);

contentPane.add(lblMensaje);

btnGuardarMedicamentos = new JButton("GUARDAR MEDICAMENTOS");

btnGuardarMedicamentos.addActionListener(this);

btnGuardarMedicamentos.setForeground(Color.BLACK);

btnGuardarMedicamentos.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnGuardarMedicamentos.setBackground(new Color(0, 250, 154));

btnGuardarMedicamentos.setBounds(42, 408, 302, 39);

ds.setCurvasButton(btnGuardarMedicamentos, "imagenes/grabar.png");

contentPane.add(btnGuardarMedicamentos);

listado();

modeloTabla();

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnGuardarMedicamentos) {

actionPerformedBtnGuardarMedicamentos(arg0);

}

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

if (arg0.getSource() == btnCancelar) {

actionPerformedBtnCancelar(arg0);

}

if (arg0.getSource() == btnAceptar) {

actionPerformedBtnAceptar(arg0);

}

if (arg0.getSource() == btnModificar) {

actionPerformedBtnModificar(arg0);

}

if (arg0.getSource() == btnEliminar) {

actionPerformedBtnEliminar(arg0);

}

if (arg0.getSource() == btnConsultar) {

actionPerformedBtnConsultar(arg0);

}

if (arg0.getSource() == btnAdicionar) {

actionPerformedBtnAdicionar(arg0);

}

}

protected void actionPerformedBtnAdicionar(ActionEvent arg0) {

tipoOperacion = ADICIONAR;

lblMensaje.setText("Adicionando Medicamento");

txtCodigo.setText("" + am.codigoCorrelativo());

habilitarEntradas(true);

habilitarOperaciones(false);

txtCodigo.setEditable(false);

txtNombre.requestFocus();

}

protected void actionPerformedBtnConsultar(ActionEvent arg0) {

tipoOperacion = CONSULTAR;

lblMensaje.setText("Consultando Medicamento");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

tipoOperacion = ELIMINAR;

lblMensaje.setText("Eliminando Medicamento");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnModificar(ActionEvent arg0) {

tipoOperacion = MODIFICAR;

lblMensaje.setText("Modificando Medicamento");

habilitarBusqueda(true);

habilitarEntradas(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

consultarMedicamento();

}

protected void actionPerformedBtnAceptar(ActionEvent arg0) {

switch (tipoOperacion) {

case ADICIONAR:

adicionarMedicamento();

lblMensaje.setText("");

break;

case CONSULTAR:

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

lblMensaje.setText("");

break;

case MODIFICAR:

modificarMedicamento();

lblMensaje.setText("");

break;

case ELIMINAR:

eliminarMedicamento();

lblMensaje.setText("");

}

}

protected void actionPerformedBtnGuardarMedicamentos(ActionEvent arg0) {

if (am.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + am.getArchivo() + "\" ?");

if (ok == 0) {

am.grabarMedicamentos();

Libreria.mensajeInformacion(this, "\"" + am.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + am.getArchivo() + "\"");

}

else {

am.grabarMedicamentos();

Libreria.mensajeInformacion(this, "\"" + am.getArchivo() + "\" ha sido creado");

}

}

protected void actionPerformedBtnCancelar(ActionEvent arg0) {

if (tipoOperacion != ADICIONAR)

habilitarBusqueda(false);

lblMensaje.setText("");

habilitarEntradas(false);

habilitarOperaciones(true);

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

}

public void keyTyped(KeyEvent arg0) {

if (arg0.getSource() == txtStock) {

keyTypedTxtStock(arg0);

}

if (arg0.getSource() == txtPrecio) {

keyTypedTxtPrecio(arg0);

}

if (arg0.getSource() == txtLaboratorio) {

keyTypedTxtLaboratorio(arg0);

}

if (arg0.getSource() == txtNombre) {

keyTypedTxtNombre(arg0);

}

if (arg0.getSource() == txtCodigo) {

keyTypedTxtCodigo(arg0);

}

}

protected void keyTypedTxtCodigo(KeyEvent e) {

Validaciones.soloNumero(e, txtCodigo, 5);

char t = e.getKeyChar();

if (t == KeyEvent.VK\_ENTER) {

consultarMedicamento();

}

}

protected void keyTypedTxtNombre(KeyEvent e) {

Validaciones.cadenaTexto(e, txtNombre, 20);

}

protected void keyTypedTxtLaboratorio(KeyEvent e) {

Validaciones.cadenaTexto(e, txtLaboratorio, 20);

}

protected void keyTypedTxtPrecio(KeyEvent e) {

Validaciones.numeroReal(txtPrecio, e, 5);

}

protected void keyTypedTxtStock(KeyEvent e) {

Validaciones.soloNumero(e, txtStock, 4);

}

public void mouseClicked(MouseEvent arg0) {

if (arg0.getSource() == jtblMedicamentos) {

mouseClickedJtblMedicamentos(arg0);

}

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

}

public void mousePressed(MouseEvent e) {

}

public void mouseReleased(MouseEvent e) {

}

protected void mouseClickedJtblMedicamentos(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

if (e.getClickCount() == 2) {

int row = jtblMedicamentos.rowAtPoint(new Point(e.getX(), e.getY()));

if (row >= 0) {

int cant = cantidadaVender();

if (cant >= 0) {

if (cant <= Integer.parseInt(jtblMedicamentos.getValueAt(row, 4).toString())) {

cantidadVender = cant;

precioProducto = Double.parseDouble(jtblMedicamentos.getValueAt(row, 3).toString());

nombreProducto = jtblMedicamentos.getValueAt(row, 1).toString();

codigoProducto = jtblMedicamentos.getValueAt(row, 0).toString();

dispose();

}

else {

Libreria.mensajeInformacion(this,

"La cantidad a vender es mayor a la cantidad de stock disponible");

}

}

else {

}

}

}

}

}

// Métodos tipo void sin parámetros

void limpieza() {

txtNombre.setText("");

txtCodigo.setText("");

txtLaboratorio.setText("");

txtPrecio.setText("");

txtStock.setText("");

}

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "NOMBRE", "LABORATORIO", "PRECIO", "STOCK" };

return columnas;

}

void modeloTabla() {

TableColumnModel tc = jtblMedicamentos.getColumnModel();

tc.getColumn(0).setPreferredWidth(100);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(3).setCellRenderer(tcr);

tc.getColumn(4).setCellRenderer(tcr);

}

void listado() {

Libreria.limpiarJTable(jtblMedicamentos, dtm);

if (am.tamaño() == 0) {

}

else {

for (int i = 0; i < am.tamaño(); i++) {

Medicamentos x = am.obtener(i);

dtm.addRow(new Object[] { x.getCodMedicamento(), x.getNombre(), x.getLaboratorio(), x.getPrecio(),

x.getStock() });

}

}

}

void adicionarMedicamento() {

if (leerNombre().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó el NOMBRE");

txtNombre.requestFocus();

}

else if (leerLaboratorio().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó el LABORATORIO");

txtLaboratorio.setText("");

txtLaboratorio.requestFocus();

}

else if (leerPrecio() < 10) {

Libreria.mensajeAdvertencia(this, "Ingrese el PRECIO de forma correcta");

txtPrecio.setText("");

txtPrecio.requestFocus();

}

else if (leerStock() < 0) {

Libreria.mensajeAdvertencia(this, "Ingrese el STOCK de forma correcta");

txtStock.setText("");

txtStock.requestFocus();

}

else {

Medicamentos nuevo = new Medicamentos(leerCodigo(), leerNombre(), leerLaboratorio(), leerPrecio(),

leerStock());

am.adicionar(nuevo);

listado();

habilitarEntradas(false);

habilitarOperaciones(true);

}

}

void consultarMedicamento() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Paciente");

txtCodigo.requestFocus();

}

else {

Medicamentos x = am.buscar(leerCodigo());

if (x != null) {

txtNombre.setText(x.getNombre());

txtLaboratorio.setText(x.getLaboratorio());

txtPrecio.setText("" + x.getPrecio());

txtStock.setText("" + x.getStock());

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

limpieza();

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void modificarMedicamento() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Medicamento");

txtCodigo.requestFocus();

}

else {

Medicamentos x = am.buscar(leerCodigo());

if (x != null)

if (leerNombre().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó el NOMBRE");

txtNombre.requestFocus();

}

else if (leerLaboratorio().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó el LABORATORIO");

txtLaboratorio.setText("");

txtLaboratorio.requestFocus();

}

else if (leerPrecio() < 10) {

Libreria.mensajeAdvertencia(this, "Ingrese el PRECIO de forma correcta");

txtPrecio.setText("");

txtPrecio.requestFocus();

}

else if (leerStock() < 0) {

Libreria.mensajeAdvertencia(this, "Ingrese el STOCK de forma correcta");

txtStock.setText("");

txtStock.requestFocus();

}

else {

x.setNombre(leerNombre());

x.setLaboratorio(leerLaboratorio());

x.setPrecio(leerPrecio());

x.setStock(leerStock());

listado();

habilitarBusqueda(false);

habilitarEntradas(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void eliminarMedicamento() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del EMPLEADO");

txtCodigo.requestFocus();

}

else {

Medicamentos x = am.buscar(leerCodigo());

if (x != null) {

am.eliminar(x);

listado();

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

// Métodos tipo void con parámetros

void habilitarBusqueda(boolean sino) {

btnBuscar.setEnabled(sino);

txtCodigo.setEditable(sino);

if (sino)

txtCodigo.requestFocus();

}

void habilitarEntradas(boolean sino) {

if (!sino)

limpieza();

txtNombre.setEditable(sino);

txtCodigo.setEditable(sino);

txtLaboratorio.setEditable(sino);

;

txtPrecio.setEditable(sino);

txtStock.setEditable(sino);

}

void habilitarOperaciones(boolean sino) {

btnAdicionar.setEnabled(sino);

btnConsultar.setEnabled(sino);

btnModificar.setEnabled(sino);

btnEliminar.setEnabled(sino);

btnAceptar.setEnabled(!sino);

btnCancelar.setEnabled(!sino);

}

// Métodos que retornan valor sin parámetros

int cantidadaVender() {

int cant = -1;

try {

cant = Integer.parseInt(JOptionPane.showInputDialog("Ingrese la cantidad:"));

}

catch (Exception e2) {

}

return cant;

}

int leerCodigo() {

return Libreria.leerEntero(txtCodigo);

}

String leerNombre() {

return txtNombre.getText().trim();

}

String leerLaboratorio() {

return txtLaboratorio.getText().trim();

}

double leerPrecio() {

return Libreria.leerReal(txtPrecio);

}

int leerStock() {

return Libreria.leerEntero(txtStock);

}

}

DlgPaciente

package gui;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JScrollPane;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import javax.swing.table.TableRowSorter;

import org.eclipse.wb.swing.FocusTraversalOnArray;

import arreglos.ArregloPaciente;

import clases.Paciente;

import libreria.DiseñoObjetos;

import libreria.Libreria;

import libreria.Validaciones;

import java.awt.Color;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.KeyEvent;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.RowFilter;

import javax.swing.JRadioButton;

import java.awt.Component;

public class DlgPaciente extends JDialog implements ActionListener, KeyListener {

private static final long serialVersionUID = 1L;

private JLabel lblCdigo;

private JTextField txtCodigo;

private JButton btnBuscar;

private JTextField txtNombres;

private JTextField txtApellidos;

private JTextField txtTelefono;

private JTextField txtDni;

private JScrollPane scrollPane;

public JButton btnAdicionar;

public JButton btnConsultar;

public JButton btnModificar;

public JButton btnEliminar;

private JButton btnAceptar;

private JButton btnCancelar;

private JLabel lblNombres;

private JLabel lblApellidos;

private JLabel lblTelfono;

private JLabel lblDni;

private JLabel lblMensaje;

private JTable tblPacientes;

public JButton btnGuardarPacientes;

private JTextField filtro;

private JRadioButton rdbtnApellidos;

private JRadioButton rdbtnDni;

private JLabel lblBuscarPor;

private ButtonGroup bp;

// Declaración de variables globales para obtener datos de tabla

public String codigoPaciente;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

// DECLARACIÓN GLOBAL PARA PODER FILTRAR DATOS EN TABLA

TableRowSorter<DefaultTableModel> sorter;

RowFilter<DefaultTableModel, Object> rf;

/\*

\* Tipo de operación a procesar: Adicionar, Modificar, Eliminar o Consultar

\*/

private int tipoOperacion;

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// Declaración global

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

// Declaración global de libreria Validaciones

Validaciones v = new Validaciones();

// Declaración global de libreria DiseñoObjetos

DiseñoObjetos ds = new DiseñoObjetos();

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgPaciente dialog = new DlgPaciente();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgPaciente() {

getContentPane().setBackground(Color.WHITE);

setModal(true);

setResizable(false);

setTitle("MANTENIMIENTO | PACIENTE");

setBounds(100, 100, 1013, 494);

getContentPane().setLayout(null);

lblCdigo = new JLabel("C\u00D3DIGO");

lblCdigo.setForeground(Color.BLACK);

lblCdigo.setHorizontalAlignment(SwingConstants.RIGHT);

lblCdigo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblCdigo.setBounds(15, 64, 75, 14);

getContentPane().add(lblCdigo);

txtCodigo = new JTextField();

txtCodigo.addKeyListener(this);

txtCodigo.setEditable(false);

txtCodigo.setForeground(Color.BLACK);

txtCodigo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtCodigo.setBounds(100, 57, 137, 29);

getContentPane().add(txtCodigo);

txtCodigo.setColumns(10);

btnBuscar = new JButton("BUSCAR");

btnBuscar.setBackground(new Color(102, 205, 170));

btnBuscar.setEnabled(false);

btnBuscar.addActionListener(this);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnBuscar.setBounds(247, 57, 110, 29);

ds.setCurvasButton(btnBuscar, "imagenes/buscar.png");

getContentPane().add(btnBuscar);

txtNombres = new JTextField();

txtNombres.addKeyListener(this);

txtNombres.setEditable(false);

txtNombres.setForeground(Color.BLACK);

txtNombres.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtNombres.setColumns(10);

txtNombres.setBounds(100, 97, 257, 29);

getContentPane().add(txtNombres);

txtApellidos = new JTextField();

txtApellidos.addKeyListener(this);

txtApellidos.setEditable(false);

txtApellidos.setForeground(Color.BLACK);

txtApellidos.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtApellidos.setColumns(10);

txtApellidos.setBounds(100, 137, 257, 29);

getContentPane().add(txtApellidos);

txtTelefono = new JTextField();

txtTelefono.addKeyListener(this);

txtTelefono.setEditable(false);

txtTelefono.setForeground(Color.BLACK);

txtTelefono.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtTelefono.setColumns(10);

txtTelefono.setBounds(100, 177, 137, 29);

getContentPane().add(txtTelefono);

txtDni = new JTextField();

txtDni.addKeyListener(this);

txtDni.setEditable(false);

txtDni.setForeground(Color.BLACK);

txtDni.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtDni.setColumns(10);

txtDni.setBounds(100, 217, 137, 29);

getContentPane().add(txtDni);

// ASIGNANDO MODELO DE TABLA

dtm = new DefaultTableModel(null, getColumnas());

// ESTABLECIENDO A TABLEROWSORTER EL MODELO DE LA TABLA

sorter = new TableRowSorter<DefaultTableModel>(dtm);

// ESTABLECIMIENTO NULL ROWFILTER

rf = null;

tblPacientes = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

tblPacientes.setForeground(Color.BLACK);

tblPacientes.setFont(new Font("Arial", Font.BOLD, 14));

tblPacientes.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblPacientes.setRowHeight(25);

;

scrollPane = new JScrollPane(tblPacientes);

scrollPane.setBounds(367, 97, 630, 360);

getContentPane().add(scrollPane);

tblPacientes.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

if (e.getClickCount() == 2) {

int row = tblPacientes.getSelectedRow();

if (row < 0) {

}

else {

codigoPaciente = tblPacientes.getValueAt(row, 0).toString();

dispose();

}

}

}

}

});

tblPacientes.setAutoCreateRowSorter(true);

tblPacientes.setRowSorter(sorter);

btnAdicionar = new JButton("ADICIONAR");

btnAdicionar.setBackground(new Color(124, 252, 0));

btnAdicionar.addActionListener(this);

btnAdicionar.setForeground(Color.BLACK);

btnAdicionar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAdicionar.setBounds(39, 268, 137, 39);

ds.setCurvasButton(btnAdicionar, "imagenes/adicionar.png");

getContentPane().add(btnAdicionar);

btnConsultar = new JButton("CONSULTAR");

btnConsultar.setBackground(Color.CYAN);

btnConsultar.addActionListener(this);

btnConsultar.setForeground(Color.BLACK);

btnConsultar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnConsultar.setBounds(196, 268, 145, 39);

ds.setCurvasButton(btnConsultar, "imagenes/consultar.png");

getContentPane().add(btnConsultar);

btnModificar = new JButton("MODIFICAR");

btnModificar.setBackground(new Color(0, 139, 139));

btnModificar.addActionListener(this);

btnModificar.setForeground(Color.BLACK);

btnModificar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnModificar.setBounds(196, 318, 145, 39);

ds.setCurvasButton(btnModificar, "imagenes/modificar.png");

getContentPane().add(btnModificar);

btnEliminar = new JButton("ELIMINAR");

btnEliminar.setBackground(new Color(127, 255, 212));

btnEliminar.addActionListener(this);

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnEliminar.setBounds(39, 318, 137, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

getContentPane().add(btnEliminar);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.setBackground(new Color(30, 144, 255));

btnAceptar.setEnabled(false);

btnAceptar.addActionListener(this);

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnAceptar.setBounds(39, 368, 137, 39);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

getContentPane().add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setEnabled(false);

btnCancelar.addActionListener(this);

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnCancelar.setBounds(196, 368, 145, 39);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

getContentPane().add(btnCancelar);

lblNombres = new JLabel("NOMBRES");

lblNombres.setForeground(Color.BLACK);

lblNombres.setHorizontalAlignment(SwingConstants.RIGHT);

lblNombres.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblNombres.setBounds(15, 104, 75, 14);

getContentPane().add(lblNombres);

lblApellidos = new JLabel("APELLIDOS");

lblApellidos.setForeground(Color.BLACK);

lblApellidos.setHorizontalAlignment(SwingConstants.RIGHT);

lblApellidos.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblApellidos.setBounds(15, 144, 75, 14);

getContentPane().add(lblApellidos);

lblTelfono = new JLabel("TEL\u00C9FONO");

lblTelfono.setForeground(Color.BLACK);

lblTelfono.setHorizontalAlignment(SwingConstants.RIGHT);

lblTelfono.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblTelfono.setBounds(15, 184, 75, 14);

getContentPane().add(lblTelfono);

lblDni = new JLabel("DNI");

lblDni.setForeground(Color.BLACK);

lblDni.setHorizontalAlignment(SwingConstants.RIGHT);

lblDni.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblDni.setBounds(15, 224, 75, 14);

getContentPane().add(lblDni);

lblMensaje = new JLabel("");

lblMensaje.setForeground(Color.WHITE);

lblMensaje.setBackground(Color.DARK\_GRAY);

lblMensaje.setHorizontalAlignment(SwingConstants.CENTER);

lblMensaje.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 18));

lblMensaje.setOpaque(true);

lblMensaje.setBounds(15, 11, 342, 29);

getContentPane().add(lblMensaje);

btnGuardarPacientes = new JButton("GUARDAR PACIENTES");

btnGuardarPacientes.addActionListener(this);

btnGuardarPacientes.setForeground(Color.BLACK);

btnGuardarPacientes.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnGuardarPacientes.setBackground(new Color(0, 250, 154));

btnGuardarPacientes.setBounds(39, 418, 302, 39);

ds.setCurvasButton(btnGuardarPacientes, "imagenes/grabar.png");

getContentPane().add(btnGuardarPacientes);

filtro = new JTextField();

filtro.addKeyListener(this);

filtro.setHorizontalAlignment(SwingConstants.CENTER);

filtro.setForeground(Color.BLACK);

filtro.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

filtro.setColumns(10);

filtro.setBounds(556, 57, 257, 29);

getContentPane().add(filtro);

rdbtnApellidos = new JRadioButton("APELLIDOS");

rdbtnApellidos.addActionListener(this);

rdbtnApellidos.setSelected(true);

rdbtnApellidos.setBounds(606, 32, 97, 23);

getContentPane().add(rdbtnApellidos);

rdbtnDni = new JRadioButton("DNI");

rdbtnDni.addActionListener(this);

rdbtnDni.setBounds(715, 32, 60, 23);

getContentPane().add(rdbtnDni);

bp = new ButtonGroup();

bp.add(rdbtnApellidos);

bp.add(rdbtnDni);

lblBuscarPor = new JLabel("BUSCAR POR");

lblBuscarPor.setHorizontalAlignment(SwingConstants.RIGHT);

lblBuscarPor.setForeground(Color.BLACK);

lblBuscarPor.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblBuscarPor.setBounds(637, 11, 97, 14);

getContentPane().add(lblBuscarPor);

listado();

modeloTabla();

setLocationRelativeTo(this);

setFocusTraversalPolicy(new FocusTraversalOnArray(

new Component[] { txtCodigo, txtNombres, txtApellidos, txtTelefono, txtDni }));

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == rdbtnDni) {

actionPerformedRdbtnDni(arg0);

}

if (arg0.getSource() == rdbtnApellidos) {

actionPerformedRdbtnApellidos(arg0);

}

if (arg0.getSource() == btnGuardarPacientes) {

actionPerformedBtnGuardarPacientes(arg0);

}

if (arg0.getSource() == btnCancelar) {

actionPerformedBtnCancelar(arg0);

}

if (arg0.getSource() == btnAceptar) {

actionPerformedBtnAceptar(arg0);

}

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

if (arg0.getSource() == btnEliminar) {

actionPerformedBtnEliminar(arg0);

}

if (arg0.getSource() == btnModificar) {

actionPerformedBtnModificar(arg0);

}

if (arg0.getSource() == btnConsultar) {

actionPerformedBtnConsultar(arg0);

}

if (arg0.getSource() == btnAdicionar) {

actionPerformedBtnAdicionar(arg0);

}

}

protected void actionPerformedBtnAdicionar(ActionEvent arg0) {

tipoOperacion = ADICIONAR;

lblMensaje.setText("Adicionando Paciente");

txtCodigo.setText("" + ap.codigoCorrelativo());

habilitarEntradas(true);

habilitarOperaciones(false);

txtCodigo.setEditable(false);

txtNombres.requestFocus();

}

protected void actionPerformedBtnConsultar(ActionEvent arg0) {

tipoOperacion = CONSULTAR;

lblMensaje.setText("Consultando Paciente");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnModificar(ActionEvent arg0) {

tipoOperacion = MODIFICAR;

lblMensaje.setText("Modificando Paciente");

habilitarBusqueda(true);

habilitarEntradas(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

tipoOperacion = ELIMINAR;

lblMensaje.setText("Eliminando Paciente");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

consultarPaciente();

}

protected void actionPerformedBtnAceptar(ActionEvent arg0) {

switch (tipoOperacion) {

case ADICIONAR:

adicionarPaciente();

lblMensaje.setText("");

break;

case CONSULTAR:

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

lblMensaje.setText("");

break;

case MODIFICAR:

modificarPaciente();

lblMensaje.setText("");

break;

case ELIMINAR:

eliminarPaciente();

lblMensaje.setText("");

}

}

protected void actionPerformedBtnCancelar(ActionEvent arg0) {

if (tipoOperacion != ADICIONAR)

habilitarBusqueda(false);

lblMensaje.setText("");

habilitarEntradas(false);

habilitarOperaciones(true);

}

protected void actionPerformedBtnGuardarPacientes(ActionEvent arg0) {

if (ap.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + ap.getArchivo() + "\" ?");

if (ok == 0) {

ap.grabarPacientes();

Libreria.mensajeInformacion(this, "\"" + ap.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + ap.getArchivo() + "\"");

}

else {

ap.grabarPacientes();

Libreria.mensajeInformacion(this, "\"" + ap.getArchivo() + "\" ha sido creado");

}

}

protected void actionPerformedRdbtnApellidos(ActionEvent arg0) {

filtro.setText("");

filtro.requestFocus();

}

protected void actionPerformedRdbtnDni(ActionEvent arg0) {

filtro.setText("");

filtro.requestFocus();

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

if (arg0.getSource() == filtro) {

keyReleasedFiltro(arg0);

}

}

public void keyTyped(KeyEvent arg0) {

if (arg0.getSource() == txtDni) {

keyTypedTxtDni(arg0);

}

if (arg0.getSource() == txtTelefono) {

keyTypedTxtTelefono(arg0);

}

if (arg0.getSource() == txtApellidos) {

keyTypedTxtApellidos(arg0);

}

if (arg0.getSource() == txtNombres) {

keyTypedTxtNombres(arg0);

}

if (arg0.getSource() == txtCodigo) {

keyTypedTxtCodigo(arg0);

}

}

protected void keyReleasedFiltro(KeyEvent arg0) {

filtro.setText(filtro.getText().toUpperCase());

if (rdbtnApellidos.isSelected()) {

rf = RowFilter.regexFilter(filtro.getText(), 2);

}

else {

rf = RowFilter.regexFilter(filtro.getText(), 4);

}

sorter.setRowFilter(rf);

}

protected void keyTypedTxtCodigo(KeyEvent e) {

Validaciones.soloNumero(e, txtCodigo, 6);

char t = e.getKeyChar();

if (t == KeyEvent.VK\_ENTER) {

consultarPaciente();

}

}

protected void keyTypedTxtNombres(KeyEvent e) {

Validaciones.soloLetras(e, txtNombres, 20);

}

protected void keyTypedTxtApellidos(KeyEvent e) {

Validaciones.soloLetras(e, txtApellidos, 20);

}

protected void keyTypedTxtTelefono(KeyEvent e) {

Validaciones.soloNumero(e, txtTelefono, 9);

}

protected void keyTypedTxtDni(KeyEvent e) {

Validaciones.soloNumero(e, txtDni, 8);

}

// Métodos tipo void sin parámetros

void modeloTabla() {

TableColumnModel tc = tblPacientes.getColumnModel();

tc.getColumn(0).setPreferredWidth(50);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(100);

tc.getColumn(4).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tc.getColumn(4).setCellRenderer(tcr);

}

void listado() {

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

dtm.removeRow(i);

i--;

}

if (ap.tamaño() == 0) {

}

else {

for (int i = 0; i < ap.tamaño(); i++) {

Paciente x = ap.obtener(i);

dtm.addRow(new Object[] { x.getCodpaciente(), x.getNombres().toUpperCase(),

x.getApellidos().toUpperCase(), x.getTelefono(), x.getDNI() });

}

}

}

void limpieza() {

txtCodigo.setText("");

txtNombres.setText("");

txtApellidos.setText("");

txtTelefono.setText("");

txtDni.setText("");

}

void adicionarPaciente() {

if (leerNombres().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su NOMBRE");

txtNombres.requestFocus();

}

else if (leerApellidos().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su APELLIDO");

txtApellidos.setText("");

txtApellidos.requestFocus();

}

else if (leerTelefono().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su TELÉFONO");

txtTelefono.setText("");

txtTelefono.requestFocus();

}

else if (leerDni().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su DNI");

txtDni.setText("");

txtDni.requestFocus();

}

else if (validarDni()) {

Libreria.mensajeAdvertencia(this, "El DNI que ingresó ya existe");

txtDni.setText("");

txtDni.requestFocus();

}

else {

Paciente nuevo = new Paciente(leerCodigo(), leerApellidos(), leerNombres(), leerTelefono(), leerDni());

ap.adicionar(nuevo);

listado();

habilitarEntradas(false);

habilitarOperaciones(true);

}

}

void consultarPaciente() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Paciente");

txtCodigo.requestFocus();

}

else {

Paciente x = ap.buscar(leerCodigo());

if (x != null) {

txtNombres.setText(x.getNombres().toUpperCase());

txtApellidos.setText(x.getApellidos().toUpperCase());

txtTelefono.setText(x.getTelefono());

txtDni.setText(x.getDNI());

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

limpieza();

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void modificarPaciente() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Paciente");

txtCodigo.requestFocus();

}

else {

Paciente x = ap.buscar(leerCodigo());

if (x != null)

if (leerNombres().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su NOMBRE");

txtNombres.requestFocus();

}

else if (leerApellidos().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su APELLIDO");

txtApellidos.setText("");

txtApellidos.requestFocus();

}

else if (leerTelefono().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su TELÉFONO");

txtTelefono.setText("");

txtTelefono.requestFocus();

}

else if (leerDni().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó su DNI");

txtDni.setText("");

txtDni.requestFocus();

}

else if (validarDni()) {

Libreria.mensajeAdvertencia(this, "El DNI que ingresó ya existe");

txtDni.setText("");

txtDni.requestFocus();

}

else {

x.setNombres(leerNombres());

x.setApellidos(leerApellidos());

x.setTelefono(leerTelefono());

x.setDNI(leerDni());

listado();

habilitarBusqueda(false);

habilitarEntradas(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void eliminarPaciente() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Paciente");

txtCodigo.requestFocus();

}

else {

Paciente x = ap.buscar(leerCodigo());

if (x != null) {

ap.eliminar(x);

listado();

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

// Métodos tipo void con parámetros

void habilitarBusqueda(boolean sino) {

btnBuscar.setEnabled(sino);

txtCodigo.setEditable(sino);

if (sino)

txtCodigo.requestFocus();

}

void habilitarEntradas(boolean sino) {

if (!sino)

limpieza();

txtNombres.setEditable(sino);

txtApellidos.setEditable(sino);

txtTelefono.setEditable(sino);

txtDni.setEditable(sino);

txtCodigo.setEditable(sino);

}

void habilitarOperaciones(boolean sino) {

btnAdicionar.setEnabled(sino);

btnConsultar.setEnabled(sino);

btnModificar.setEnabled(sino);

btnEliminar.setEnabled(sino);

btnAceptar.setEnabled(!sino);

btnCancelar.setEnabled(!sino);

}

// Métodos que retornan valor sin parámetros

String[] getColumnas() {

String columnas[] = new String[] { "CODIGO", "NOMBRES", "APELLIDOS", "TELEFONO", "DNI" };

return columnas;

}

boolean validarDni() {

Paciente p;

for (int i = 0; i < ap.tamaño(); i++) {

p = ap.obtener(i);

if (p.getCodpaciente() != leerCodigo()) {

if (p.getDNI().equals(leerDni())) {

return true;

}

}

else {

if (p.getDNI().equals(leerDni())) {

return false;

}

}

}

return false;

}

int leerCodigo() {

return Libreria.leerEntero(txtCodigo);

}

String leerNombres() {

return Libreria.leerCadena(txtNombres);

}

String leerApellidos() {

return Libreria.leerCadena(txtApellidos);

}

String leerTelefono() {

return Libreria.leerCadena(txtTelefono);

}

String leerDni() {

return Libreria.leerCadena(txtDni);

}

}

DlgRegistroConsumo

package gui;

import javax.swing.JDialog;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloConsumo;

import arreglos.ArregloDetalleConsumo;

import arreglos.ArregloMedicamento;

import clases.Consumo;

import clases.DetalleConsumo;

import clases.Medicamentos;

import libreria.DiseñoObjetos;

import libreria.Fecha;

import libreria.Libreria;

import javax.swing.JTextField;

import java.awt.Color;

import javax.swing.JScrollPane;

import javax.swing.JComboBox;

import javax.swing.JTable;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JPanel;

public class DlgRegistroConsumo extends JDialog implements ActionListener {

private static final long serialVersionUID = 1L;

private JLabel lblCodigoConsumo;

private JTextField txtCodigoConsumo;

private JScrollPane scrollPane;

private JTextField txtTotal;

private JLabel lblTotal;

private JComboBox<String> cboDia;

private JComboBox<String> cboMes;

private JComboBox<String> cboAno;

private JLabel lblFecha;

private JTextField txtCodigoPaciente;

private JLabel lblCodigoPaciente;

private JTable jtblConsumo;

private JButton btnBuscarPaciente;

private JButton btnSeleccionarMedicamento;

private JButton btnSeleccionarServicio;

private JButton btnEliminar;

private JButton btnRegistrar;

private JPanel panel;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

// Declaración global para arreglos

ArregloConsumo ac = new ArregloConsumo("consumos.txt");

ArregloDetalleConsumo adet = new ArregloDetalleConsumo("detallesconsumos.txt");

ArregloMedicamento am = new ArregloMedicamento("medicamentos.txt");

// Declaración global para libreria

Fecha f = new Fecha();

DiseñoObjetos ds = new DiseñoObjetos();

// Declaración variable global para la suma total de la venta

private double total = 0.0;

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgRegistroConsumo dialog = new DlgRegistroConsumo();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgRegistroConsumo() {

setModal(true);

setResizable(false);

setTitle("Registro de Consumo");

getContentPane().setBackground(Color.WHITE);

setBounds(100, 100, 754, 552);

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 149, 728, 309);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblConsumo = new JTable(dtm) {

public static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblConsumo.setFont(new Font("Arial", Font.BOLD, 14));

jtblConsumo.setForeground(Color.BLACK);

jtblConsumo.setRowHeight(25);

scrollPane.setViewportView(jtblConsumo);

txtTotal = new JTextField();

txtTotal.setText("S/.00.00");

txtTotal.setBackground(Color.WHITE);

txtTotal.setEditable(false);

txtTotal.setHorizontalAlignment(SwingConstants.RIGHT);

txtTotal.setForeground(Color.RED);

txtTotal.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 18));

txtTotal.setColumns(10);

txtTotal.setBounds(619, 469, 119, 39);

getContentPane().add(txtTotal);

lblTotal = new JLabel("TOTAL :");

lblTotal.setForeground(Color.BLACK);

lblTotal.setHorizontalAlignment(SwingConstants.RIGHT);

lblTotal.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblTotal.setBounds(462, 482, 147, 14);

getContentPane().add(lblTotal);

btnEliminar = new JButton("ELIMINAR PRODUCTO DE LISTA");

btnEliminar.addActionListener(this);

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnEliminar.setBackground(new Color(135, 206, 235));

btnEliminar.setBounds(60, 470, 263, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

getContentPane().add(btnEliminar);

btnRegistrar = new JButton("REGISTRAR");

btnRegistrar.addActionListener(this);

btnRegistrar.setForeground(Color.BLACK);

btnRegistrar.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnRegistrar.setBackground(new Color(0, 255, 127));

btnRegistrar.setBounds(358, 470, 147, 39);

ds.setCurvasButton(btnRegistrar, "imagenes/aceptar.png");

getContentPane().add(btnRegistrar);

panel = new JPanel();

panel.setBorder(UIManager.getBorder("ScrollPane.border"));

panel.setBackground(Color.WHITE);

panel.setBounds(10, 5, 728, 133);

panel.setBorder(BorderFactory.createLineBorder(Color.RED, 2, true));

getContentPane().add(panel);

panel.setLayout(null);

lblCodigoConsumo = new JLabel("CODIGO CONSUMO :");

lblCodigoConsumo.setBounds(10, 18, 124, 14);

panel.add(lblCodigoConsumo);

lblCodigoConsumo.setForeground(Color.BLACK);

lblCodigoConsumo.setHorizontalAlignment(SwingConstants.RIGHT);

lblCodigoConsumo.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

lblCodigoPaciente = new JLabel("CODIGO PACIENTE :");

lblCodigoPaciente.setBounds(10, 58, 124, 14);

panel.add(lblCodigoPaciente);

lblCodigoPaciente.setForeground(Color.BLACK);

lblCodigoPaciente.setHorizontalAlignment(SwingConstants.RIGHT);

lblCodigoPaciente.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

txtCodigoConsumo = new JTextField();

txtCodigoConsumo.setBounds(144, 11, 119, 29);

panel.add(txtCodigoConsumo);

txtCodigoConsumo.setBackground(Color.WHITE);

txtCodigoConsumo.setForeground(Color.BLACK);

txtCodigoConsumo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigoConsumo.setEditable(false);

txtCodigoConsumo.setColumns(10);

txtCodigoPaciente = new JTextField();

txtCodigoPaciente.setBounds(144, 51, 119, 29);

panel.add(txtCodigoPaciente);

txtCodigoPaciente.setBackground(Color.WHITE);

txtCodigoPaciente.setForeground(Color.BLACK);

txtCodigoPaciente.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigoPaciente.setEditable(false);

txtCodigoPaciente.setColumns(10);

btnBuscarPaciente = new JButton("");

btnBuscarPaciente.setBounds(273, 51, 47, 29);

panel.add(btnBuscarPaciente);

btnBuscarPaciente.addActionListener(this);

btnBuscarPaciente.setForeground(Color.BLACK);

btnBuscarPaciente.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

ds.setCurvasButton(btnBuscarPaciente, "imagenes/buscar.png");

btnBuscarPaciente.setBackground(new Color(144, 238, 144));

btnSeleccionarServicio = new JButton("Seleccionar Servicio");

btnSeleccionarServicio.addActionListener(this);

btnSeleccionarServicio.setBounds(251, 91, 195, 29);

panel.add(btnSeleccionarServicio);

btnSeleccionarServicio.setForeground(Color.BLACK);

btnSeleccionarServicio.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnSeleccionarServicio.setBackground(new Color(0, 128, 0));

ds.setCurvasButton(btnSeleccionarServicio, "imagenes/servicios.png");

btnSeleccionarMedicamento = new JButton("Seleccionar Medicamento");

btnSeleccionarMedicamento.setBounds(10, 91, 231, 29);

panel.add(btnSeleccionarMedicamento);

btnSeleccionarMedicamento.addActionListener(this);

btnSeleccionarMedicamento.setForeground(Color.BLACK);

btnSeleccionarMedicamento.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

btnSeleccionarMedicamento.setBackground(new Color(0, 255, 255));

ds.setCurvasButton(btnSeleccionarMedicamento, "imagenes/medicamentos.png");

cboAno = new JComboBox<String>();

cboAno.setBounds(633, 11, 85, 29);

panel.add(cboAno);

cboAno.setForeground(Color.BLACK);

cboAno.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboAno.setEnabled(false);

Fecha.colocarItems(cboAno, Fecha.añoActual(), 2000);

cboMes = new JComboBox<String>();

cboMes.setBounds(516, 11, 107, 29);

panel.add(cboMes);

cboMes.setForeground(Color.BLACK);

cboMes.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboMes.setEnabled(false);

Fecha.colocarMeses(cboMes);

Fecha.colocarMesActual(cboMes);

cboDia = new JComboBox<String>();

cboDia.setBounds(456, 11, 50, 29);

panel.add(cboDia);

cboDia.setForeground(Color.BLACK);

cboDia.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboDia.setEnabled(false);

Fecha.colocarItems(cboDia, 1, 31);

Fecha.colocarDiaActual(cboDia);

lblFecha = new JLabel("FECHA :");

lblFecha.setBounds(362, 18, 84, 14);

panel.add(lblFecha);

lblFecha.setForeground(Color.BLACK);

lblFecha.setHorizontalAlignment(SwingConstants.RIGHT);

lblFecha.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 14));

setCodigoConsumo();

modeloTabla();

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnSeleccionarServicio) {

actionPerformedBtnSeleccionarServicio(arg0);

}

if (arg0.getSource() == btnRegistrar) {

actionPerformedBtnRegistrar(arg0);

}

if (arg0.getSource() == btnEliminar) {

actionPerformedBtnEliminar(arg0);

}

if (arg0.getSource() == btnSeleccionarMedicamento) {

actionPerformedBtnSeleccionarMedicamento(arg0);

}

if (arg0.getSource() == btnBuscarPaciente) {

actionPerformedBtnBuscarPaciente(arg0);

}

}

protected void actionPerformedBtnBuscarPaciente(ActionEvent arg0) {

DlgHospitalizacion hos = new DlgHospitalizacion();

hos.setTitle("PACIENTES HOSPITALIZADOS");

hos.btnBuscarPaciente.setEnabled(false);

hos.rdbtnEconomico.setEnabled(false);

hos.rdbtnEjecutivo.setEnabled(false);

hos.cboNroCamas.setEnabled(false);

hos.btnAceptar.setEnabled(false);

hos.btnCancelar.setEnabled(false);

hos.btnGrabarHospitalizacion.setEnabled(false);

hos.rdbtnFiltrarPagado.setEnabled(false);

hos.rdbtnFiltrarTodo.setEnabled(false);

hos.setVisible(true);

txtCodigoPaciente.setText(hos.codigoPaciente);

}

protected void actionPerformedBtnSeleccionarServicio(ActionEvent arg0) {

DlgServicios ser = new DlgServicios();

ser.btnAdicionar.setEnabled(false);

ser.btnConsultar.setEnabled(false);

ser.btnEliminar.setEnabled(false);

ser.btnModificar.setEnabled(false);

ser.btnGuardarServicios.setEnabled(false);

ser.setVisible(true);

if (!ser.codServicio.equals("") && ser.precioServicio > 0) {

if (jtblConsumo.getRowCount() == 0) {

dtm.addRow(new Object[] { ser.codServicio, ser.nomServicio, 1,

Libreria.formatoDecimales(ser.precioServicio),

Libreria.formato2Decimales(ser.precioServicio) });

total += ser.precioServicio;

txtTotal.setText(Libreria.formatoSoles(total));

}

else {

int c = 0;

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

if (ser.codServicio.equals(jtblConsumo.getValueAt(i, 0).toString())) {

c++;

break;

}

}

if (c == 0) {

dtm.addRow(new Object[] { ser.codServicio, ser.nomServicio, 1,

Libreria.formatoDecimales(ser.precioServicio),

Libreria.formato2Decimales(ser.precioServicio) });

total += ser.precioServicio;

txtTotal.setText(Libreria.formatoSoles(total));

}

else {

Libreria.mensajeInformacion(this, "El SERVICIO ya fue agregado a la lista.");

}

}

}

}

protected void actionPerformedBtnSeleccionarMedicamento(ActionEvent arg0) {

DlgMedicamento med = new DlgMedicamento();

med.btnAdicionar.setEnabled(false);

med.btnConsultar.setEnabled(false);

med.btnEliminar.setEnabled(false);

med.btnModificar.setEnabled(false);

med.btnGuardarMedicamentos.setEnabled(false);

med.setVisible(true);

if (!med.nombreProducto.equals("") && med.precioProducto > 0 && med.cantidadVender > 0) {

if (jtblConsumo.getRowCount() == 0) {

double subtotal = med.precioProducto \* med.cantidadVender;

dtm.addRow(new Object[] { med.codigoProducto, med.nombreProducto, med.cantidadVender,

med.precioProducto, Libreria.formato2Decimales(subtotal) });

total += subtotal;

txtTotal.setText(Libreria.formatoSoles(total));

}

else {

int c = 0;

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

if (med.codigoProducto.equals(jtblConsumo.getValueAt(i, 0).toString())) {

c++;

break;

}

}

if (c == 0) {

double subtotal = med.precioProducto \* med.cantidadVender;

dtm.addRow(new Object[] { med.codigoProducto, med.nombreProducto, med.cantidadVender,

med.precioProducto, Libreria.formato2Decimales(subtotal) });

total += subtotal;

txtTotal.setText(Libreria.formatoSoles(total));

}

else {

Libreria.mensajeInformacion(this, "El PRODUCTO ya fue agregado a la lista.");

}

}

}

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

eliminarProductoDeLista();

}

protected void actionPerformedBtnRegistrar(ActionEvent arg0) {

if (leerCodConsumo() == -1) {

Libreria.mensajeInformacion(this, "EL CAMPO CÓDIGO DE CONSUMO NO PUEDE ESTAR VACÍO");

}

else if (leerCodPaciente() == -1) {

Libreria.mensajeInformacion(this, "DEBE SELECCIONAR EL CÓDIGO DEL PACIENTE");

}

else {

if (jtblConsumo.getRowCount() != 0) {

int msj = Libreria.confirmacion(this, "¿Desea registrar el detalle de consumo?");

if (msj == 0) {

ac.adicionar(new Consumo(leerCodConsumo(), leerCodPaciente(), Fecha.getFecha(cboDia, cboMes, cboAno), 0,

total));

ac.grabarConsumo();

int codProducto = 0, cantidad = 0;

double precioUnitario = 0.0, subtotal = 0.0;

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

codProducto = Integer.parseInt(jtblConsumo.getValueAt(i, 0).toString());

cantidad = Integer.parseInt(jtblConsumo.getValueAt(i, 2).toString());

precioUnitario = Double.parseDouble(jtblConsumo.getValueAt(i, 3).toString());

subtotal = cantidad \* precioUnitario;

adet.adicionar(

new DetalleConsumo(leerCodConsumo(), codProducto, cantidad, precioUnitario, subtotal));

}

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

Medicamentos med;

for (int j = 0; j < am.tamaño(); j++) {

med = am.obtener(j);

if (med.getCodMedicamento() == Integer.parseInt(jtblConsumo.getValueAt(i, 0).toString())){

med.restarStock(Integer.parseInt(jtblConsumo.getValueAt(i, 2).toString()));

}

}

}

am.grabarMedicamentos();

adet.grabarDetalleConsumo();

nuevoRegistroConsumo();

}

}

}

}

// Métodos void sin parámetros

void nuevoRegistroConsumo() {

setCodigoConsumo();

txtCodigoPaciente.setText("");

Libreria.limpiarJTable(jtblConsumo, dtm);

total = 0;

txtTotal.setText("S/.00.00");

}

void setCodigoConsumo() {

txtCodigoConsumo.setText("" + ac.codigoCorrelativo());

}

void modeloTabla() {

TableColumnModel tc = jtblConsumo.getColumnModel();

tc.getColumn(0).setPreferredWidth(100);

tc.getColumn(1).setPreferredWidth(200);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(4).setCellRenderer(tcr);

tc.getColumn(2).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

}

void eliminarProductoDeLista() {

int row = jtblConsumo.getSelectedRow();

if (jtblConsumo.getRowCount() == 0) {

Libreria.mensajeInformacion(this, "La lista está vacía.");

}

else if (row < 0) {

Libreria.mensajeInformacion(this, "Seleccione el producto a eliminar.");

}

else {

double precio, subtotal;

int cantidad;

cantidad = Integer.parseInt(jtblConsumo.getValueAt(row, 2).toString());

precio = Double.parseDouble(jtblConsumo.getValueAt(row, 3).toString());

subtotal = precio \* cantidad;

total -= subtotal;

txtTotal.setText(Libreria.formatoSoles(total));

dtm.removeRow(row);

}

}

// Métodos que retornar valores

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO PRODUCTO", "NOMBRE PRODUCTO", "CANTIDAD", "PRECIO", "SUBTOTAL" };

return columnas;

}

int leerCodConsumo() {

return Libreria.leerEntero(txtCodigoConsumo);

}

int leerCodPaciente() {

return Libreria.leerEntero(txtCodigoPaciente);

}

}

DlgServicios

package gui;

import java.awt.BorderLayout;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloServicio;

import clases.Servicio;

import libreria.DiseñoObjetos;

import libreria.Validaciones;

import libreria.Libreria;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.JComboBox;

import javax.swing.JScrollPane;

import java.awt.Color;

import java.awt.Font;

import java.awt.Point;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.JTable;

import java.awt.event.KeyListener;

import java.awt.event.KeyEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseEvent;

public class DlgServicios extends JDialog implements ActionListener, KeyListener, MouseListener {

private static final long serialVersionUID = 1L;

private final JPanel contentPanel = new JPanel();

private JLabel lblCodigo;

private JTextField txtCodigo;

private JLabel lblDescripcion;

private JTextField txtDescripcion;

private JLabel lblTipo;

private JComboBox<String> cboTipo;

private JLabel lblPrecio;

private JTextField txtPrecio;

private JButton btnAceptar;

private JButton btnCancelar;

public JButton btnEliminar;

public JButton btnConsultar;

public JButton btnModificar;

public JButton btnAdicionar;

private JButton btnBuscar;

private JLabel lblMensaje;

public JButton btnGuardarServicios;

private JScrollPane scrollPane;

private JTable jtblServicios;

// Declaración global para modelo de tabla

private DefaultTableModel dtm;

// Tipo de operación a procesar:

// Adicionar, Modificar, Eliminar o Consultar

private int tipoOperacion;

// Constantes para los tipos de operaciones

public final static int ADICIONAR = 0;

public final static int MODIFICAR = 1;

public final static int ELIMINAR = 2;

public final static int CONSULTAR = 3;

// declaracion global

ArregloServicio as = new ArregloServicio("servicios.txt");

// Declaración global de libreria Validaciones

Validaciones v = new Validaciones();

// Declaración global de libreria DiseñoObjetos

DiseñoObjetos ds = new DiseñoObjetos();

// Declaración global de libreria

Libreria lb = new Libreria();

//DECLARACIÓN GLOBAL PARA OBTENER CÓDIGO Y PRECIO DEL SERVICO

public String codServicio = "";

public double precioServicio = 0.0;

public String nomServicio = "";

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Throwable e) {

e.printStackTrace();

}

try {

DlgServicios dialog = new DlgServicios();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgServicios() {

setResizable(false);

setModal(true);

setTitle("MANTENIMIENTO | SERVICIO");

setBounds(100, 100, 1111, 439);

getContentPane().setLayout(new BorderLayout());

contentPanel.setForeground(Color.BLACK);

contentPanel.setBackground(Color.WHITE);

contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

getContentPane().add(contentPanel, BorderLayout.CENTER);

contentPanel.setLayout(null);

lblCodigo = new JLabel("C\u00D3DIGO");

lblCodigo.setForeground(Color.BLACK);

lblCodigo.setHorizontalAlignment(SwingConstants.RIGHT);

lblCodigo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblCodigo.setBounds(10, 56, 89, 14);

contentPanel.add(lblCodigo);

txtCodigo = new JTextField();

txtCodigo.setEditable(false);

txtCodigo.addKeyListener(this);

txtCodigo.setForeground(Color.BLACK);

txtCodigo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtCodigo.setBounds(109, 49, 96, 29);

contentPanel.add(txtCodigo);

txtCodigo.setColumns(10);

lblDescripcion = new JLabel("DESCRIPCI\u00D3N");

lblDescripcion.setForeground(Color.BLACK);

lblDescripcion.setHorizontalAlignment(SwingConstants.RIGHT);

lblDescripcion.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblDescripcion.setBounds(10, 96, 89, 14);

contentPanel.add(lblDescripcion);

txtDescripcion = new JTextField();

txtDescripcion.setEditable(false);

txtDescripcion.addKeyListener(this);

txtDescripcion.setForeground(Color.BLACK);

txtDescripcion.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtDescripcion.setBounds(109, 89, 252, 29);

contentPanel.add(txtDescripcion);

txtDescripcion.setColumns(10);

lblTipo = new JLabel("TIPO");

lblTipo.setForeground(Color.BLACK);

lblTipo.setHorizontalAlignment(SwingConstants.RIGHT);

lblTipo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblTipo.setBounds(10, 136, 88, 14);

contentPanel.add(lblTipo);

cboTipo = new JComboBox<String>();

cboTipo.setForeground(Color.BLACK);

cboTipo.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboTipo.addItem("ANÁLISIS CLÍNICOS");

cboTipo.addItem("INTERVENCIONES QUIRÚRJICAS");

cboTipo.setBounds(109, 129, 252, 29);

contentPanel.add(cboTipo);

lblPrecio = new JLabel("PRECIO");

lblPrecio.setForeground(Color.BLACK);

lblPrecio.setHorizontalAlignment(SwingConstants.RIGHT);

lblPrecio.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblPrecio.setBounds(10, 176, 88, 14);

contentPanel.add(lblPrecio);

txtPrecio = new JTextField();

txtPrecio.setEditable(false);

txtPrecio.addKeyListener(this);

txtPrecio.setForeground(Color.BLACK);

txtPrecio.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

txtPrecio.setBounds(111, 169, 94, 29);

contentPanel.add(txtPrecio);

txtPrecio.setColumns(10);

btnAceptar = new JButton("ACEPTAR");

btnAceptar.setForeground(Color.BLACK);

btnAceptar.setBackground(new Color(30, 144, 255));

btnAceptar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnAceptar.setEnabled(false);

btnAceptar.addActionListener(this);

btnAceptar.setBounds(36, 310, 146, 39);

ds.setCurvasButton(btnAceptar, "imagenes/aceptar.png");

contentPanel.add(btnAceptar);

btnCancelar = new JButton("CANCELAR");

btnCancelar.setForeground(Color.BLACK);

btnCancelar.setBackground(new Color(0, 255, 255));

btnCancelar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnCancelar.setEnabled(false);

btnCancelar.addActionListener(this);

btnCancelar.setBounds(192, 310, 145, 39);

ds.setCurvasButton(btnCancelar, "imagenes/eliminar.png");

contentPanel.add(btnCancelar);

btnEliminar = new JButton("ELIMINAR");

btnEliminar.setForeground(Color.BLACK);

btnEliminar.setBackground(new Color(127, 255, 212));

btnEliminar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnEliminar.addActionListener(this);

btnEliminar.setBounds(37, 260, 145, 39);

ds.setCurvasButton(btnEliminar, "imagenes/eliminar.png");

contentPanel.add(btnEliminar);

btnConsultar = new JButton("CONSULTAR");

btnConsultar.setForeground(Color.BLACK);

btnConsultar.setBackground(Color.CYAN);

btnConsultar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnConsultar.addActionListener(this);

btnConsultar.setBounds(192, 209, 146, 40);

ds.setCurvasButton(btnConsultar, "imagenes/consultar.png");

contentPanel.add(btnConsultar);

btnModificar = new JButton("MODIFICAR");

btnModificar.setForeground(Color.BLACK);

btnModificar.setBackground(new Color(0, 139, 139));

btnModificar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnModificar.addActionListener(this);

btnModificar.setBounds(191, 260, 146, 39);

ds.setCurvasButton(btnModificar, "imagenes/modificar.png");

contentPanel.add(btnModificar);

btnAdicionar = new JButton("ADICIONAR");

btnAdicionar.setForeground(Color.BLACK);

btnAdicionar.setBackground(new Color(124, 252, 0));

btnAdicionar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnAdicionar.addActionListener(this);

btnAdicionar.setBounds(37, 210, 145, 39);

ds.setCurvasButton(btnAdicionar, "imagenes/adicionar.png");

contentPanel.add(btnAdicionar);

btnBuscar = new JButton("BUSCAR");

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setBackground(new Color(102, 205, 170));

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnBuscar.setEnabled(false);

btnBuscar.addActionListener(this);

btnBuscar.setBounds(225, 49, 136, 29);

ds.setCurvasButton(btnBuscar, "imagenes/buscar.png");

contentPanel.add(btnBuscar);

lblMensaje = new JLabel("");

lblMensaje.setOpaque(true);

lblMensaje.setHorizontalAlignment(SwingConstants.CENTER);

lblMensaje.setForeground(Color.WHITE);

lblMensaje.setFont(new Font("Source Sans Pro Semibold", Font.BOLD, 18));

lblMensaje.setBackground(Color.DARK\_GRAY);

lblMensaje.setBounds(10, 9, 351, 29);

contentPanel.add(lblMensaje);

btnGuardarServicios = new JButton("GUARDAR SERVICIOS");

btnGuardarServicios.setForeground(Color.BLACK);

btnGuardarServicios.setBackground(new Color(0, 250, 154));

btnGuardarServicios.addActionListener(this);

btnGuardarServicios.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnGuardarServicios.setBounds(36, 360, 302, 39);

ds.setCurvasButton(btnGuardarServicios, "imagenes/grabar.png");

contentPanel.add(btnGuardarServicios);

scrollPane = new JScrollPane();

scrollPane.setBounds(371, 9, 724, 390);

contentPanel.add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblServicios = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblServicios.addMouseListener(this);

jtblServicios.setForeground(Color.BLACK);

jtblServicios.setFont(new Font("Arial", Font.BOLD, 14));

jtblServicios.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblServicios.setRowHeight(25);

scrollPane.setViewportView(jtblServicios);

listado();

modeloTabla();

this.setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnGuardarServicios) {

actionPerformedBtnGuardarServicios(arg0);

}

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

if (arg0.getSource() == btnAdicionar) {

actionPerformedBtnAdicionar(arg0);

}

if (arg0.getSource() == btnEliminar) {

actionPerformedBtnEliminar(arg0);

}

if (arg0.getSource() == btnModificar) {

actionPerformedBtnModificar(arg0);

}

if (arg0.getSource() == btnCancelar) {

actionPerformedBtnCancelar(arg0);

}

if (arg0.getSource() == btnConsultar) {

actionPerformedBtnConsultar(arg0);

}

if (arg0.getSource() == btnAceptar) {

actionPerformedBtnAceptar(arg0);

}

}

protected void actionPerformedBtnAdicionar(ActionEvent arg0) {

tipoOperacion = ADICIONAR;

lblMensaje.setText("Adicionando Servicio");

txtCodigo.setText("" + as.codigoCorrelativo());

habilitarEntradas(true);

habilitarOperaciones(false);

txtCodigo.setEditable(false);

txtDescripcion.requestFocus();

}

protected void actionPerformedBtnConsultar(ActionEvent arg0) {

tipoOperacion = CONSULTAR;

lblMensaje.setText("Consultando Servicio");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnModificar(ActionEvent arg0) {

tipoOperacion = MODIFICAR;

lblMensaje.setText("Modificando Servicio");

habilitarBusqueda(true);

habilitarEntradas(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnEliminar(ActionEvent arg0) {

tipoOperacion = ELIMINAR;

lblMensaje.setText("Eliminando Servicio");

habilitarBusqueda(true);

habilitarOperaciones(false);

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

consultarServicio();

}

protected void actionPerformedBtnAceptar(ActionEvent arg0) {

switch (tipoOperacion) {

case ADICIONAR:

adicionarEmpleado();

lblMensaje.setText("");

break;

case CONSULTAR:

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

lblMensaje.setText("");

break;

case MODIFICAR:

modificarServicio();

lblMensaje.setText("");

break;

case ELIMINAR:

eliminarEmpleado();

lblMensaje.setText("");

}

}

protected void actionPerformedBtnCancelar(ActionEvent arg0) {

if (tipoOperacion != ADICIONAR)

habilitarBusqueda(false);

lblMensaje.setText("");

habilitarEntradas(false);

habilitarOperaciones(true);

}

protected void actionPerformedBtnGuardarServicios(ActionEvent arg0) {

if (as.existeArchivo()) {

int ok = Libreria.confirmacion(this, "¿ Desea actualizar \"" + as.getArchivo() + "\" ?");

if (ok == 0) {

as.grabarServicios();

Libreria.mensajeInformacion(this, "\"" + as.getArchivo() + "\" ha sido actualizado");

}

else

Libreria.mensajeInformacion(this, "No se actualizó \"" + as.getArchivo() + "\"");

}

else {

as.grabarServicios();

Libreria.mensajeInformacion(this, "\"" + as.getArchivo() + "\" ha sido creado");

}

}

public void keyPressed(KeyEvent arg0) {

}

public void keyReleased(KeyEvent arg0) {

}

public void keyTyped(KeyEvent arg0) {

if (arg0.getSource() == txtPrecio) {

keyTypedTxtPrecio(arg0);

}

if (arg0.getSource() == txtDescripcion) {

keyTypedTxtDescripcion(arg0);

}

if (arg0.getSource() == txtCodigo) {

keyTypedTxtCodigo(arg0);

}

}

protected void keyTypedTxtCodigo(KeyEvent e) {

Validaciones.soloNumero(e, txtCodigo, 8);

char t = e.getKeyChar();

if (t == KeyEvent.VK\_ENTER)

consultarServicio();

}

protected void keyTypedTxtDescripcion(KeyEvent e) {

Validaciones.cadenaTexto(e, txtDescripcion, 20);

}

protected void keyTypedTxtPrecio(KeyEvent e) {

Validaciones.numeroReal(txtPrecio, e, 5);

}

public void mouseClicked(MouseEvent arg0) {

if (arg0.getSource() == jtblServicios) {

mouseClickedJtblServicios(arg0);

}

}

public void mouseEntered(MouseEvent arg0) {

}

public void mouseExited(MouseEvent arg0) {

}

public void mousePressed(MouseEvent arg0) {

}

public void mouseReleased(MouseEvent arg0) {

}

protected void mouseClickedJtblServicios(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

if (e.getClickCount() == 2) {

int row = jtblServicios.rowAtPoint(new Point(e.getX(), e.getY()));

if (row >= 0) {

codServicio = jtblServicios.getValueAt(row, 0).toString();

precioServicio = Double.parseDouble(jtblServicios.getValueAt(row, 3).toString());

nomServicio = jtblServicios.getValueAt(row, 1).toString();

dispose();

}

}

}

}

// Métodos tipo void sin parámetros

void limpieza() {

txtCodigo.setText("");

txtDescripcion.setText("");

txtPrecio.setText("");

cboTipo.setSelectedIndex(0);

}

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "DESCRIPCIÓN", "TIPO", "PRECIO" };

return columnas;

}

void modeloTabla() {

TableColumnModel tc = jtblServicios.getColumnModel();

tc.getColumn(0).setPreferredWidth(50);

tc.getColumn(1).setPreferredWidth(300);

tc.getColumn(2).setPreferredWidth(200);

tc.getColumn(3).setPreferredWidth(40);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(2).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(3).setCellRenderer(tcr);

}

void listado() {

Libreria.limpiarJTable(jtblServicios, dtm);

if (as.tamaño() == 0) {

}

else {

for (int i = 0; i < as.tamaño(); i++) {

Servicio x = as.obtener(i);

dtm.addRow(new Object[] { x.getCodServicio(), x.getDescripcion(), getTipoServicio(x.getTipo()),

x.getPrecio() });

}

}

}

void adicionarEmpleado() {

if (leerDescripcion().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó la DESCRIPCIÓN");

txtDescripcion.requestFocus();

}

else if (leerPrecio() < 0) {

Libreria.mensajeAdvertencia(this, "No ingresó el PRECIO");

txtPrecio.setText("");

txtPrecio.requestFocus();

}

else {

Servicio nuevo = new Servicio(leerCodigo(), leerDescripcion(), leerTipo(), leerPrecio());

as.adicionar(nuevo);

listado();

habilitarEntradas(false);

habilitarOperaciones(true);

}

}

void consultarServicio() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del SERVICIO");

txtCodigo.requestFocus();

}

else {

Servicio x = as.buscar(leerCodigo());

if (x != null) {

txtDescripcion.setText(x.getDescripcion());

cboTipo.setSelectedIndex(x.getTipo());

txtPrecio.setText("" + x.getPrecio());

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

limpieza();

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void modificarServicio() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del Empleado");

txtCodigo.requestFocus();

}

else {

Servicio x = as.buscar(leerCodigo());

if (x != null)

if (leerDescripcion().equals("")) {

Libreria.mensajeAdvertencia(this, "No ingresó la DESCRIPCIÓN");

txtDescripcion.requestFocus();

}

else if (leerPrecio() < 0) {

Libreria.mensajeAdvertencia(this, "No ingresó el PRECIO");

txtPrecio.setText("");

txtPrecio.requestFocus();

}

else {

x.setCodServicio(leerCodigo());

x.setDescripcion(leerDescripcion());

x.setPrecio(leerPrecio());

x.setTipo(leerTipo());

listado();

habilitarBusqueda(false);

habilitarEntradas(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

void eliminarEmpleado() {

int codigo = leerCodigo();

if (codigo == -1) {

Libreria.mensajeAdvertencia(this, "No ha ingresado el CÓDIGO del SERVICIO");

txtCodigo.requestFocus();

}

else {

Servicio x = as.buscar(leerCodigo());

if (x != null) {

as.eliminar(x);

listado();

limpieza();

habilitarBusqueda(false);

habilitarOperaciones(true);

}

else {

Libreria.mensajeAdvertencia(this, "El código " + leerCodigo() + " no existe");

txtCodigo.setText("");

txtCodigo.requestFocus();

}

}

}

// Métodos tipo void con parámetros

void habilitarBusqueda(boolean sino) {

btnBuscar.setEnabled(sino);

txtCodigo.setEditable(sino);

if (sino)

txtCodigo.requestFocus();

}

void habilitarEntradas(boolean sino) {

if (!sino)

limpieza();

txtDescripcion.setEditable(sino);

txtPrecio.setEditable(sino);

txtCodigo.setEditable(sino);

}

void habilitarOperaciones(boolean sino) {

btnAdicionar.setEnabled(sino);

btnConsultar.setEnabled(sino);

btnModificar.setEnabled(sino);

btnEliminar.setEnabled(sino);

btnAceptar.setEnabled(!sino);

btnCancelar.setEnabled(!sino);

}

// Métodos que retornan valor sin parámetros

int leerCodigo() {

return Libreria.leerEntero(txtCodigo);

}

String leerDescripcion() {

return Libreria.leerCadena(txtDescripcion);

}

double leerPrecio() {

return Libreria.leerReal(txtPrecio);

}

int leerTipo() {

return cboTipo.getSelectedIndex();

}

// Métodos que retornan valor con parámetros

String getTipoServicio(int tipo) {

if (tipo == 0) {

return "ANÁLISIS CLÍNICOS";

}

else {

return "INTERVENCIONES QUIRÚRJICAS";

}

}

}

DlgReporteEmpleadosHospitalizaciones

package gui;

import javax.swing.JDialog;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloEmpleado;

import arreglos.ArregloHospitalizacion;

import clases.Empleado;

import clases.Hospitalizacion;

import java.awt.Color;

import java.awt.Font;

public class DlgReporteEmpleadosHospitalizaciones extends JDialog {

private static final long serialVersionUID = 1L;

private JScrollPane scrollPane;

private JTable jtblEmpleados;

private DefaultTableModel dtm;

// DECLARACIÓN GLOBAL DE ARREGLOS HOSPITALIZACIÓN

ArregloEmpleado ae = new ArregloEmpleado("empleados.txt");

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Exception e) {

e.printStackTrace();

}

try {

DlgReporteEmpleadosHospitalizaciones dialog = new DlgReporteEmpleadosHospitalizaciones();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

public DlgReporteEmpleadosHospitalizaciones() {

getContentPane().setBackground(Color.WHITE);

setResizable(false);

setModal(true);

setBounds(100, 100, 826, 553);

setTitle("EMPLEADOS QUE REALIZARON HOSPITALIZACIONES");

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 11, 799, 503);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblEmpleados = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblEmpleados.setFont(new Font("Arial", Font.BOLD, 14));

jtblEmpleados.setForeground(Color.BLACK);

jtblEmpleados.setRowHeight(25);

jtblEmpleados.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

scrollPane.setViewportView(jtblEmpleados);

formatoTabla();

listadoPacientes();

setLocationRelativeTo(this);

}

// MÉTODOS VOID SIN PARÁMETROS

void formatoTabla() {

TableColumnModel tc = jtblEmpleados.getColumnModel();

tc.getColumn(0).setPreferredWidth(50);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(50);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

}

void listadoPacientes() {

Empleado e;

for (int j = 0; j < ae.tamaño(); j++) {

e = ae.obtener(j);

Hospitalizacion h;

for (int j2 = 0; j2 < ah.tamaño(); j2++) {

h = ah.obtener(j2);

if (h.getCodEmpl() == e.getCodEmpleado()) {

if (jtblEmpleados.getRowCount() == 0 ){

dtm.addRow(new Object[] { e.getCodEmpleado(), e.getNombres().toUpperCase(),

e.getApellidos().toUpperCase(), getTipoEmpleado(e.getTipoEmpleado()) });

}else {

int c = 0;

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

if (e.getCodEmpleado() == Integer.parseInt(jtblEmpleados.getValueAt(i, 0).toString())){

c++;

break;

}

}

if (c == 0) {

dtm.addRow(new Object[] { e.getCodEmpleado(), e.getNombres().toUpperCase(),

e.getApellidos().toUpperCase(), getTipoEmpleado(e.getTipoEmpleado()) });

}

}

}

}

}

}

// MÉTODOS CON RETORNO SIN PARÁMETROS

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO EMPLEADO", "NOMBRES", "APELLIDOS", "TIPO" };

return columnas;

}

// MÉTODOS CON RETORNO CON PARÁMETROS

String getTipoEmpleado(int tipo) {

if (tipo == 0) {

return "ADMINISTRADOR";

}

else {

return "CAJERO";

}

}

}

DlgReporteHospitalizacionesPorFecha

package gui;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloCamas;

import arreglos.ArregloEmpleado;

import arreglos.ArregloHospitalizacion;

import arreglos.ArregloPaciente;

import clases.Empleado;

import clases.Hospitalizacion;

import clases.Paciente;

import libreria.Fecha;

import libreria.Libreria;

public class DlgReporteHospitalizacionesPorFecha extends JDialog implements ActionListener {

private static final long serialVersionUID = 1L;

private JScrollPane scrollPane;

private JTable jtblHospitalizaciones;

private JLabel label;

private JComboBox<String> cboDia;

private JComboBox<String> cboMes;

private JComboBox<String> cboAño;

private JButton btnBuscar;

private DefaultTableModel dtm;

// DECLARACIÓN GLOBAL DE ARREGLOS

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

ArregloCamas ac = new ArregloCamas("camas.txt");

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

ArregloEmpleado ae = new ArregloEmpleado("empleados.txt");

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Exception e) {

e.printStackTrace();

}

try {

DlgReporteHospitalizacionesPorFecha dialog = new DlgReporteHospitalizacionesPorFecha();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgReporteHospitalizacionesPorFecha() {

setResizable(false);

setModal(true);

getContentPane().setBackground(Color.WHITE);

setTitle("BUSCAR HOSPITALIZACIONES");

setBounds(100, 100, 1069, 490);

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 62, 1044, 388);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblHospitalizaciones = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblHospitalizaciones.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblHospitalizaciones.setFont(new Font("Arial", Font.BOLD, 14));

scrollPane.setViewportView(jtblHospitalizaciones);

label = new JLabel("FECHA LLEGADA");

label.setHorizontalAlignment(SwingConstants.LEFT);

label.setForeground(Color.BLACK);

label.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

label.setBounds(10, 18, 103, 14);

getContentPane().add(label);

cboDia = new JComboBox<String>();

cboDia.setForeground(Color.BLACK);

cboDia.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboDia.setBounds(123, 6, 59, 38);

Fecha.colocarItems(cboDia, 1, 31);

Fecha.colocarDiaActual(cboDia);

getContentPane().add(cboDia);

cboMes = new JComboBox<String>();

cboMes.setForeground(Color.BLACK);

cboMes.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboMes.setBounds(183, 6, 116, 38);

Fecha.colocarMeses(cboMes);

Fecha.colocarMesActual(cboMes);

getContentPane().add(cboMes);

cboAño = new JComboBox<String>();

cboAño.setForeground(Color.BLACK);

cboAño.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboAño.setBounds(300, 6, 83, 38);

Fecha.colocarItems(cboAño, 2000, Fecha.añoActual());

cboAño.setSelectedIndex(16);

getContentPane().add(cboAño);

btnBuscar = new JButton("BUSCAR");

btnBuscar.addActionListener(this);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnBuscar.setBounds(393, 6, 149, 38);

getContentPane().add(btnBuscar);

modeloTabla();

listar(0);

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

listar(Fecha.getFecha(cboDia, cboMes, cboAño));

}

// MÉTODOS VOID CON PARÁMETROS

void listar(int fecha) {

Libreria.limpiarJTable(jtblHospitalizaciones, dtm);

Hospitalizacion h;

for (int i = 0; i < ah.tamaño(); i++) {

h = ah.obtener(i);

Paciente p;

for (int j = 0; j < ap.tamaño(); j++) {

p = ap.obtener(j);

if (h.getCodPaci() == p.getCodpaciente()) {

Empleado e;

for (int j2 = 0; j2 < ae.tamaño(); j2++) {

e = ae.obtener(j2);

if (e.getCodEmpleado() == h.getCodEmpl()) {

if (fecha != 0) {

if (h.getFecLleg() == fecha ) {

dtm.addRow(new Object[] { h.getCodHosp(), p.getNombres().toUpperCase(),

p.getApellidos().toUpperCase(), e.getNombres().toUpperCase(),

e.getApellidos().toUpperCase(), h.getNumCama(),

Fecha.dd\_mm\_aaaa(h.getFecLleg()), Fecha.hh\_mm\_ss(h.getHorLleg())});

}

}else {

dtm.addRow(new Object[] { h.getCodHosp(), p.getNombres().toUpperCase(),

p.getApellidos().toUpperCase(), e.getNombres().toUpperCase(),

e.getApellidos().toUpperCase(), h.getNumCama(),

Fecha.dd\_mm\_aaaa(h.getFecLleg()), Fecha.hh\_mm\_ss(h.getHorLleg())});

}

}

}

}

}

}

}

// MÉTODOS VOID SIN PARÁMETROS

void modeloTabla() {

TableColumnModel tc = jtblHospitalizaciones.getColumnModel();

tc.getColumn(0).setPreferredWidth(60);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(150);

tc.getColumn(4).setPreferredWidth(150);

tc.getColumn(5).setPreferredWidth(60);

tc.getColumn(6).setPreferredWidth(100);

tc.getColumn(7).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(5).setCellRenderer(tcr);

tc.getColumn(6).setCellRenderer(tcr);

tc.getColumn(7).setCellRenderer(tcr);

}

// MÉTODOS CON RETORNO SIN PARÁMETROS

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "NOMBRES PACIENTE", "APELLIDOS PACIENTE", "NOMBRE EMPLEADO",

"APELLIDO EMPLEADO", "NRO CAMA", "FECHA LLEGADA", "HORA LLEGADA" };

return columnas;

}

}

DlgReportePacientesAdmitidos

package gui;

import javax.swing.JDialog;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloCamas;

import arreglos.ArregloHospitalizacion;

import arreglos.ArregloPaciente;

import clases.Camas;

import clases.Hospitalizacion;

import clases.Paciente;

import libreria.Libreria;

import java.awt.Color;

import java.awt.Font;

public class DlgReportePacientesAdmitidos extends JDialog {

private static final long serialVersionUID = 1L;

private JScrollPane scrollPane;

private JTable jtblPacientes;

private DefaultTableModel dtm;

// DECLARACIÓN GLOBAL DE ARREGLOS HOSPITALIZACIÓN, CAMA Y PACIENTES

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

ArregloCamas ac = new ArregloCamas("camas.txt");

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Exception e) {

e.printStackTrace();

}

try {

DlgReportePacientesAdmitidos dialog = new DlgReportePacientesAdmitidos();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgReportePacientesAdmitidos() {

getContentPane().setBackground(Color.WHITE);

setResizable(false);

setModal(true);

setBounds(100, 100, 826, 553);

setTitle("PACIENTES ADMITIDOS");

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 11, 799, 503);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblPacientes = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblPacientes.setFont(new Font("Arial", Font.BOLD, 14));

jtblPacientes.setForeground(Color.BLACK);

jtblPacientes.setRowHeight(25);

jtblPacientes.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

scrollPane.setViewportView(jtblPacientes);

formatoTabla();

listadoPacientes();

setLocationRelativeTo(this);

}

// MÉTODOS VOID SIN PARÁMETROS

void formatoTabla() {

TableColumnModel tc = jtblPacientes.getColumnModel();

tc.getColumn(0).setPreferredWidth(50);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(50);

tc.getColumn(4).setPreferredWidth(100);

tc.getColumn(5).setPreferredWidth(100);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tc.getColumn(3).setCellRenderer(tcr);

tc.getColumn(4).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(5).setCellRenderer(tcr);

}

void listadoPacientes() {

Hospitalizacion h;

for (int i = 0; i < ah.tamaño(); i++) {

h = ah.obtener(i);

if (h.getEstado() == 1) {

Paciente p;

for (int j = 0; j < ap.tamaño(); j++) {

p = ap.obtener(j);

if (p.getCodpaciente() == h.getCodPaci()) {

Camas c;

for (int j2 = 0; j2 < ac.tamaño(); j2++) {

c = ac.obtener(j2);

if (c.getNroCama() == h.getNumCama()) {

dtm.addRow(new Object[] { p.getCodpaciente(), p.getNombres().toUpperCase(),

p.getApellidos().toUpperCase(), h.getNumCama(), getCategoria(c.getCategoria()),

Libreria.formato2Decimales(c.getPrecioxDia()) });

}

}

}

}

}

}

}

// MÉTODOS CON RETORNO SIN PARÁMETROS

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO", "NOMBRES", "APELLIDOS", "NRO CAMA", "CATEGORÍA",

"PRECIO POR DÍA" };

return columnas;

}

// MÉTODOS CON RETORNO CON PARÁMETROS

String getCategoria(int est) {

if (est == 0) {

return "ECONÓMICO";

}

else {

return "EJECUTIVO";

}

}

}

DlgReportePacientesPagado

package gui;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.ListSelectionModel;

import javax.swing.SwingConstants;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumnModel;

import arreglos.ArregloConsumo;

import arreglos.ArregloHospitalizacion;

import arreglos.ArregloPaciente;

import clases.Consumo;

import clases.Hospitalizacion;

import clases.Paciente;

import libreria.Fecha;

import libreria.Libreria;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class DlgReportePacientesPagado extends JDialog implements ActionListener {

private static final long serialVersionUID = 1L;

private JScrollPane scrollPane;

private JTable jtblPagoDePacientes;

private JLabel lblFechaLlegada;

private JComboBox<String> cboDia;

private JComboBox<String> cboMes;

private JComboBox<String> cboAño;

private JButton btnBuscar;

private DefaultTableModel dtm;

// DECLARACIÓN GLOBAL DE ARREGLOS

ArregloPaciente ap = new ArregloPaciente("pacientes.txt");

ArregloConsumo ac = new ArregloConsumo("consumos.txt");

ArregloHospitalizacion ah = new ArregloHospitalizacion("hospitalizaciones.txt");

public static void main(String[] args) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

}

catch (Exception e) {

e.printStackTrace();

}

try {

DlgReportePacientesPagado dialog = new DlgReportePacientesPagado();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

}

catch (Exception e) {

e.printStackTrace();

}

}

public DlgReportePacientesPagado() {

setResizable(false);

setModal(true);

getContentPane().setBackground(Color.WHITE);

setTitle("BÚSQUEDA DE PAGO DE PACIENTES");

setBounds(100, 100, 688, 423);

getContentPane().setLayout(null);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 62, 662, 324);

getContentPane().add(scrollPane);

dtm = new DefaultTableModel(null, getColumnas());

jtblPagoDePacientes = new JTable(dtm) {

private static final long serialVersionUID = 1L;

public boolean isCellEditable(int rowIndex, int vColIndex) {

return false;

}

};

jtblPagoDePacientes.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

jtblPagoDePacientes.setFont(new Font("Arial", Font.BOLD, 14));

scrollPane.setViewportView(jtblPagoDePacientes);

lblFechaLlegada = new JLabel("FECHA LLEGADA");

lblFechaLlegada.setHorizontalAlignment(SwingConstants.LEFT);

lblFechaLlegada.setForeground(Color.BLACK);

lblFechaLlegada.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

lblFechaLlegada.setBounds(10, 18, 103, 14);

getContentPane().add(lblFechaLlegada);

cboDia = new JComboBox<String>();

cboDia.setForeground(Color.BLACK);

cboDia.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboDia.setBounds(123, 6, 59, 38);

Fecha.colocarItems(cboDia, 1, 31);

Fecha.colocarDiaActual(cboDia);

getContentPane().add(cboDia);

cboMes = new JComboBox<String>();

cboMes.setForeground(Color.BLACK);

cboMes.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboMes.setBounds(183, 6, 116, 38);

Fecha.colocarMeses(cboMes);

Fecha.colocarMesActual(cboMes);

getContentPane().add(cboMes);

cboAño = new JComboBox<String>();

cboAño.setForeground(Color.BLACK);

cboAño.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

cboAño.setBounds(300, 6, 83, 38);

Fecha.colocarItems(cboAño, 2000, Fecha.añoActual());

cboAño.setSelectedIndex(16);

getContentPane().add(cboAño);

btnBuscar = new JButton("BUSCAR");

btnBuscar.addActionListener(this);

btnBuscar.setForeground(Color.BLACK);

btnBuscar.setFont(new Font("Source Sans Pro Semibold", Font.PLAIN, 14));

btnBuscar.setBounds(393, 6, 149, 38);

getContentPane().add(btnBuscar);

modeloTabla();

listar(0);

setLocationRelativeTo(this);

}

public void actionPerformed(ActionEvent arg0) {

if (arg0.getSource() == btnBuscar) {

actionPerformedBtnBuscar(arg0);

}

}

protected void actionPerformedBtnBuscar(ActionEvent arg0) {

listar(Fecha.getFecha(cboDia, cboMes, cboAño));

}

// MÉTODOS VOID CON PARÁMETROS

void listar(int fecha) {

Libreria.limpiarJTable(jtblPagoDePacientes, dtm);

Hospitalizacion h;

for (int i = 0; i < ah.tamaño(); i++) {

h = ah.obtener(i);

if (h.getEstado() == 2) {

Paciente p;

for (int j = 0; j < ap.tamaño(); j++) {

p = ap.obtener(j);

if (h.getCodPaci() == p.getCodpaciente()) {

Consumo c;

for (int j2 = 0; j2 < ac.tamaño(); j2++) {

c = ac.obtener(j2);

if (c.getEstado() == 1) {

if (c.getCodPaciente() == p.getCodpaciente()) {

if (fecha != 0) {

if (h.getFecSali() == fecha) {

dtm.addRow(new Object[] { c.getCodConsumo(), p.getApellidos().toUpperCase(),

p.getNombres().toUpperCase(),

Libreria.formato2Decimales(c.getTotalPago()) });

}

}

else {

dtm.addRow(new Object[] { c.getCodConsumo(), p.getApellidos().toUpperCase(),

p.getNombres().toUpperCase(),

Libreria.formato2Decimales(c.getTotalPago()) });

}

}

}

}

}

}

}

}

}

// MÉTODOS VOID SIN PARÁMETROS

void modeloTabla() {

TableColumnModel tc = jtblPagoDePacientes.getColumnModel();

tc.getColumn(0).setPreferredWidth(70);

tc.getColumn(1).setPreferredWidth(150);

tc.getColumn(2).setPreferredWidth(150);

tc.getColumn(3).setPreferredWidth(50);

DefaultTableCellRenderer tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.CENTER);

tc.getColumn(0).setCellRenderer(tcr);

tcr = new DefaultTableCellRenderer();

tcr.setHorizontalAlignment(SwingConstants.RIGHT);

tc.getColumn(3).setCellRenderer(tcr);

}

// MÉTODOS CON RETORNO SIN PARÁMETROS

String[] getColumnas() {

String columnas[] = new String[] { "CÓDIGO CONSUMO", "APELLIDOS PACIENTE", "NOMBRES PACIENTE",

"TOTAL A PAGAR" };

return columnas;

}

}

Librerías

Diseño Objetos

package libreria;

import javax.swing.ImageIcon;

import javax.swing.JButton;

public class DiseñoObjetos {

// Métodos void con parámetros

public void setCurvasButton(JButton btn, String ruta) {

btn.setIcon(new ImageIcon(ruta));

}

}

Validaciones

package libreria;

import java.awt.event.KeyEvent;

import javax.swing.JTextField;

public class Validaciones {

//Métodos static void con parámetros

public static void soloNumero(KeyEvent e, JTextField txt, int longitud) {

if (txt.getText().length() >= longitud) {

e.consume();

}

char car = e.getKeyChar();

if (!Character.isDigit(car)) {

e.consume();

}

}

public static void numeroReal(JTextField txt, KeyEvent e, int longitud) {

if (txt.getText().length() >= longitud) {

e.consume();

}

char car = e.getKeyChar();

String numero = txt.getText().trim();

if (!Character.isDigit(car) && car != '.') {

e.consume();

}else {

if (numero.indexOf(".") != -1 && !Character.isDigit(car)) {

e.consume();

}

}

}

public static void soloLetras(KeyEvent e, JTextField txt, int longitud) {

if (txt.getText().length() >= longitud) {

e.consume();

}

//OBTIENES EL VALOR DEL CARACTER

char car = e.getKeyChar();

if (!Character.isLetter(car) && car != KeyEvent.VK\_SPACE) {

e.consume();

}

}

public static void letrasyNumeros(KeyEvent e, JTextField txt, int longitud) {

if (txt.getText().length() >= longitud) {

e.consume();

}

char car = e.getKeyChar();

if (!Character.isLetter(car) && !Character.isDigit(car)) {

e.consume();

}

}

public static void cadenaTexto(KeyEvent e, JTextField txt, int longitud) {

if (txt.getText().length() >= longitud) {

e.consume();

}

char car = e.getKeyChar();

if (!Character.isLetter(car) && car != KeyEvent.VK\_SPACE && !Character.isDigit(car)) {

e.consume();

}

}

}

Librería

package libreria;

import java.text.DecimalFormat;

import java.text.NumberFormat;

import java.util.Locale;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

public class Libreria {

// Métodos static void que no retornan valor con parámetros

public static void limpiarJTable(JTable jtbl, DefaultTableModel dtm) {

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

dtm.removeRow(i);

i--;

}

}

public static void mensajeAdvertenciaFrame(JFrame jf, String s) {

JOptionPane.showMessageDialog(jf, s, "Advertencia", JOptionPane.WARNING\_MESSAGE);

}

public static void mensajeError(JDialog jd, String s) {

JOptionPane.showMessageDialog(jd, s, "Error", JOptionPane.ERROR\_MESSAGE);

}

public static void mensajeAdvertencia(JDialog jd, String s) {

JOptionPane.showMessageDialog(jd, s, "Advertencia", JOptionPane.WARNING\_MESSAGE);

}

public static void mensajeInformacion(JDialog jd, String s) {

JOptionPane.showMessageDialog(jd, s, "Información", JOptionPane.INFORMATION\_MESSAGE);

}

// Métodos static que retornan valor con parámetros

public static int confirmacion(JDialog jd, String s) {

return JOptionPane.showConfirmDialog(jd, s, "Advertencia", JOptionPane.YES\_NO\_OPTION,

JOptionPane.WARNING\_MESSAGE);

}

public static String formatoSoles(double numero) {

Locale lp = new Locale("es", "PE");

NumberFormat nf = NumberFormat.getCurrencyInstance(lp);

return nf.format(numero);

}

public static String formatoDecimales(double numero) {

DecimalFormat df = new DecimalFormat("##.##");

return df.format(numero);

}

public static String formato2Decimales(double numero) {

return String.format("%.2f", numero);

}

public static String getPassword(JPasswordField txt) {

return new String(txt.getPassword());

}

public static int leerEntero(JTextField txt) {

try {

return Integer.parseInt(txt.getText().trim());

}

catch (Exception e) {

return -1;

}

}

public static double leerReal(JTextField txt) {

try {

return Double.parseDouble(txt.getText().trim());

}

catch (Exception e) {

return -1.0;

}

}

public static String leerCadena(JTextField txt) {

return txt.getText().trim();

}

}

Fecha

package libreria;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.swing.JComboBox;

public class Fecha {

// Métodos static tipo void (con parámetros)

public static void colocarMeses(JComboBox<String> cbo) {

cbo.addItem("Enero");

cbo.addItem("Febrero");

cbo.addItem("Marzo");

cbo.addItem("Abril");

cbo.addItem("Mayo");

cbo.addItem("Junio");

cbo.addItem("Julio");

cbo.addItem("Agosto");

cbo.addItem("Setiembre");

cbo.addItem("Octubre");

cbo.addItem("Noviembre");

cbo.addItem("Diciembre");

}

public static void colocarDiaActual(JComboBox<String> cboDia) {

Calendar calendario = new GregorianCalendar();

cboDia.setSelectedIndex(calendario.get(Calendar.DAY\_OF\_MONTH) - 1);

}

public static void colocarMesActual(JComboBox<String> cbo) {

Calendar calendario = new GregorianCalendar();

cbo.setSelectedIndex(calendario.get(Calendar.MONTH));

}

public static void colocarItems(JComboBox<String> cbo, int inicio, int fin) {

if (inicio < fin)

while (inicio <= fin) {

cbo.addItem("" + inicio);

inicio++;

}

else

while (inicio >= fin) {

cbo.addItem("" + inicio);

inicio--;

}

}

public static int obtenerHoraActual() {

String minuto = "";

String segundo = "";

Calendar calendario = new GregorianCalendar();

int h = calendario.get(Calendar.HOUR\_OF\_DAY);

int m = calendario.get(Calendar.MINUTE);

if (m < 10)

minuto = "0" + m;

else

minuto = "" + m;

int s = calendario.get(Calendar.SECOND);

if (s < 10)

segundo = "0" + s;

else

segundo = "" + s;

return Integer.parseInt(h + minuto + segundo);

}

public static void setFecha(JComboBox<String> dia, JComboBox<String> mes, JComboBox<String> año, int fecha) {

int dd = fecha % 100, mm = (fecha / 100) % 100, aa = fecha / 10000;

dia.setSelectedIndex(dd - 1);

mes.setSelectedIndex(mm - 1);

año.setSelectedItem(aa + "");

}

// Métodos static que retornan valor (sin parámetros)

public static int añoActual() {

Calendar calendario = new GregorianCalendar();

return calendario.get(Calendar.YEAR);

}

// Métodos static que retornan valor (con parámetros)

public static String dd\_mm\_aaaa(int fecha) {

String s = "" + fecha;

return "" + s.charAt(6) + s.charAt(7) + '/' + s.charAt(4) + s.charAt(5) + '/' + s.charAt(0) + s.charAt(1)

+ s.charAt(2) + s.charAt(3);

}

public static int obtenerDiaDeFecha(int fecha) {

String s = "" + fecha;

return s.charAt(6) + s.charAt(7);

}

public static int obtenerMesDeFecha(int fecha) {

String s = "" + fecha;

return s.charAt(4) + s.charAt(5);

}

public static int obtenerAñoDeFecha(int fecha) {

String s = "" + fecha;

return s.charAt(0) + s.charAt(1) + s.charAt(2) + s.charAt(3);

}

public static String hh\_mm\_ss(int hora) {

String s = "" + hora;

if (s.length() == 4) {

return "00" + ':' + s.charAt(0) + s.charAt(1) + ':' + s.charAt(2) + s.charAt(3);

}

else if (s.length() == 5) {

return "0" + s.charAt(0) + ':' + s.charAt(1) + s.charAt(2) + ':' + s.charAt(3) + s.charAt(4);

}

else {

return "" + s.charAt(0) + s.charAt(1) + ':' + s.charAt(2) + s.charAt(3) + ':' + s.charAt(4) + s.charAt(5);

}

}

public static int getFecha(JComboBox<String> dia, JComboBox<String> mes, JComboBox<String> año) {

int dd = dia.getSelectedIndex() + 1, mm = mes.getSelectedIndex() + 1,

aa = Integer.parseInt(año.getSelectedItem().toString());

return (aa \* 100 + mm) \* 100 + dd;

}

public static int CantidadDiasEntre2fechas(int Y1, int M1, int D1, int Y2, int M2, int D2) {

GregorianCalendar date = new GregorianCalendar(Y1, M1, D1);

GregorianCalendar date2 = new GregorianCalendar(Y2, M2, D2);

long difms = date2.getTimeInMillis() - date.getTimeInMillis();

long difd = difms / (1000 \* 60 \* 60 \* 24);

return (int) difd;

}

}

FocusTraversalOnArray

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright (c) 2011 Google, Inc.

\* All rights reserved. This program and the accompanying materials

\* are made available under the terms of the Eclipse Public License v1.0

\* which accompanies this distribution, and is available at

\* http://www.eclipse.org/legal/epl-v10.html

\*

\* Contributors:

\* Google, Inc. - initial API and implementation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package org.eclipse.wb.swing;

import java.awt.Component;

import java.awt.Container;

import java.awt.FocusTraversalPolicy;

/\*\*

\* Cyclic focus traversal policy based on array of components.

\* <p>

\* This class may be freely distributed as part of any application or plugin.

\*

\* @author scheglov\_ke

\*/

public class FocusTraversalOnArray extends FocusTraversalPolicy {

private final Component m\_Components[];

////////////////////////////////////////////////////////////////////////////

//

// Constructor

//

////////////////////////////////////////////////////////////////////////////

public FocusTraversalOnArray(Component components[]) {

m\_Components = components;

}

////////////////////////////////////////////////////////////////////////////

//

// Utilities

//

////////////////////////////////////////////////////////////////////////////

private int indexCycle(int index, int delta) {

int size = m\_Components.length;

int next = (index + delta + size) % size;

return next;

}

private Component cycle(Component currentComponent, int delta) {

int index = -1;

loop : for (int i = 0; i < m\_Components.length; i++) {

Component component = m\_Components[i];

for (Component c = currentComponent; c != null; c = c.getParent()) {

if (component == c) {

index = i;

break loop;

}

}

}

// try to find enabled component in "delta" direction

int initialIndex = index;

while (true) {

int newIndex = indexCycle(index, delta);

if (newIndex == initialIndex) {

break;

}

index = newIndex;

//

Component component = m\_Components[newIndex];

if (component.isEnabled() && component.isVisible() && component.isFocusable()) {

return component;

}

}

// not found

return currentComponent;

}

////////////////////////////////////////////////////////////////////////////

//

// FocusTraversalPolicy

//

////////////////////////////////////////////////////////////////////////////

public Component getComponentAfter(Container container, Component component) {

return cycle(component, 1);

}

public Component getComponentBefore(Container container, Component component) {

return cycle(component, -1);

}

public Component getFirstComponent(Container container) {

return m\_Components[0];

}

public Component getLastComponent(Container container) {

return m\_Components[m\_Components.length - 1];

}

public Component getDefaultComponent(Container container) {

return getFirstComponent(container);

}

}