Manual Técnico

Se creó un programa que permitía al usuario ingresar productos con precio para su envío a diferentes sitios, utilizando los hilos para poder visualizar en tiempo real el trayecto de cada vehículo utilizado y poder calcular los tiempos de envíos de estos.

Además de poder registrarlos para el historial de pedidos almacenados en una tabla en la ventana de historial que permitía conocer este dato.

Diccionario de Clases

El form del menú permite en cada botón de la cabecera abrir un JFrame distinto, enviándole el arraylist necesario a cada uno si se requiere

```
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Registrar registrar = new Registrar(lista);
        registrar.setVisible(b: true);
        registrar.setTitle(title: "Registros");
        registrar.setLocationRelativeTo(c: null);
    }

    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Historial historial = new Historial(pedidos);
        historial.setVisible(b: true);
        historial.setTitle(title: "Historial de Pedidos");
        historial.setLocationRelativeTo(c: null);
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        FrmPedido pedido = new FrmPedido(lista, pedidos);
        pedido.setVisible(b: true);
        pedido.setTitle(title: "Pedidos");
        pedido.setLocationRelativeTo(c: null);
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        VisualizarPedido ver = new VisualizarPedido(pedidos);
        ver.setVisible(b: true);
        ver.setTitle(title: "Historial de Pedidos");
        ver.setLocationRelativeTo(c: null);
    }
```

El JFrame de registro captura los datos de los JTextField y los pasa a las variables del objeto Producto y luego guarda el objeto en un arraylist llamado lista.

```
private void btnRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    Producto producto = new Producto();

    try{
        producto.setNombre(nombre: txtNombre.getText());
        producto.setPrecio(precio: Double.parseDouble(s: txtPrecio.getText()));
        lista.addProducto(miProducto: producto);
        JOptionPane.showMessageDialog(parentComponent:null, "Se registro."+lista
        limpiarTextos();
    } catch (Exception ex){
        JOptionPane.showMessageDialog(parentComponent:null, message:"Error");
        limpiarTextos();
    }
}
```

La clase de Producto mantiene los atributos del registro de productos y contiene sus setters y getters para almacenarlos en un objeto.

```
public class Producto implements Serializable{
    private static RegistroProductos lista;
    private String nombre;
    private double precio;

    public Producto() {
        this.nombre = "";
        this.precio = 0.0;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public void mostrarDatos(){
        System.out.println("Nombre: "+nombre);
        System.out.println("Precio: "+precio);
    }

}
```

La clase de RegistroProductos es la clase donde se creó el ArrayList donde se almacena los objetos de la clase Producto. Cuenta con sus funciones donde puede conocer el index, añadir objetos y conocer el tamaño.

```
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class RegistroProductos {
    private ArrayList<Producto> lista;

    public RegistroProductos(){
        lista = new ArrayList <Producto>();
    }

    public void addProducto(Producto miProducto){
        lista.add(e: miProducto);
    }

    public int getRegistros(){
        return lista.size();
    }

    public Producto getRegistroProducto(int posicion){
        try{
            Producto producto = lista.get(index: posicion);
            return producto;
        } catch (Exception ex){
            JOptionPane.showMessageDialog(parentComponent:null, message:"Posicion inv
            return new Producto();
        }
    }

}
```

La clase de FrmPedido permite visualizar en una tabla todos los productos almacenados en el arraylist. También permite tomar cada fila del producto, colocarle un vehículo y una distancia, y agregarlos a la tabla de pedidos donde se almacenan los productos que serán enviados a sus clientes, también cuenta con un JLabel donde puede conocerse el monto total a pagar por los pedidos. Para esto se crea un modelo de tabla con DefaultTableModel para ingresar los datos de la primera tabla y conforme se seleccionan por medio del JButton se agregan a la segunda tabla con el modelo 2.

Modelo primera tabla.

```
//Método para llenar la tabla
public void MostrarDatos(Producto producto){
    modelo.insertRow(row:con, new Object[]{});
    modelo.setValueAt(aValue: producto.getNombre(), row:con, column: 0);
    modelo.setValueAt(aValue: producto.getPrecio(), row:con, column: 1);
    con++;
}

//Método para crer la tabla
public void CargarInterfaz1(){
    String data[][] = {};
    String col[] = {"Nombre", "Precio"};
    modelo = new DefaultTableModel(data, columnNames:col);
    tblProducto.setModel(dataModel: modelo);
}
```

Constructor donde se crea la segunda tabla

```
//Constructor de JFrame
public FrmPedido(RegistroProductos lista, Pedidos pedidos) {
    initComponents();
    this.pedidos = pedidos;
    modelo2 = new DefaultTableModel();
    CargarInterfaz1();
    Producto producto;
    for (int i = 0; i < lista.getRegistros(); i++) {
        producto = (Producto)lista.getRegistroProducto(posicion: i);
        MostrarDatos(producto);
    }
    modelo2.addColumn(columnName: "Vehículo");
    modelo2.addColumn(columnName: "Nombre");
    modelo2.addColumn(columnName: "Precio");
    modelo2.addColumn(columnName: "Distancia");
    this.tblPedido.setModel(dataModel: modelo2);
}
```

Botón que agrega los pedidos a un nuevo ArrayList para juntarlo a los tiempos de entrega

```
private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Pedido pedido = new Pedido();
    File archivo = new File(pathname: "pedido.dat");
    String []info = new String[4];
    try{
        nombre[index] = String.valueOf(obj:tblProducto.getValueAt(row:seleccionFila, column: 0));
        precio[index] = Double.parseDouble(s: String.valueOf(obj:tblProducto.getValueAt(row:seleccionFila, column: 1)));
        distancia[index] = Integer.parseInt(s: txtDistancia.getText());
        vehiculo[index] = cboVehiculo.getSelectedItem().toString();
        pedido.setNombre(nombre[index]);
        pedido.setPrecio(precio[index]);
        pedido.setDistancia(distancia[index]);
        pedido.setFechaInicio(fechaInicio:String.valueOf(obj:LocalDateTime.now()));
        try {
            FileOutputStream fos = new FileOutputStream(file: archivo);
            ObjectOutputStream oos = new ObjectOutputStream(out:fos);
            oos.writeObject(obj:pedido);
            oos.close();
            fos.close();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(parentComponent:null, message:e);
        }
        pedidos.addPedido(miPedido: pedido);

        //int index = cboVehiculo.getSelectedIndex();
        info[0] = vehiculo[index];
        info[1] = nombre[index];
        info[2] = String.valueOf(precio[index]);
        info[3] = String.valueOf(distancia[index]);
        modelo2.addRow(rowData:info);
        txtDistancia.setText(t: "");
        index++;

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Error");
    }
}
```

ArrayList donde se almacenan los pedidos y sus tiempos de entrega y salida.

```
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class Pedidos {
    private ArrayList<Pedido> pedidos;

    public Pedidos(){
        pedidos = new ArrayList <Pedido>();
    }

    public void addPedido(Pedido miPedido){
        pedidos.add(e: miPedido);
    }

    public int getSize(){
        return pedidos.size();
    }

    public Pedido getRegistroPedido(int posicion){
        try{
            Pedido pedido = pedidos.get(index: posicion);
            return pedido;
        } catch (Exception ex){
            JOptionPane.showMessageDialog(parentComponent:null, message:"Posicion invalida");
            return new Pedido();
        }
    }

}
```

Clase donde se almacenan los pedidos como objeto.

```
public class Pedido implements Serializable{
    private static Pedidos pedidos;
    private int index;
    private String nombre;
    private double precio;
    private int distancia;
    private String fechaInicio;
    private String fechaFinal;

    public Pedido(){
        this.index = 0;
        this.nombre = "";
        this.precio = 0.0;
        this.distancia = 0;
        this.fechaInicio = "";
        this.fechaFinal = "";
    }
```

Se creó una clase hilo para permitir conocer la animación de los vehículos y sus tiempos de entrega.

```
@Override
public void run(){
    int moto1;
    int moto2;
    int moto3;

    while(true){
        try{

            sleep((int)(Math.random() * 1000));
            moto1 = moto.getPrimerMoto().getLocation().x;
            moto2 = moto.getSegundoMoto().getLocation().x;
            moto3 = moto.getTercerMoto().getLocation().x;

            if (moto1 < moto.getMeta().getLocation().x - 125 && moto2 < moto.getMeta().getLocation().x - 50 && moto3 < moto.getMeta().getLocation().x - 50) {
                etiqueta.setLocation(etiqueta.getLocation().x + 10, y: etiqueta.getLocation().y);
                moto.repaint();
            } else {
                for (int i = 0; i < moto.fechasFinal.length; i++) {
                    moto.fechasFinal[i] = String.valueOf(obj: LocalDateTime.now());

                }
                break;
            }

        } catch (InterruptedException e) {
            JOptionPane.showMessageDialog(parentComponent: null, message: e);
        }

    }
}
```

Se implementó la clase Hilo en el JFrame de VisualizarPedido para utilizarlo en la animación de las motos de entrega.

Código del botón enviar todos. El resto de botones solo contienen menos inicios del hilo.

```
private void btnEnviarTodosActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    moto1.setLocation(x: 0,y: moto1.getLocation().y);
    moto2.setLocation(x: 0,y: moto2.getLocation().y);
    moto3.setLocation(x: 0,y: moto3.getLocation().y);
    Hilo vehiculo1 = new Hilo(etiqueta: moto1, moto: this);
    Hilo vehiculo2 = new Hilo(etiqueta: moto2, moto: this);
    Hilo vehiculo3 = new Hilo(etiqueta: moto3, moto: this);
    vehiculo1.start();
    vehiculo2.start();
    vehiculo3.start();

}
```

Tabla de la clase Historial que permite visualizar todo el historial de pedidos realizados a lo largo del uso del programa.

```
public Historial(Pedidos pedidos) {
    initComponents();
    this.pedidos = pedidos;
    CargarInterfaz();
    Pedido pedido;
    for (int i = 0; i < pedidos.getSize(); i++) {
        pedido = (Pedido)pedidos.getRegistroPedido(posicion: i);
        MostrarDatos(pedido);
    }
}

public void MostrarDatos(Pedido pedido){
    modelo.insertRow(row:con, new Object[]{});
    modelo.setValueAt(aValue: pedido.getNombre(), row:con, column: 0);
    modelo.setValueAt(aValue: pedido.getPrecio(), row:con, column: 1);
    modelo.setValueAt(aValue: pedido.getDistancia(), row:con, column: 2);
    modelo.setValueAt(aValue: pedido.getFechaInicio(), row:con, column: 3);
    con++;
}

//Método para crer la tabla
public void CargarInterfaz(){
    String data[][] = {};
    String col[] = {"Vehiculo", "Distancia", "Monto", "Hora de salida"};
    modelo = new DefaultTableModel(data, columnNames:col);
    tblHistorial.setModel(dataModel: modelo);
}
```