

Universidad Nacional de Colombia

Tarea6 - Parte1

Autores:

Juan José Jiménez Maya

Programa: Programación Orientada a Objetos


Grupo: 3

Imagen: Interfaz.png

Gestor de Contactos

Nombre	Teléfono
Juan	477854774
Lucas	325998751
Camilo	488754215
Sara	423655587

Añadir Contacto



Nombre:

Teléfono:

Aceptar

Cancelar

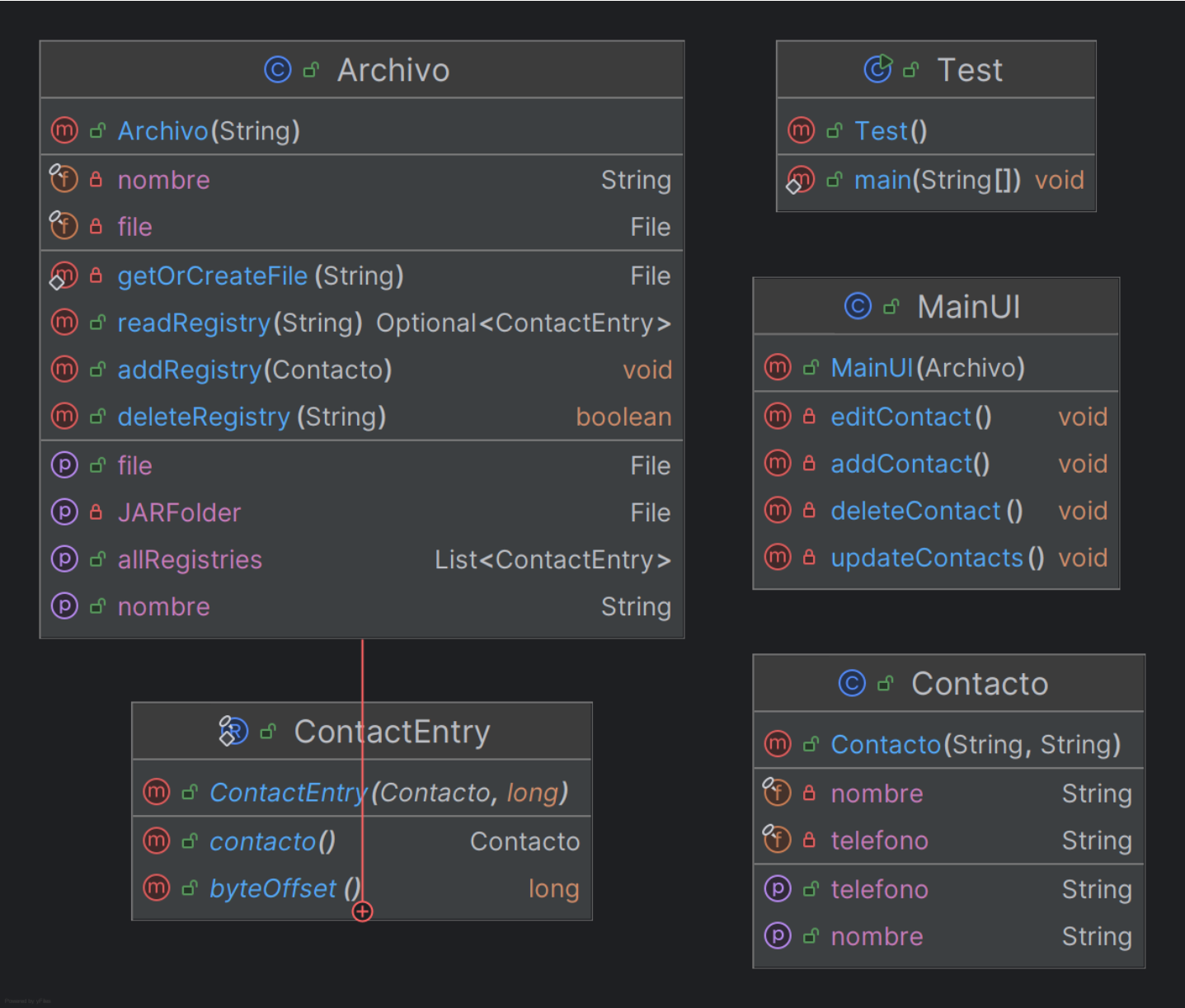
Actualizar

Añadir

Eliminar

Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/Interfaz.png>

Imagen: DiagramaUML.png



Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/DiagramaUML.png>

Código: Test.java

```
package Tareas.Tarea6.Partel1;

public class Test {

    public static void main(String[] args) {
        try {
            Archivo archivo = new Archivo("contactos.txt");
            MainUI mainUI = new MainUI(archivo);
            mainUI.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/Test.java>

Código: Archivo.java

```
package Tareas.Tarea6.Partel1;

import java.io.*;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class Archivo {

    private final String nombre;
    private final File file;

    private static final String SEPARADOR = "!";

    public Archivo(String nombre) throws IOException, URISyntaxException {
        this.nombre = nombre;
        this.file = getOrCreateFile(nombre);
    }

    private static File getOrCreateFile(String nombre) throws IOException, URISyntaxException {
        File file = new File(getJARFolder().getPath() + File.separator + nombre);
        if (!file.exists()) {
            if (!file.createNewFile()) {
                throw new IOException("Failed to create file: " + file.getAbsolutePath());
            }
        }
        return file;
    }

    private static File getJARFolder() throws URISyntaxException {
        return new
File(Archivo.class.getProtectionDomain().getCodeSource().getLocation().toURI()).getParentFile();
    }

    public String getNombre() {
        return nombre;
    }

    public File getFile() {
        return file;
    }

    public synchronized void addRegistry(Contacto contacto) throws IOException {
        Optional<ContactEntry> entry = readRegistry(contacto.nombre());

        try (RandomAccessFile accessFile = new RandomAccessFile(file, "rw")) {
            if (entry.isPresent()) {
                registro
                accessFile.seek(entry.get().byteOffset()); // Se posiciona en el byte en el que inicia el
                accessFile.writeBytes(contacto.nombre() + SEPARADOR + contacto.telefono() +
System.lineSeparator());
            } else {
                accessFile.seek(accessFile.length());
                accessFile.writeBytes(contacto.nombre() + SEPARADOR + contacto.telefono() +
System.lineSeparator());
            }
        }
    }

    public synchronized Optional<ContactEntry> readRegistry(String nombre) throws IOException {
        try (RandomAccessFile accessFile = new RandomAccessFile(file, "r")) {
```

```

String line;

// Se necesita guardar el byte en el que inicia el registro, para poder actualizarlo
long byteOffset = 0; // Toma nota de la posición del byte en la que se encuentra la línea actual

while ((line = accessFile.readLine()) != null) {
    long currentOffset = byteOffset; // Guarda el byte en el que inicia el registro actual
    byteOffset = accessFile.getFilePointer(); // Actualiza la posición del byte a la del siguiente
registro

    String[] parts = line.split(SEPARADOR);
    if (parts[0].equals(nombre)) {
        return Optional.of(new ContactEntry(new Contacto(parts[0], parts[1]), currentOffset));
    }
}
return Optional.empty();
}

public synchronized List<ContactEntry> getAllRegistries() throws IOException {
    String line;
    List<ContactEntry> entries = new ArrayList<>();

    try (RandomAccessFile accessFile = new RandomAccessFile(file, "r")) {
        long byteOffset = 0;
        while ((line = accessFile.readLine()) != null) {
            long currentOffset = byteOffset;
            byteOffset = accessFile.getFilePointer();

            String[] parts = line.split(SEPARADOR);
            entries.add(new ContactEntry(new Contacto(parts[0], parts[1]), currentOffset));
        }
    }
    return entries;
}

public synchronized boolean deleteRegistry(String nombre) throws IOException {
    List<ContactEntry> entries = getAllRegistries();
    boolean deleted = false;
    try (RandomAccessFile accessFile = new RandomAccessFile(file, "rw")) {
        accessFile.setLength(0);
        for (ContactEntry entry : entries) {
            if (!entry.contacto().nombre().equals(nombre)) {
                accessFile.writeBytes(entry.contacto().nombre() + SEPARADOR + entry.contacto().telefono() +
System.lineSeparator());
            }
            else {
                deleted = true;
            }
        }
    }
    return deleted;
}

public record ContactEntry(Contacto contacto, long byteOffset) {}
}

```

Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/Archivo.java>

Código: MainUI.java

```
package Tareas.Tarea6.Partel1;

import javax.swing.*;
import javax.swing.event.TableModelEvent;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.IOException;

public class MainUI extends JFrame {

    private final Archivo archivo;
    private final DefaultTableModel tableModel;
    private final JTable table;

    public MainUI(Archivo archivo) {
        this.archivo = archivo;

        setTitle("Gestor de Contactos");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Se crea un DefaultTableModel que permite editar únicamente la columna de teléfono
        tableModel = new DefaultTableModel(new Object[]{"Nombre", "Teléfono"}, 0) {
            @Override
            public boolean isCellEditable(int row, int column) {
                return column != 0;
            }
        };

        table = new JTable(tableModel);

        // Al cambiar el número de teléfono de un contacto, se actualiza el registro
        tableModel.addTableModelListener(e -> {
            if (e.getType() == TableModelEvent.UPDATE) {
                editContact();
            }
        });

        JScrollPane scrollPane = new JScrollPane(table);

        // Se crean los botones de actualizar, añadir y eliminar
        JButton updateButton = new JButton("Actualizar");
        JButton addButton = new JButton("Añadir");
        JButton deleteButton = new JButton("Eliminar");

        updateButton.addActionListener(e -> updateContacts());
        addButton.addActionListener(e -> addContact());
        deleteButton.addActionListener(e -> deleteContact());

        JPanel buttonPanel = new JPanel();
        buttonPanel.add(updateButton);
        buttonPanel.add(addButton);
        buttonPanel.add(deleteButton);

        add(scrollPane, BorderLayout.CENTER);
        add(buttonPanel, BorderLayout.SOUTH);

        // Lee los contactos del archivo y los muestra en la tabla
        updateContacts();
    }
}
```

```

/**
 * Añade un contacto al archivo.
 * Si el nombre o el teléfono están vacíos, muestra un mensaje de advertencia.
 * Si el contacto se añade correctamente, actualiza la tabla.
 */
private void addContact() {
    JTextField nameField = new JTextField();
    JTextField phoneField = new JTextField();

    int option = JOptionPane.showConfirmDialog(this, new Object[]{"Nombre:", nameField, "Teléfono:",
phoneField},
        "Añadir Contacto", JOptionPane.OK_CANCEL_OPTION);

    if (option == JOptionPane.OK_OPTION) {
        String name = nameField.getText().trim();
        String phone = phoneField.getText().trim();

        if (!name.isEmpty() && !phone.isEmpty()) {
            try {
                archivo.addRegistry(new Contacto(name, phone));
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error al guardar el contacto.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this, "Todos los campos son obligatorios.", "Advertencia",
JOptionPane.WARNING_MESSAGE);
        }

        updateContacts();
    }
}

/**
 * Lee los contactos del archivo y los muestra en la tabla.
 */
private void updateContacts() {
    try {
        tableModel.setRowCount(0);
        archivo.getAllRegistries().forEach(entry -> tableModel
            .addRow(new Object[]{entry.contacto().nombre(), entry.contacto().telefono()}));
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

/**
 * En realidad, este método elimina el contacto y lo vuelve a añadir.
 */
private void editContact() {
    int selectedRow = table.getSelectedRow();
    if (selectedRow >= 0) {
        String name = (String) tableModel.getValueAt(selectedRow, 0);
        String phone = (String) tableModel.getValueAt(selectedRow, 1);
        try {
            if (archivo.deleteRegistry(name)) {
                archivo.addRegistry(new Contacto(name, phone));
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error al actualizar el contacto.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}

/**

```



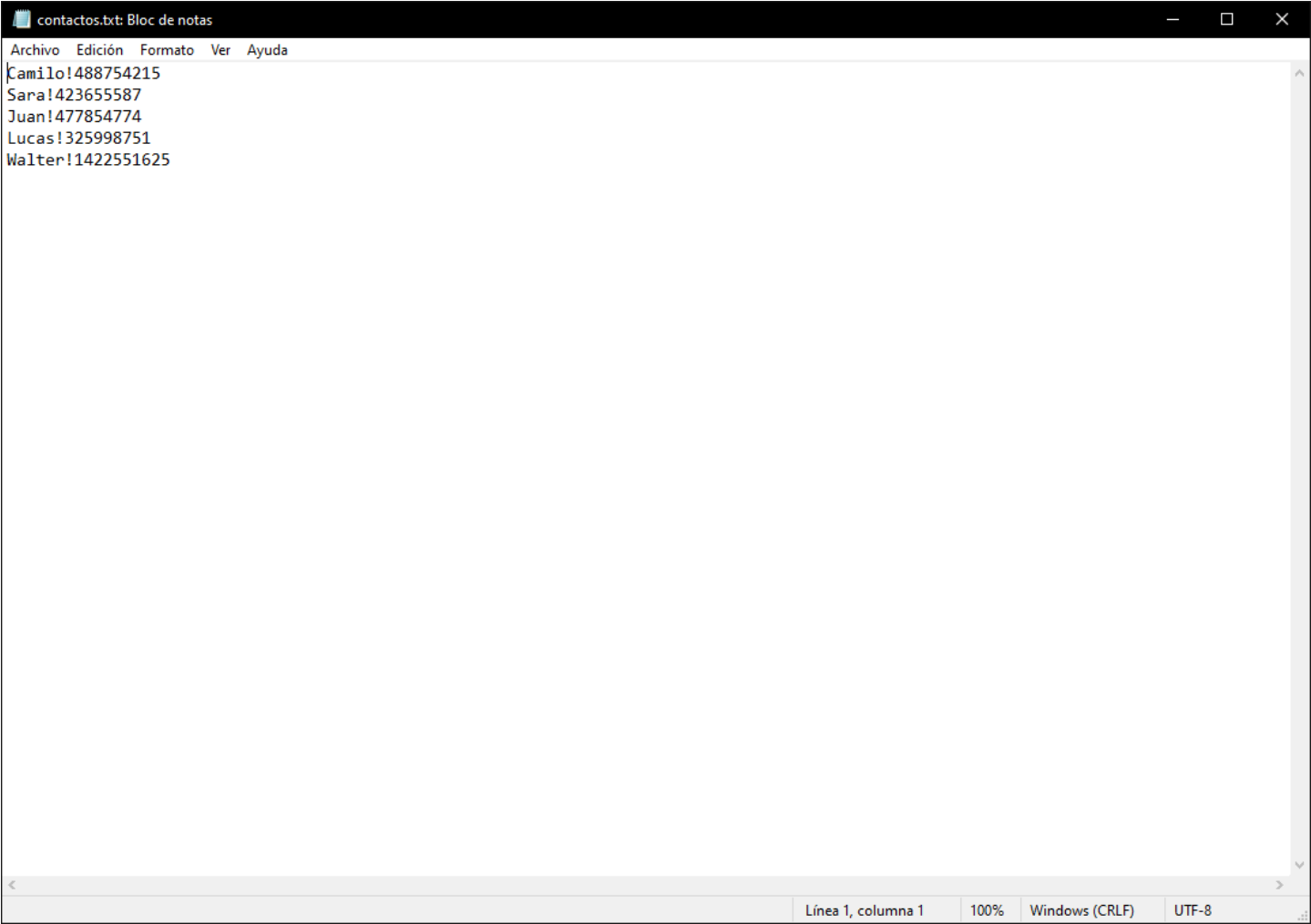
```

* Elimina un contacto del archivo.
* Si no se ha seleccionado un contacto, muestra un mensaje de advertencia.
* Si el contacto se elimina correctamente, actualiza la tabla.
*/
private void deleteContact() {
    int selectedRow = table.getSelectedRow();
    if (selectedRow >= 0) {
        String name = (String) tableModel.getValueAt(selectedRow, 0);
        int option = JOptionPane.showConfirmDialog(this, "¿Desea eliminar a " + name + "?", "Eliminar
Contacto",
            JOptionPane.YES_NO_OPTION);
        if (option == JOptionPane.YES_OPTION) {
            try {
                archivo.deleteRegistry(name);
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Error al eliminar el contacto.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
        updateContacts();
    } else {
        JOptionPane.showMessageDialog(this, "Seleccione un contacto para eliminar.", "Advertencia",
JOptionPane.WARNING_MESSAGE);
    }
}
}

```

Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/MainUI.java>

Imagen: Archivo.png



Enlace: <https://github.com/Simppplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/Archivo.png>

Código: Contacto.java

```
package Tareas.Tarea6.Partel;  
  
public record Contacto(String nombre, String telefono) {  
  
}
```

Enlace: <https://github.com/Simpplay/POO-2024-2/tree/master//Tareas/Tarea6/Parte1/Contacto.java>